Source: C:\Users\Admin\Desktop\CMC\ms1_emailIngestion\concurrent_storage\session_manager.py

```
"""
clear_session.py
Script ■■ clear session c■ và reset Redis state
"""
import sys
from concurrent_storage.redis_manager import get_redis_storage
from datetime import datetime, timezone


def clear_session(force=False):
    """
    Clear session hi■n t■i trong Redis

    Args:
        force: Clear without confirmation
    """
    print("=" * 70)
    print("CLEAR SESSION TOOL")
    print("=" * 70)

    redis = get_redis_storage()

    # Get current session
    session_data = redis.get_session_state()

    if not session_data:
        print("\n✓ No active session found")
        return True

    # Show current session info
    print("\nCurrent Session:")
    print(f"  Session ID: {session_data.get('session_id')}")
    print(f"  State: {session_data.get('state')}")
    print(f"  Start Time: {session_data.get('start_time')}")
    print(f"  Processed Count: {redis.get_processed_count()}")
    print(f"  Pending Count: {redis.get_pending_count()}")
    print(f"  Failed Count: {redis.get_failed_count()}")

    # Confirm
    if not force:
        print("\n■ This will:")
        print("  - Terminate current session")
        print("  - Save to history")
        print("  - Clear session state")
        print("\nNote: Processed email IDs will be kept (with TTL)")

        response = input("\nDo you want to continue? (yes/no): ")
        if response.lower() not in ['yes', 'y']:
            print("\n✗ Cancelled")
            return False

    # Clear session
    print("\n■ Clearing session...")

    # Update state to terminated
    redis.update_session_field("state", "terminated")
    redis.update_session_field("end_time", datetime.now(timezone.utc).isoformat())
    redis.update_session_field("termination_reason", "manual_clear")

    # Save to history
    final_state = redis.get_session_state()
    redis.save_session_history(final_state)
    print("  ✓ Saved to history")

    # Delete current session
    redis.delete_session()
    print("  ✓ Session cleared")

    print("\n✓ Session cleared successfully")
    print("=" * 70)
    return True
def show_session_info():
    """Show detailed session information"""
    print("=" * 70)
    print("SESSION INFORMATION")
    print("=" * 70)

    redis = get_redis_storage()

    # Current session
    session_data = redis.get_session_state()

    if session_data:
        print("\nCurrent Session:")
        for key, value in session_data.items():
            print(f"  {key}: {value}")
    else:
        print("\n✓ No active session")

    # Statistics
    print("\nStatistics:")
    print(f"  Processed Emails: {redis.get_processed_count()}")
    print(f"  Pending Emails: {redis.get_pending_count()}")
    print(f"  Failed Emails: {redis.get_failed_count()}")

    # Recent history
    print("\nRecent Sessions:")
    history = redis.get_session_history(limit=5)
    if history:
        for i, session in enumerate(history, 1):
            print(f"  {i}. {session.get('session_id')} - {session.get('state')}")
    else:
        print("  No history found")

    print("=" * 70)


def reset_all_data(confirm_phrase="RESET ALL"):
    """
    ■ DANGER: Reset t■t c■ data trong Redis
    Ch■ dùng cho development/testing
    """
    print("=" * 70)
    print("■ DANGER ZONE: RESET ALL DATA")
    print("=" * 70)

    print("\nThis will DELETE ALL data in Redis:")
    print("  - All sessions (current + history)")
    print("  - All processed email IDs")
    print("  - All pending/failed emails")
    print("  - All metrics and counters")
    print("  - All locks and subscriptions")

    print(f"\nType '{confirm_phrase}' to confirm:")
    response = input("> ")

    if response != confirm_phrase:
        print("\n✗ Cancelled (phrase didn't match)")
        return False

    redis = get_redis_storage()
    print("\n■ Deleting all data...")

    # Get all keys
    all_keys = redis.get_all_keys()
    print(f"  Found {len(all_keys)} keys")

    # Delete
    redis.flush_all(confirm=True)

    print("\n✓ All data deleted")
    print("=" * 70)
    return True


def main():
    """Main CLI"""
    import argparse
    parser = argparse.ArgumentParser(
        description="Session management tool for Email Ingestion Microservice"
    )

    parser.add_argument(
        'action',
        choices=['clear', 'info', 'reset'],
        help='Action to perform'
    )

    parser.add_argument(
        '--force',
        action='store_true',
        help='Skip confirmation prompts'
    )

    args = parser.parse_args()

    if args.action == 'clear':
        clear_session(force=args.force)
    elif args.action == 'info':
        show_session_info()
    elif args.action == 'reset':
        if args.force:
            reset_all_data()
        else:
            print("■ Use --force flag for reset (dangerous operation)")


if __name__ == "__main__":
    # If no args, show interactive menu
    if len(sys.argv) == 1:
        print("=" * 70)
        print("SESSION MANAGEMENT TOOL")
        print("=" * 70)
        print("\nChoose an action:")
        print("  1. Show session info")
        print("  2. Clear current session")
        print("  3. Reset all data (■ DANGER)")
        print("  0. Exit")

        choice = input("\nEnter choice: ")

        if choice == '1':
            show_session_info()
        elif choice == '2':
            clear_session()
        elif choice == '3':
            reset_all_data()
        elif choice == '0':
            print("Goodbye!")
        else:
            print("Please only choose from 0 to 3")
    else:
        main()
```