Pi-Hole - Add DNS to your Pi-Hole from Active Directory LDAPS

This pulls DNS node names from Active Directory over LDAPS, resolves A records against the AD DNS server (dig), imports into Pi-hole's custom.list, and restarts DNS.

Make sure you have the ROOT cert on the pi-hole side:

Certificate Trust Issue (by far the #1 cause with AD LDAPS) Active Directory DCs usually use certificates issued by your internal Enterprise CA (or a self-signed one). Your Pi-hole doesn't trust it by default.

Setup option:

- **Recommended (secure)**: Export your root CA certificate from AD and trust it on the Pi-hole.
  - On a Windows machine: Open MMC → Certificates → Trusted Root Certification Authorities → find your domain CA → Export as Base-64 encoded X.509 (.CER).
  - Copy the .cer file to Pi-hole: `/usr/local/share/ca-certificates/ad-root-ca.crt`
  - Run: `update-ca-certificates`

How it works:

- Queries both **DomainDnsZones** and **ForestDnsZones**
- Handles hostnames with multiple IP addresses
- Improved skipping of junk records
- Proper error handling and logging
- Compatible with **Pi-hole v6** (uses systemctl restart pihole-FTL or fallback)
- Uses your domain onling.com as example (easily changeable)
- Safer password handling via file
- Backup of previous records

```
#!/bin/bash
#
```

```
# ================================================================================
# AD-to-Pi-hole DNS Sync Script (via LDAPS with LDAP fallback)
# Pulls DNS node names from Active Directory, resolves A records,
# imports into Pi-hole's custom.list, and restarts DNS.
#
# Requirements: sudo apt install -y ldap-utils dnsutils
# Run as root (or via sudo). Recommended via cron.
# Created by Steve Ling 2026-04-15
#
# ================================================================================
set -euo pipefail

# ============================ ANSI COLORS ============================
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
CYAN='\033[0;36m'
MAGENTA='\033[0;35m'
BLUE='\033[0;34m'
NC='\033[0m' # No Color

success() { echo -e "${GREEN}[✓] $1${NC}"; }
info()    { echo -e "${CYAN}[i] $1${NC}"; }
warn()    { echo -e "${YELLOW}[!] $1${NC}"; }
error()   { echo -e "${RED}[✗] $1${NC}"; }
header()  { echo -e "${MAGENTA}=== $1 ===${NC}"; }
# ================================================================================

# ============================ CONFIGURATION ============================
AD_DC="sfl-dom-001.onling.com"
BIND_DN="CN=svcDevices,OU=Special_Users_Groups,DC=onling,DC=com"
BIND_PASS_FILE="/root/.ad_bind_pass"
DOMAIN="onling.com"

CUSTOM_LIST="/etc/pihole/custom.list"
BACKUP_DIR="/root/dns_backups"
```

```
# Optional: skip junk records (space-separated)
SKIP_PATTERNS="@ . _msdcs _tcp _udp _sites _gc _kerberos _kpasswd _ldap _smtp gc _domaincontroller"

# LDAPTLS settings (set to "never" only for testing)
export LDAPTLS_REQCERT=demand
# ============================================================

header "Starting Active Directory → Pi-hole DNS Sync"
echo -e "${BLUE}Started at: $(date '+%Y-%m-%d %H:%M:%S')${NC}\n"

# Create backup directory
mkdir -p "$BACKUP_DIR"
TIMESTAMP=$(date +"%Y%m%d-%H%M%S")
BACKUP_FILE="${BACKUP_DIR}/custom.list.${TIMESTAMP}.bak"

# Read password securely
if [ -f "$BIND_PASS_FILE" ]; then
    BIND_PASS=$(cat "$BIND_PASS_FILE")
    success "Password loaded from ${BIND_PASS_FILE}"
else
    error "Password file $BIND_PASS_FILE not found!"
    echo "Create it with:"
    echo "  echo 'YourPassword' > $BIND_PASS_FILE && chmod 600 $BIND_PASS_FILE"
    exit 1
fi

# Temporary files
TEMP_NAMES=$(mktemp)
TEMP_DOMAIN=$(mktemp)
TEMP_FOREST=$(mktemp)
TEMP_RECORDS=$(mktemp)
trap 'rm -f "$TEMP_NAMES" "$TEMP_DOMAIN" "$TEMP_FOREST" "$TEMP_RECORDS"' EXIT

# ====================== LDAP QUERY WITH FALLBACK ======================
DOMAIN_DN="DC=$(echo "$DOMAIN" | sed 's/\./,DC=/g')"

query_ldap() {
    local uri=$1
    local desc=$2
```

```
    info "Trying ${desc} (${uri})..."

    # DomainDnsZones
    info "  → Searching DomainDnsZones..."
    ldapsearch -H "${uri}" \
      -x -D "${BIND_DN}" -w "${BIND_PASS}" \
      -b "CN=MicrosoftDNS,DC=DomainDnsZones,${DOMAIN_DN}" \
      -s sub "(objectClass=dnsNode)" dc 2>"$TEMP_DOMAIN.err" | \
      grep -E "^dc:" | awk '{print $2}' | sort -u > "$TEMP_DOMAIN"

    # ForestDnsZones
    info "  → Searching ForestDnsZones..."
    ldapsearch -H "${uri}" \
      -x -D "${BIND_DN}" -w "${BIND_PASS}" \
      -b "CN=MicrosoftDNS,DC=ForestDnsZones,${DOMAIN_DN}" \
      -s sub "(objectClass=dnsNode)" dc 2>"$TEMP_FOREST.err" | \
      grep -E "^dc:" | awk '{print $2}' | sort -u > "$TEMP_FOREST"

    # Check if we got results
    if [ -s "$TEMP_DOMAIN" ] || [ -s "$TEMP_FOREST" ]; then
        success "LDAP query successful via ${desc}"
        return 0
    else
        warn "No results or error via ${desc}. Checking error output..."
        cat "$TEMP_DOMAIN.err" "$TEMP_FOREST.err" 2>/dev/null | tail -n 10
        return 1
    fi
}

# Try LDAPS first, then fallback to LDAP
if query_ldap "ldaps://${AD_DC}" "LDAPS (secure)"; then
    LDAP_SUCCESS=1
else
    warn "LDAPS failed. Falling back to plain LDAP (port 389)..."
    if query_ldap "ldap://${AD_DC}" "LDAP (fallback)"; then
        LDAP_SUCCESS=1
    else
        error "Both LDAPS and LDAP queries failed. Check credentials, network, firewall, and AD DC availability."
```

```
        exit 1
    fi
fi

# Combine and deduplicate
cat "$TEMP_DOMAIN" "$TEMP_FOREST" 2>/dev/null | sort -u > "$TEMP_NAMES"
NAME_COUNT=$(wc -l < "$TEMP_NAMES")

success "Found ${NAME_COUNT} unique DNS node names from AD."

if [ "$NAME_COUNT" -eq 0 ]; then
    error "No DNS nodes found. Please verify bind credentials and search bases."
    exit 1
fi

# ======================== RESOLVE A RECORDS ========================
header "Resolving A records against ${AD_DC}"
> "$TEMP_RECORDS"

while IFS= read -r name || [ -n "$name" ]; do
    [ -z "$name" ] && continue

    # Skip unwanted patterns
    if [[ " ${SKIP_PATTERNS} " == *" ${name} "* ]]; then
        info "  Skipping: ${name}"
        continue
    fi

    resolved=false
    for fqdn in "${name}.${DOMAIN}" "${name}"; do
        info "  Resolving: ${fqdn}"

        mapfile -t ips < <(dig @"${AD_DC}" +short +time=3 +tries=2 "${fqdn}" A 2>/dev/null | \
                    grep -E '^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$')

        if [ ${#ips[@]} -gt 0 ]; then
            for ip in "${ips[@]}"; do
                echo "${ip} ${name}" >> "$TEMP_RECORDS"
                if [ "${fqdn}" != "${name}" ]; then
```

```
                echo "${ip} ${fqdn}" >> "$TEMP_RECORDS"
            fi
        done
        success "    → ${#ips[@]} IP(s) found for ${fqdn}"
        resolved=true
        break
      fi
    done

    if ! $resolved; then
      warn "    No A record found for ${name}"
    fi
done < "$TEMP_NAMES"

sort -u "$TEMP_RECORDS" -o "$TEMP_RECORDS"
RECORD_COUNT=$(wc -l < "$TEMP_RECORDS")

success "Resolved ${RECORD_COUNT} A record entries."

# ====================== BACKUP & IMPORT ======================
header "Updating Pi-hole custom.list"

echo "Backing up current custom.list..."
if [ -f "$CUSTOM_LIST" ]; then
    cp -f "$CUSTOM_LIST" "$BACKUP_FILE"
    success "Backup created: ${BACKUP_FILE}"
else
    warn "No existing custom.list found (first run?)"
fi

info "Writing ${RECORD_COUNT} new records to ${CUSTOM_LIST}..."
cp "$TEMP_RECORDS" "$CUSTOM_LIST"
chown pihole:pihole "$CUSTOM_LIST" 2>/dev/null || true
chmod 644 "$CUSTOM_LIST"
success "Custom list updated."

# ====================== RESTART PI-HOLE ======================
#header "Restarting Pi-hole DNS"
#if command -v systemctl >/dev/null 2>&1; then
```

```
#    systemctl restart pihole-FTL && success "Restarted via: systemctl restart pihole-FTL" || error "Failed to restart pihole-FTL"
#else
#    service pihole-FTL restart && success "Restarted via: service pihole-FTL restart" || error "Failed to restart pihole-FTL"
#fi
header "Reloading Pi-hole DNS Lists"
#pihole reloaddns reload-lists && success "Reloaded DNS lists smoothly" || error "Failed to reload lists"
pihole reloadlists && success "Reloaded DNS lists smoothly" || error "Failed to reload lists"
pihole reloaddns && success "Reloaded DNS smoothly" || error "Failed to reload DNS"

# ====================== SUMMARY ======================
header "Sync Completed Successfully"
echo -e "${GREEN}Imported ${RECORD_COUNT} entries from Active Directory.${NC}"
echo -e "Total unique hostnames processed: ${NAME_COUNT}"
echo -e "Backup: ${BACKUP_FILE}"
echo ""
echo -e "${CYAN}Quick verification commands:${NC}"
echo "  pihole -q <hostname>"
echo "  sudo journalctl -u pihole-FTL -n 50 --no-pager"
echo ""
echo -e "${BLUE}Finished at: $(date '+%Y-%m-%d %H:%M:%S')${NC}"

exit 0
```

How to Deploy

1. Save as /usr/local/bin/ad-pihole-dns-sync.sh
2. Make executable: chmod +x /usr/local/bin/ad-pihole-dns-sync.sh
3. Create the password file securely:

   ```
   echo 'YourActualPassword' > /root/.ad_bind_pass
   chmod 600 /root/.ad_bind_pass
   ```

4. Update AD_DC and BIND_DN if needed.
5. Test: sudo /usr/local/bin/ad-pihole-dns-sync.sh
6. Add to cron (example: every hour):

   ```
   0 * * * * /usr/local/bin/ad-pihole-dns-sync.sh >> /var/log/ad-dns-sync.log 2>&1
   ```

Revision #3
Created 15 April 2026 22:35:01 by Steve Ling

Updated 11 August 2026 17:56:27 by Steve Ling