User Migration Guide for WP Subdomain Login

Migrating Existing Users to Separate Database Setup

This guide walks you through the process of migrating existing WordPress users from your main site to a subdomain with a separate database, ensuring seamless authentication with the WP Subdomain Login plugin.

Table of Contents

Understanding the Migration

What Needs to Match

For the WP Subdomain Login plugin to work with separate databases, users must have:

Field	Must Match?	Notes
Username	✅ YES	Critical - used for authentication token
Password	✅ YES	Must be identical (same hash)
User ID	❌ NO	Can be different on each site
Email	⚠ OPTIONAL	Only if users login with email
Display Name	❌ NO	Can be different
User Role	❌ NO	Can assign different roles per site
User Meta	❌ NO	Stored separately on each site

Migration Scenarios

Scenario 1: New Subdomain Setup

- Main site has existing users
- Subdomain is new with no users
- **Action**: Export users from main site, import to subdomain

Scenario 2: Both Sites Have Users

- Main site and subdomain both have existing users
- Some users may overlap
- **Action**: Merge user lists, handle conflicts

Scenario 3: Gradual Migration

- Migrate users in batches
- Keep both systems running temporarily
- **Action**: Incremental sync with verification

Pre-Migration Checklist

1. Backup Everything

```
# Backup main site database
mysqldump -u username -p main_database > main_site_backup.sql

# Backup subdomain database
mysqldump -u username -p subdomain_database > subdomain_backup.sql

# Backup WordPress files (both sites)
tar -czf main_site_files.tar.gz /path/to/main/site
tar -czf subdomain_files.tar.gz /path/to/subdomain
```

2. Document Current State

Create a spreadsheet with:

- Total user count on main site
- Total user count on subdomain (if any)
- User roles distribution
- Any custom user meta fields you need to preserve

3. Install Required Plugins

On Main Site:

- WP Subdomain Login (v1.0.1+)
- Export Users to CSV (optional, for manual review)

On Subdomain:

- WP Subdomain Login (v1.0.1+)
- Import Users from CSV (optional, for manual import)

4. Test Environment

Set up a staging environment to test the migration before running on production.

Migration Methods

Method 1: SQL Direct Export/Import (Recommended)

Best for: Large user bases (100+ users), technical users

Pros:

- Fast and efficient
- Preserves password hashes
- Handles bulk migration

Cons:

- Requires database access
- More technical

Method 2: CSV Export/Import

Best for: Small to medium user bases (< 100 users), non-technical users

Pros:

- User-friendly
- Good for reviewing data before import
- Easy to modify in spreadsheet

Cons:

- Cannot export password hashes (users need to reset)
- Manual process

Method 3: Custom PHP Script

Best for: Complex migrations with custom requirements

Pros:

- Full control over the process
- Can handle custom logic
- Can sync additional data

Cons:

- Requires PHP knowledge
- Time to develop and test

Step-by-Step Migration Process

Method 1: SQL Direct Export/Import

Step 1: Export Users from Main Site

```
-- Connect to main site database
-- Export users table
SELECT * FROM wp_users
INTO OUTFILE '/tmp/users_export.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';

-- Export user metadata (optional)
SELECT * FROM wp_usermeta
WHERE user_id IN (SELECT ID FROM wp_users)
INTO OUTFILE '/tmp/usermeta_export.csv'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n';
```

Step 2: Prepare Import Script

Create a file called `import_users.php` and upload to your subdomain's root:

```php
<?php
/**
 * User Migration Script for WP Subdomain Login
 *
 * IMPORTANT: Delete this file after migration is complete!
 */

// Load WordPress
require_once('wp-load.php');

// Security check - only allow from localhost or specific IP
if (!in_array($_SERVER['REMOTE_ADDR'], ['127.0.0.1', 'YOUR_IP_HERE'])) {
    die('Access denied');
}

// Prevent timeout
set_time_limit(0);

// Database connection for main site
$main_db = new wpdb(
    'main_db_user',      // Main site DB username
    'main_db_password',  // Main site DB password
    'main_db_name',      // Main site DB name
    'main_db_host'       // Main site DB host
);

// Get all users from main site
$main_users = $main_db->get_results("SELECT * FROM wp_users");

$results = [
    'total' => count($main_users),
    'created' => 0,
    'skipped' => 0,
    'errors' => []
];

foreach ($main_users as $user) {
    // Check if user already exists on subdomain
    $existing_user = get_user_by('login', $user->user_login);

    if ($existing_user) {
        $results['skipped']++;
        echo "Skipped: {$user->user_login} (already exists)<br>";
        continue;
    }
```

```
    // Create user on subdomain with same credentials
    $user_data = [
        'user_login'    => $user->user_login,
        'user_pass'     => $user->user_pass,  // Already hashed
        'user_email'    => $user->user_email,
        'user_url'      => $user->user_url,
        'display_name'  => $user->display_name,
        'nickname'      => $user->user_nicename,
        'first_name'    => '',  // Will be set from meta
        'last_name'     => '',  // Will be set from meta
        'description'   => '',
        'user_registered' => $user->user_registered,
    ];

    // Insert user with pre-hashed password
    global $wpdb;

    // Temporarily remove password hashing
    remove_all_filters('pre_user_pass');

    $user_id = wp_insert_user($user_data);

    if (is_wp_error($user_id)) {
        $results['errors'][] = "Error creating {$user->user_login}: " .
$user_id->get_error_message();
        echo "Error: {$user->user_login}<br>";
        continue;
    }

    // Update password hash directly (bypass WordPress hashing)
    $wpdb->update(
        $wpdb->users,
        ['user_pass' => $user->user_pass],
        ['ID' => $user_id]
    );

    // Copy user meta
    $user_meta = $main_db->get_results($wpdb->prepare(
        "SELECT meta_key, meta_value FROM wp_usermeta WHERE user_id = %d",
        $user->ID
    ));

    foreach ($user_meta as $meta) {
        // Skip certain meta keys that should be site-specific
        if (in_array($meta->meta_key, ['wp_capabilities', 'wp_user_level']))
```

```
{
            continue;
        }
        update_user_meta($user_id, $meta->meta_key, maybe_unserialize($meta->meta_value));
    }

    $results['created']++;
    echo "Created: {$user->user_login} (ID: {$user_id})<br>";
}

echo "<hr>";
echo "<h2>Migration Complete!</h2>";
echo "<p>Total users: {$results['total']}</p>";
echo "<p>Created: {$results['created']}</p>";
echo "<p>Skipped: {$results['skipped']}</p>";

if (!empty($results['errors'])) {
    echo "<h3>Errors:</h3>";
    foreach ($results['errors'] as $error) {
        echo "<p style='color: red;'>{$error}</p>";
    }
}

echo "<p style='color: red; font-weight: bold;'>IMPORTANT: Delete this file now!</p>";
?>
```

Step 3: Run the Migration

1. Upload `import_users.php` to your subdomain root
2. Update the database credentials in the script
3. Visit: `https://subdomain.example.com/import_users.php`
4. Wait for completion
5. **DELETE the script file immediately**

Step 4: Assign User Roles

```php
<?php
/**
 * Assign roles to migrated users
 * Run this after import_users.php
 */

require_once('wp-load.php');

// Get all users
$users = get_users(['fields' => ['ID', 'user_login']]);

foreach ($users as $user) {
    $user_obj = new WP_User($user->ID);

    // Remove all roles
    $user_obj->set_role('');

    // Assign default role (adjust as needed)
    $user_obj->add_role('subscriber');

    echo "Updated role for: {$user->user_login}<br>";
}

echo "Role assignment complete!";
?>
```

Method 2: CSV Export/Import

Step 1: Install Export Plugin

On main site, install “Export Users to CSV” plugin.

Step 2: Export Users

1. Go to **Users → Export Users**
2. Select all user roles
3. **Important**: Check “Include password hashes” (if available)
4. Download CSV file

Step 3: Prepare CSV

Open the CSV and ensure these columns exist:

- `user_login` (required)
- `user_email` (required)
- `user_pass` (password hash - if not available, users will need to reset)
- `display_name`
- `first_name`
- `last_name`
- `role`

Step 4: Import to Subdomain

Option A: Using Plugin

1. Install “Import Users from CSV” plugin on subdomain
2. Go to **Users → Import from CSV**
3. Upload your CSV file
4. Map columns
5. **Important**: Select “Update existing users” if needed
6. Run import

Option B: Manual PHP Script

```php
<?php
/**
 * CSV User Import Script
 */

require_once('wp-load.php');

$csv_file = 'users_export.csv';

if (!file_exists($csv_file)) {
    die('CSV file not found');
}

$handle = fopen($csv_file, 'r');
$header = fgetcsv($handle); // Get column headers

$results = ['created' => 0, 'skipped' => 0, 'errors' => []];

while (($row = fgetcsv($handle)) !== false) {
    $user_data = array_combine($header, $row);

    // Check if user exists
    if (username_exists($user_data['user_login'])) {
        $results['skipped']++;
        continue;
    }

    // Create user
    $user_id = wp_insert_user([
        'user_login' => $user_data['user_login'],
        'user_email' => $user_data['user_email'],
        'user_pass'  => wp_generate_password(), // Random password
        'display_name' => $user_data['display_name'] ?? '',
        'first_name' => $user_data['first_name'] ?? '',
        'last_name'  => $user_data['last_name'] ?? '',
        'role'       => $user_data['role'] ?? 'subscriber',
    ]);

    if (is_wp_error($user_id)) {
        $results['errors'][] = $user_data['user_login'];
        continue;
    }

    // If password hash is available, update it
    if (!empty($user_data['user_pass']) && strlen($user_data['user_pass']) >
```

```
20) {
        global $wpdb;
        $wpdb->update(
            $wpdb->users,
            ['user_pass' => $user_data['user_pass']],
            ['ID' => $user_id]
        );
    }

    $results['created']++;
}

fclose($handle);

echo "Import complete!\n";
echo "Created: {$results['created']}\n";
echo "Skipped: {$results['skipped']}\n";
echo "Errors: " . count($results['errors']);
?>
```

Password Synchronization

Challenge

The WP Subdomain Login plugin requires users to have **identical passwords** on both sites. This is challenging because:

1. WordPress password hashes are one-way (cannot be decrypted)
2. Users changing passwords on one site won't automatically update the other

Solutions

Solution 1: Migrate Password Hashes (Recommended)

Use the SQL method above to copy the exact password hashes from main site to subdomain. This ensures passwords match immediately.

Solution 2: Force Password Reset

If you cannot migrate password hashes:

1. Import users without passwords
2. Use WordPress “Send Password Reset” feature
3. Users set new passwords on subdomain
4. **Important**: Inform users to use the same password on both sites

Solution 3: Password Sync Plugin (Advanced)

Create a custom plugin to sync passwords between sites:

```php
<?php
/**
 * Plugin Name: Password Sync for Subdomain Login
 * Description: Syncs password changes between main site and subdomain
 */

// When user changes password on main site, update subdomain
add_action('password_reset', 'sync_password_to_subdomain', 10, 2);
add_action('profile_update', 'sync_password_on_profile_update', 10, 2);

function sync_password_to_subdomain($user, $new_pass) {
    sync_password_between_sites($user->user_login, $new_pass);
}

function sync_password_on_profile_update($user_id, $old_user_data) {
    $user = get_userdata($user_id);

    // Check if password was changed
    if ($user->user_pass !== $old_user_data->user_pass) {
        // Get the new password hash
        $new_hash = $user->user_pass;
        sync_password_hash_to_subdomain($user->user_login, $new_hash);
    }
}

function sync_password_hash_to_subdomain($username, $password_hash) {
    // Connect to subdomain database
    $subdomain_db = new wpdb(
        'subdomain_db_user',
        'subdomain_db_password',
        'subdomain_db_name',
        'subdomain_db_host'
    );

    // Get user ID on subdomain
    $user = $subdomain_db->get_row($subdomain_db->prepare(
        "SELECT ID FROM wp_users WHERE user_login = %s",
        $username
    ));

    if ($user) {
        // Update password hash
        $subdomain_db->update(
            'wp_users',
            ['user_pass' => $password_hash],
```

```
            ['ID' => $user->ID]
        );
    }
}
?>
```

Install this plugin on BOTH sites to keep passwords in sync.

Testing & Verification

Pre-Launch Testing Checklist

Create a test spreadsheet with sample users:

Username	Main Site Login	Subdomain Login	Token Valid	Notes
testuser1	✅	✅	✅	Success
testuser2	✅	❌	❌	Check password
testuser3	❌	✅	❌	User not on main

Test Procedure

1. **Test 1: User Exists on Both Sites**
    - Login via main site form
    - Verify redirect to subdomain
    - Verify logged in on subdomain
    - Check user role and permissions
2. **Test 2: User Only on Main Site**
    - Attempt login
    - Should see error: “User account not found on this site”
    - Create user on subdomain
    - Retry login - should succeed

3. **Test 3: Password Mismatch**
    - Change password on main site only
    - Attempt login
    - Should fail authentication
    - Update password on subdomain
    - Retry - should succeed
4. **Test 4: Token Expiry**
    - Initiate login
    - Wait 6 minutes (token expires after 5 minutes)
    - Complete login
    - Should see “Invalid or expired authentication token”
5. **Test 5: Different User IDs**
    - Find a user with different IDs on each site
    - Login via main site
    - Verify correct user is logged in on subdomain
    - Check user data matches

Verification SQL Queries

```sql
-- Check if usernames match
-- Run on main site
SELECT user_login, user_email FROM wp_users ORDER BY user_login;

-- Run on subdomain - compare results
SELECT user_login, user_email FROM wp_users ORDER BY user_login;

-- Check password hashes match
-- Main site
SELECT user_login, user_pass FROM wp_users WHERE user_login = 'testuser';

-- Subdomain - hash should be identical
SELECT user_login, user_pass FROM wp_users WHERE user_login = 'testuser';

-- Check for duplicate emails (can cause issues)
SELECT user_email, COUNT(*) as count
FROM wp_users
GROUP BY user_email
HAVING count > 1;
```

Troubleshooting

Issue 1: “User account not found on this site”

Cause: User doesn’t exist on subdomain

Solution:

```sql
-- Check if user exists on subdomain
SELECT * FROM wp_users WHERE user_login = 'username';

-- If not found, create user or run migration again
```

Issue 2: Authentication fails but user exists

Cause: Password hashes don’t match

Solution:

```
-- Compare password hashes
-- Main site
SELECT user_pass FROM wp_users WHERE user_login = 'username';

-- Subdomain
SELECT user_pass FROM wp_users WHERE user_login = 'username';

-- If different, copy hash from main to subdomain
UPDATE wp_users
SET user_pass = 'hash_from_main_site'
WHERE user_login = 'username';
```

Issue 3: Wrong user logged in on subdomain

Cause: Plugin using user ID instead of username (shouldn't happen in v1.0.1+)

Solution: Ensure you're using plugin version 1.0.1 or later

Issue 4: Some users can login, others cannot

Cause: Incomplete migration

Solution:

```
-- Find users on main site not on subdomain
SELECT m.user_login
FROM main_db.wp_users m
LEFT JOIN subdomain_db.wp_users s ON m.user_login = s.user_login
WHERE s.user_login IS NULL;

-- Re-run migration for these users
```

Issue 5: User roles are wrong on subdomain

Cause: Roles not migrated or different role names

Solution:

```php
// Reset user role
$user = get_user_by('login', 'username');
$user->set_role('desired_role');
```

Post-Migration Best Practices

1. Monitor Login Activity

Install a login monitoring plugin to track:

- Failed login attempts
- Successful cross-site logins
- Token expiration issues

2. User Communication

Send an email to all users:

```
Subject: Important: Login System Update

Dear [User],

We've updated our login system to improve your experience. Here's what you
need to know:

1. You can now login from our main site and automatically access [subdomain
name]
2. Your username and password remain the same
3. If you experience any login issues, please reset your password

If you need assistance, contact support@example.com

Best regards,
[Your Team]
```

3. Implement Password Sync

Install the password sync plugin (see above) to keep passwords synchronized going forward.

4. Regular Audits

Schedule monthly audits:

```
-- Count users on each site
SELECT COUNT(*) FROM wp_users; -- Run on both sites

-- Find users with mismatched passwords (requires custom table)
-- You'll need to create a sync log table to track this
```

5. Backup Strategy

- Daily database backups
- Keep migration scripts for reference
- Document any custom modifications

6. User Self-Service

Create a help page explaining:

- How to use the new login system
- What to do if login fails
- How to reset password on both sites

Advanced: Automated Sync System

For ongoing synchronization, consider implementing:

Real-Time Sync via REST API

```php
<?php
/**
 * Real-time user sync between sites
 */

// On main site - send updates to subdomain
add_action('user_register', 'sync_new_user_to_subdomain');
add_action('profile_update', 'sync_user_update_to_subdomain');
add_action('delete_user', 'sync_user_deletion_to_subdomain');

function sync_new_user_to_subdomain($user_id) {
    $user = get_userdata($user_id);

    $response = wp_remote_post('https://subdomain.example.com/wp-
json/custom/v1/sync-user', [
        'body' => json_encode([
            'action' => 'create',
            'user_login' => $user->user_login,
            'user_email' => $user->user_email,
            'user_pass' => $user->user_pass,
            'display_name' => $user->display_name,
        ]),
        'headers' => [
            'Content-Type' => 'application/json',
            'X-API-Key' => 'your-secure-api-key',
        ],
    ]);
}

// On subdomain - receive and process updates
add_action('rest_api_init', function() {
    register_rest_route('custom/v1', '/sync-user', [
        'methods' => 'POST',
        'callback' => 'handle_user_sync',
        'permission_callback' => 'verify_sync_api_key',
    ]);
});

function verify_sync_api_key($request) {
    $api_key = $request->get_header('X-API-Key');
    return $api_key === 'your-secure-api-key';
}
```

```php
function handle_user_sync($request) {
    $data = $request->get_json_params();

    switch ($data['action']) {
        case 'create':
            // Create user on subdomain
            wp_insert_user([
                'user_login' => $data['user_login'],
                'user_email' => $data['user_email'],
                'user_pass' => $data['user_pass'],
                'display_name' => $data['display_name'],
            ]);
            break;

        case 'update':
            // Update existing user
            break;

        case 'delete':
            // Delete user
            break;
    }

    return new WP_REST_Response(['success' => true], 200);
}
?>
```

Conclusion

Migrating users to a separate database setup requires careful planning and execution. Key takeaways:

1. **Always backup** before starting
2. **Test thoroughly** in staging environment
3. **Verify password hashes** are copied correctly
4. **Communicate with users** about the change
5. **Monitor** for issues post-migration
6. **Implement sync** for ongoing password changes

For additional support, contact Douglas LaMar at LaMarSOFT.com.

Document Version: 1.0
Last Updated: January 7, 2026
Author: Douglas LaMar - LaMarSOFT.com