JavaScript Widget Integration Guide

Generated on July 2, 2026

Ticket System SSO Integration Guide

Overview

This guide explains how to integrate Single Sign-On (SSO) between your main website and the Central Tickets system, allowing users to seamlessly authenticate from the logout landing page.

How It Works

When a user clicks "Sign In with [Your Company]" on the logout landing page:

1. User is redirected to your website's SSO endpoint: `https://yourwebsite.com/ticket-sso`
2. Your website checks if the user is logged in
3. If logged in, your website calls the Ticket System API with user credentials
4. The API validates your tenant credentials and generates an encrypted authentication token
5. User is redirected back to the ticket system with the token
6. Ticket system validates the token and creates a session
7. User lands on their dashboard, fully authenticated

Required Parameters

The logout page passes these URL parameters to your SSO endpoint:

- `redirect_url`: Where to send the user after authentication (e.g., `https://ticket.yourcompany.com/yourcompany/dashboard`)
- `callback_api`: The API endpoint to call for authentication (e.g., `https://tickets.flare99.com/api/auth/redirect`)
- `tenant`: Your tenant slug (e.g., `yourcompany`) — must be passed as `tenant` in the POST body.
- `X-API-Key` / `X-API-Secret`: Required headers for the redirect endpoint.

Implementation Options

Option 1: Server-Side (PHP - Recommended)

Create a file at `https://yourwebsite.com/ticket-sso.php`:

```php
<?php
// Check if user is logged in
session_start();
if (!isset($_SESSION['user_id'])) {
    // Redirect to login with return URL
    $returnUrl = urlencode($_SERVER['REQUEST_URI']);
    header("Location: /login?return=" . $returnUrl);
    exit;
}

// Get user data from your session/database
$user = getUserById($_SESSION['user_id']); // Your user retrieval function

// Get parameters
$redirectUrl = $_GET['redirect_url'] ?? '';
$callbackApi = $_GET['callback_api'] ?? '';
$tenantSlug = $_GET['tenant'] ?? '';

// Validate parameters
if (empty($redirectUrl) || empty($callbackApi)) {
    die('Invalid SSO request');
}

// Call ticket system API
$userData = [
    'email' => $user['email'],
    'name' => $user['name'],
    'redirect_url' => $redirectUrl
];

// Include API credentials when calling the redirect endpoint
$headers = [
    'X-API-Key' => getenv('TICKET_SYSTEM_API_KEY'),
    'X-API-Secret' => getenv('TICKET_SYSTEM_API_SECRET'),
];

$ch = curl_init($callbackApi);
curl_setopt_array($ch, [
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_POST => true,
    CURLOPT_HTTPHEADER => [
        'Content-Type: application/json',
        'Accept: application/json'
    ],
    CURLOPT_POSTFIELDS => json_encode($userData),
    CURLOPT_FOLLOWLOCATION => false,
    CURLOPT_HEADER => true
]);

$response = curl_exec($ch);
$httpCode = curl_getinfo($ch, CURLINFO_HTTP_CODE);

// Extract Location header from response
if ($httpCode === 302) {
    preg_match('/Location: (.+)/', $response, $matches);
    if (isset($matches[1])) {
        $redirectTo = trim($matches[1]);
        curl_close($ch);
        header("Location: " . $redirectTo);
```

```
        exit;
    }
}

curl_close($ch);
die('Authentication failed');
?>
```

Option 2: Server-Side (Laravel)

Route (routes/web.php):

```
Route::get('/ticket-sso', [SSOController::class, 'handleTicketSSO'])
    ->middleware('auth')
    ->name('ticket.sso');
```

Controller (app/Http/Controllers/SSOController.php):

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Http;
use Illuminate\Support\Facades\Auth;

class SSOController extends Controller
{
    public function handleTicketSSO(Request $request)
    {
        // Validate user is logged in
        if (!Auth::check()) {
            return redirect('/login?return=' . urlencode($request->fullUrl()));
        }

        // Get parameters
        $redirectUrl = $request->query('redirect_url');
        $callbackApi = $request->query('callback_api');
        $tenantSlug = $request->query('tenant');

        // Validate parameters
        if (!$redirectUrl || !$callbackApi || !$tenantSlug) {
            abort(400, 'Invalid SSO request parameters');
        }

        // Get authenticated user
        $user = Auth::user();

        // Call ticket system API
        try {
            $response = Http::withHeaders([
                'Accept' => 'application/json',
                'Content-Type' => 'application/json',
            ])->post($callbackApi, [
                'email' => $user->email,
                'name' => $user->name,
                'redirect_url' => $redirectUrl,
            ]);

            // Check if response is a redirect
            if ($response->redirect()) {
                return redirect($response->header('Location'));
            }

            // If JSON response with redirect_url
            $data = $response->json();
            if (isset($data['redirect_url'])) {
                return redirect($data['redirect_url']);
            }

            abort(500, 'Invalid API response');

        } catch (\Exception $e) {
            \Log::error('Ticket SSO failed', [
                'error' => $e->getMessage(),
                'user_id' => $user->id,
            ]);
```

```
            return view('ticket-sso-error', [
                'message' => 'Unable to authenticate with ticket system. Please try again.'
            ]);
        }
    }
}
```

Option 3: Client-Side (JavaScript - Advanced)

Create a page at `https://yourwebsite.com/ticket-sso.html`:

```
<!DOCTYPE html>
<html>
<head>
    <title>Signing you in...</title>
    <style>
        body {
            font-family: system-ui;
            display: flex;
            justify-content: center;
            align-items: center;
            min-height: 100vh;
            background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
        }
        .container {
            background: white;
            padding: 2rem;
            border-radius: 1rem;
            text-align: center;
            box-shadow: 0 20px 60px rgba(0,0,0,0.3);
        }
        .spinner {
            border: 4px solid #f3f3f3;
            border-top: 4px solid #667eea;
            border-radius: 50%;
            width: 50px;
            height: 50px;
            animation: spin 1s linear infinite;
            margin: 0 auto 1rem;
        }
        @keyframes spin {
            0% { transform: rotate(0deg); }
            100% { transform: rotate(360deg); }
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="spinner"></div>
        <h1>Signing you in...</h1>
        <p id="status">Authenticating with ticket system...</p>
    </div>

    <script>
        // Get URL parameters
        const urlParams = new URLSearchParams(window.location.search);
        const redirectUrl = urlParams.get('redirect_url');
        const callbackApi = urlParams.get('callback_api');
        const tenantSlug = urlParams.get('tenant');

        // Get user data from your authentication system
        // This example assumes you have a function that returns logged-in user data
        async function getUserData() {
            // Replace with your actual user data retrieval
            const response = await fetch('/api/user', {
                credentials: 'include'
            });
            if (!response.ok) return null;
            return await response.json();
        }
```

```
        async function authenticate() {
            try {
                // Get current user
                const user = await getUserData();

                if (!user) {
                    // Not logged in - redirect to login
                    window.location.href = '/login?return=' + encodeURIComponent(window.locatio
                    return;
                }

                // Call ticket system API
                const response = await fetch(callbackApi, {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json',
                        'Accept': 'application/json'
                    },
                    body: JSON.stringify({
                        email: user.email,
                        name: user.name,
                        redirect_url: redirectUrl
                    })
                });

                if (response.redirected) {
                    window.location.href = response.url;
                } else {
                    const data = await response.json();
                    if (data.redirect_url) {
                        window.location.href = data.redirect_url;
                    } else {
                        throw new Error('No redirect URL received');
                    }
                }

            } catch (error) {
                console.error('SSO Error:', error);
                document.querySelector('.container').innerHTML = `
                    <h1 style="color: #e53e3e;">Authentication Failed</h1>
                    <p>Unable to sign you in. Please try again.</p>
                    <button onclick="history.back()" style="margin-top: 1rem; padding: 0.75rem
                        Go Back
                    </button>
                `;
            }
        }

        // Start authentication
        authenticate();
    </script>
</body>
</html>
```

SSO Redirect to a Specific Ticket

When your system creates a ticket on behalf of a user (server-to-server via the API), you can redirect that user directly to the newly created ticket — rather than their dashboard — by including a `redirect_url` in the SSO payload.

How It Works

1. Your server creates a ticket via `POST /api/v2/tickets`
2. The API response includes `ticket_url`, `sso_redirect.callback_url`, and a `jwt_payload` template
3. Your server generates a JWT using the provided payload (which includes `redirect_url` pointing to the ticket)
4. You redirect the user's browser to `{callback_url}?token={jwt}`
5. The ticket system authenticates the user and lands them directly on the ticket page

Step 1: Create the Ticket via API

```
curl -X POST https://tickets.flare99.com/api/v2/tickets \
  -H "X-API-Key: tk_..." \
  -H "X-API-Secret: ..." \
  -H "Content-Type: application/json" \
  -d '{
    "email": "user@example.com",
    "name": "John Doe",
    "external_user_id": "12345",
    "title": "Login Issue",
    "description": "Cannot login to the system",
    "priority": "high"
  }'
```

Response (201):

```
{
  "success": true,
  "data": {
    "ticket": { "id": 42, "title": "Login Issue", "status": "open", ... },
    "ticket_url": "https://tickets.yourcompany.com/tickets/42",
    "user": {
      "id": 7,
      "name": "John Doe",
      "email": "user@example.com",
      "external_user_id": "12345"
    },
    "sso_redirect": {
      "callback_url": "https://tickets.yourcompany.com/auth/callback",
      "instructions": "Generate a JWT with the payload below...",
      "jwt_payload": {
        "email": "user@example.com",
        "name": "John Doe",
        "role": "user",
        "redirect_url": "/tickets/42",
        "iat": "(current unix timestamp)",
        "exp": "(current unix timestamp + 120)",
        "jti": "(unique id)"
      }
    }
  }
}
```


Step 2: SSO the User to the Ticket

Option A: Using JWT (Recommended)

Generate a JWT using the payload template from the API response, sign it with your `jwt_secret`, and redirect the user's browser to the callback URL.

PHP (Laravel) Example:

```php
// After creating the ticket via API...
$apiResponse = $response->json();
$ssoRedirect = $apiResponse['data']['sso_redirect'];

// Build the JWT payload
$header = json_encode(['alg' => 'HS256', 'typ' => 'JWT']);
$payload = json_encode([
    'email' => $ssoRedirect['jwt_payload']['email'],
    'name'  => $ssoRedirect['jwt_payload']['name'],
    'role'  => 'user',
    'redirect_url' => $ssoRedirect['jwt_payload']['redirect_url'], // e.g. "/tickets/42"
    'iat'   => time(),
    'exp'   => time() + 120,
    'jti'   => uniqid('sso_', true),
]);

// Base64url encode
$headerB64  = rtrim(strtr(base64_encode($header), '+/', '-_'), '=');
$payloadB64 = rtrim(strtr(base64_encode($payload), '+/', '-_'), '=');
$signature  = rtrim(strtr(base64_encode(
    hash_hmac('sha256', "$headerB64.$payloadB64", $jwtSecret, true)
), '+/', '-_'), '=');

$token = "$headerB64.$payloadB64.$signature";

// Redirect the user's browser
return redirect("{$ssoRedirect['callback_url']}?token={$token}");
```

Option B: Using `/api/auth/redirect`

If you prefer the encrypted token approach, POST to `/api/auth/redirect` with the ticket path as `redirect_url`:

```php
$ticketPath = $apiResponse['data']['sso_redirect']['jwt_payload']['redirect_url']; // "/tickets

$response = Http::withHeaders([
    'X-API-Key'    => env('TICKET_API_KEY'),
    'X-API-Secret' => env('TICKET_API_SECRET'),
])->post('https://tickets.flare99.com/api/auth/redirect', [
    'email'        => $user->email,
    'name'         => $user->name,
    'tenant'       => 'your-tenant-slug',
    'redirect_url' => $ticketPath,  // The ticket path
]);

// Follow the redirect
return redirect($response->header('Location'));
```

Supported `redirect_url` Formats

Format	Example	Works?
Relative path	`/tickets/42`	? Yes (recommended)
Full URL (same host)	`https://tickets.yourcompany.com/tickets/42`	? Yes
External URL (allowlisted host)	`https://yourcompany.com/some-page`	? Yes
External URL (unknown host)	`https://evil.com/phish`	? Blocked

Testing

1. **Test with logged-in user:**
    - Log in to your main website
    - Go to ticket system logout page
    - Click "Sign In with [Your Company]"
    - You should be redirected through your SSO page and land on the ticket dashboard

2. **Test with logged-out user:**
    - Log out from your main website
    - Go to ticket system logout page
    - Click "Sign In with [Your Company]"
    - You should be redirected to your website's login page

3. **Test SSO to specific ticket:**

    - Create a ticket via the API
    - Use the SSO redirect flow with `redirect_url` set to `/tickets/{id}`
    - Verify the user lands directly on the ticket page, not the dashboard

Troubleshooting

"Authentication Failed" Error

- Verify your SSO endpoint URL is correct (`https://yourwebsite.com/ticket-sso`)
- Check that the user is logged in on your main website
- Verify the API call to `/api/auth/redirect` (with `tenant` in post body) is successful
- Check server logs for CURL errors or API failures

Redirect Loop

- Ensure your SSO page doesn't redirect back to itself
- Check that the API returns a proper Location header (302 redirect)
- Verify the encrypted token is being passed correctly

User Lands on Dashboard Instead of Ticket

- Ensure `redirect_url` is included in the JWT payload (not just the API request body)
- Use a relative path like `/tickets/42` instead of a full URL
- If using a full URL, ensure the host matches the tenant's ticket domain exactly

CORS Errors

- Use server-side implementation (Option 1 or 2) to avoid CORS issues
- Client-side fetch() cannot access cookies from different domains

Security Notes

- Always validate that the user is authenticated before making the API call
- Use HTTPS for all SSO endpoints

- The ticket system API uses encrypted tokens with 2-minute expiration
- Tokens are single-use and cannot be replayed
- The `redirect_url` is validated: only internal paths and allowlisted external hosts are accepted

Support

For additional help, refer to:

- `TENANT_INTEGRATION_GUIDE.md` - Complete integration documentation
- `/integration` route - Integration hub with downloadable examples
- Example files in `tenant-examples/` directory