External Authentication Integration Guide

Generated on February 14, 2026

Tenant Integration Guide for Central Tickets System

Overview

This guide explains how external applications and tenants can integrate with the Central Tickets multi-tenant SaaS system. The system supports secure authentication redirects where external apps can authenticate users and seamlessly redirect them to tenant-specific ticket systems.

Integration Methods

1. Authentication Redirect Flow (Recommended)

Best for: Web applications that need to authenticate users and redirect them to the ticket system

How It Works

1. External app authenticates user
2. External app calls Central Tickets API with user data
3. User gets redirected to tenant subdomain with encrypted token
4. User is automatically logged in and redirected to final destination

API Endpoint

```
POST https://ticketsystem.flare99.com/api/auth/redirect/{tenant_slug}
Content-Type: application/json

{
  "email": "user@example.com",
  "name": "John Doe",
  "redirect_url": "https://your-app.com/dashboard"
}
```

Example Implementation

JavaScript/Node.js:

```
async function redirectToTickets(email, name, tenantSlug, redirectUrl) {
    const response = await fetch(`https://ticketsystem.flare99.com/api/auth/redirect/${tenantSl
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({
            email: email,
            name: name,
            redirect_url: redirectUrl
        })
    });

    if (response.redirected) {
        window.location.href = response.url;
    }
}
```

PHP:

```php
function redirectToTickets($email, $name, $tenantSlug, $redirectUrl) {
    $ch = curl_init();

    curl_setopt($ch, CURLOPT_URL, "https://ticketsystem.flare99.com/api/auth/redirect/{$tenantS
    curl_setopt($ch, CURLOPT_POST, true);
    curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode([
        'email' => $email,
        'name' => $name,
        'redirect_url' => $redirectUrl
    ]));
    curl_setopt($ch, CURLOPT_HTTPHEADER, [
        'Content-Type: application/json'
    ]);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_FOLLOWLOCATION, true);

    $response = curl_exec($ch);
    $redirectUrl = curl_getinfo($ch, CURLINFO_EFFECTIVE_URL);

    curl_close($ch);

    header("Location: {$redirectUrl}");
    exit;
}
```

Python:

```python
import requests

def redirect_to_tickets(email, name, tenant_slug, redirect_url):
    response = requests.post(
        f"https://ticketsystem.flare99.com/api/auth/redirect/{tenant_slug}",
        json={
            "email": email,
            "name": name,
            "redirect_url": redirect_url
        },
        allow_redirects=False
    )

    if response.status_code == 302:
        redirect_url = response.headers.get('Location')
        return redirect(redirect_url)
```

2. Direct API Integration

Best for: Applications that need full programmatic access to ticket management

Authentication

Use tenant-specific API tokens:

```
Authorization: Bearer {tenant_api_token}
X-Tenant-API-Token: {tenant_api_token}
```

Available Endpoints

```
# Tickets
POST   /api/tickets             # Create ticket
GET    /api/tickets             # List tickets
GET    /api/tickets/{id}        # Get ticket details
PUT    /api/tickets/{id}        # Update ticket
DELETE /api/tickets/{id}        # Delete ticket

# Categories
GET    /api/categories          # List categories

# Tenant Info
GET    /api/tenant              # Get tenant information
```

Example API Usage

```
// Create a ticket
const response = await fetch('https://ticketsystem.flare99.com/api/tickets', {
    method: 'POST',
    headers: {
        'Authorization': 'Bearer YOUR_TENANT_API_TOKEN',
        'Content-Type': 'application/json'
    },
    body: JSON.stringify({
        title: 'Support Request',
        description: 'I need help with...',
        priority: 'medium'
    })
});
```

4. JWT Login Integration (New - Recommended for Seamless Authentication)

Best for: Applications that want direct user login without redirects

How It Works

1. External app generates JWT token with user data
2. External app calls JWT login endpoint
3. User gets authenticated and session created
4. User can access ticket system directly

API Endpoint

```
POST https://ticketsystem.flare99.com/api/jwt/login
Content-Type: application/json

{
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...",
  "tenant_slug": "your-tenant-slug"
}
```

JWT Token Generation

Your application must generate a JWT token signed with your tenant's JWT secret:

```
// Node.js example
const jwt = require('jsonwebtoken');

function generateJWTToken(email, name, tenantSecret) {
    const payload = {
        email: email,
        name: name,
        iat: Math.floor(Date.now() / 1000),
        exp: Math.floor(Date.now() / 1000) + (2 * 60) // 2 minutes
    };

    return jwt.sign(payload, tenantSecret, { algorithm: 'HS256' });
}
```

Complete Integration Example

JavaScript (Browser):

```
async function loginToTickets(email, name, tenantSlug, tenantJwtSecret) {
    // Generate JWT token
    const token = generateJWTToken(email, name, tenantJwtSecret);

    // Call login endpoint
    const response = await fetch('https://ticketsystem.flare99.com/api/jwt/login', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        credentials: 'include', // Include cookies for session
        body: JSON.stringify({
            token: token,
            tenant_slug: tenantSlug
        })
    });

    const result = await response.json();

    if (result.success) {
        // User is now logged in
        window.location.href = result.redirect_url;
    } else {
        console.error('Login failed:', result.error);
    }
}
```

PHP (Server-side):

```php
<?php
use Firebase\JWT\JWT;

function generateJWTToken($email, $name, $tenantSecret) {
    $payload = [
        'email' => $email,
        'name' => $name,
        'iat' => time(),
        'exp' => time() + (2 * 60) // 2 minutes
    ];

    return JWT::encode($payload, $tenantSecret, 'HS256');
}

function loginToTickets($email, $name, $tenantSlug, $tenantSecret) {
    $token = generateJWTToken($email, $name, $tenantSecret);

    $ch = curl_init('https://ticketsystem.flare99.com/api/jwt/login');
    curl_setopt($ch, CURLOPT_POST, true);
    curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode([
        'token' => $token,
        'tenant_slug' => $tenantSlug
    ]));
    curl_setopt($ch, CURLOPT_HTTPHEADER, [
        'Content-Type: application/json'
    ]);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

    $response = curl_exec($ch);
    $result = json_decode($response, true);

    if ($result['success']) {
        // Redirect user to ticket system
        header('Location: ' . $result['redirect_url']);
        exit;
    }
}
?>
```

Security Notes

- JWT tokens expire in 2 minutes for security
- Tokens must be signed with your tenant's JWT secret
- Use HTTPS for all API calls
- Store tenant JWT secret securely (server-side only)

Step-by-Step Integration Process

Step 1: Tenant Setup

1. **Create Tenant Account**: Sign up at `https://ticketsystem.flare99.com`
2. **Configure Domain**: Set up your tenant subdomain (e.g., `tickets.yourcompany.com`)
3. **Get API Token**: Retrieve your tenant API token from settings
4. **Configure DNS**: Point your subdomain to the ticket system server

Step 2: Domain Configuration

```
# DNS Configuration
tickets.yourcompany.com CNAME ticketsystem.flare99.com
# or
tickets.yourcompany.com A YOUR_SERVER_IP
```

Step 3: SSL Certificate Setup

Ensure HTTPS is configured for your tenant domain:

```
# Let's Encrypt example
certbot certonly --webroot -w /var/www/html -d tickets.yourcompany.com
```

Step 4: Integration Implementation

Choose your integration method and implement the code in your application.

Step 5: Testing

Use the provided test endpoints:

```
# Test redirect (returns JSON instead of redirect)
POST https://ticketsystem.flare99.com/api/test/auth/redirect/{tenant}

# Test pages
GET https://ticketsystem.flare99.com/test-auth-redirect
GET https://ticketsystem.flare99.com/test-auth-redirect-form
```

Step 6: Go Live

1. Update your code to use production endpoints
2. Test the complete flow
3. Monitor integration logs
4. Handle error cases gracefully

Security Considerations

Token Security

- Tokens expire in 2 minutes
- Replay prevention using JTI (JWT ID)
- Encrypted using AES-256-CBC
- Domain binding validation

Rate Limiting

- 60 requests per hour per IP for auth redirects
- Configurable rate limits for API endpoints

Data Isolation

- Complete tenant data isolation
- User sessions scoped to tenant domains
- API tokens tenant-specific

Error Handling

Common Error Codes

- `400 Bad Request`: Invalid request data
- `401 Unauthorized`: Invalid API token
- `403 Forbidden`: Domain not verified for tenant
- `404 Not Found`: Tenant or endpoint not found

- `429 Too Many Requests`: Rate limit exceeded
- `500 Internal Server Error`: Server error

Error Response Format

```
{
  "error": "Error message",
  "code": "ERROR_CODE",
  "details": "Additional information"
}
```

Monitoring & Support

Logs

- Security events logged with correlation IDs
- API usage metrics available
- Error tracking and alerting

Support

- Integration documentation: `/integration` endpoint
- Test tools available at `/test-auth-redirect`
- API documentation in repository

JavaScript Integration Examples

Laravel Blade Template Approach (Recommended)

Best for: Laravel applications where you can modify Blade templates

```
<!DOCTYPE html>
<html>
<head>
    <title>My Website</title>
    <meta name="csrf-token" content="{{ csrf_token() }}">
</head>
<body>
    <button onclick="contactSupport()">Contact Support</button>

    <script>
        // Pass user data from Laravel to JavaScript
        const currentUser = @json([
            'email' => auth()->user()->email ?? null,
            'name' => auth()->user()->name ?? null,
            'tenant_slug' => auth()->user()->tenant->slug ?? 'your-tenant-slug'
        ]);

        function getCurrentUserEmail() {
            return currentUser.email;
        }

        function getCurrentUserName() {
            return currentUser.name;
        }

        function contactSupport() {
            const email = getCurrentUserEmail();
            const name = getCurrentUserName();

            if (!email || !name) {
                alert('Please log in to contact support');
                return;
            }

            // Create form and submit
            const form = document.createElement('form');
            form.method = 'POST';
            form.action = `https://ticketsystem.flare99.com/api/auth/redirect/${currentUser.ten

            const emailField = document.createElement('input');
            emailField.type = 'hidden';
            emailField.name = 'email';
            emailField.value = email;
            form.appendChild(emailField);

            const nameField = document.createElement('input');
            nameField.type = 'hidden';
            nameField.value = name;
            form.appendChild(nameField);

            const redirectField = document.createElement('input');
            redirectField.type = 'hidden';
            redirectField.name = 'redirect_url';
            redirectField.value = window.location.href; // Return to current page
            form.appendChild(redirectField);

            document.body.appendChild(form);
            form.submit();
        }
```

```
    </script>
</body>
</html>
```

Meta Tags Approach

Best for: When you can't modify the main template structure

```
<!DOCTYPE html>
<html>
<head>
    <title>My Website</title>
    <meta name="user-email" content="{{ auth()->user()->email ?? '' }}">
    <meta name="user-name" content="{{ auth()->user()->name ?? '' }}">
    <meta name="tenant-slug" content="{{ auth()->user()->tenant->slug ?? 'your-tenant-slug' }}"
</head>
<body>
    <button onclick="contactSupport()">Contact Support</button>

    <script>
        function getCurrentUserEmail() {
            return document.querySelector('meta[name="user-email"]').getAttribute('content');
        }

        function getCurrentUserName() {
            return document.querySelector('meta[name="user-name"]').getAttribute('content');
        }

        function contactSupport() {
            const email = getCurrentUserEmail();
            const name = getCurrentUserName();
            const tenantSlug = document.querySelector('meta[name="tenant-slug"]').getAttribute(

            if (!email || !name) {
                alert('Please log in to contact support');
                return;
            }

            // Create form and submit
            const form = document.createElement('form');
            form.method = 'POST';
            form.action = `https://ticketsystem.flare99.com/api/auth/redirect/${tenantSlug}`;

            const emailField = document.createElement('input');
            emailField.type = 'hidden';
            emailField.name = 'email';
            emailField.value = email;
            form.appendChild(emailField);

            const nameField = document.createElement('input');
            nameField.type = 'hidden';
            nameField.name = 'name';
            nameField.value = name;
            form.appendChild(nameField);

            const redirectField = document.createElement('input');
            redirectField.type = 'hidden';
            redirectField.name = 'redirect_url';
            redirectField.value = window.location.href;
            form.appendChild(redirectField);

            document.body.appendChild(form);
            form.submit();
        }
    </script>
</body>
</html>
```

AJAX Approach

Best for: Single Page Applications or when you need dynamic user data

```
<!DOCTYPE html>
<html>
<head>
    <title>My Website</title>
</head>
<body>
    <button onclick="contactSupport()">Contact Support</button>

    <script>
        async function getCurrentUserData() {
            try {
                const response = await fetch('/api/user/current', {
                    method: 'GET',
                    headers: {
                        'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-token"]').getAt
                        'Content-Type': 'application/json'
                    }
                });

                if (response.ok) {
                    return await response.json();
                } else {
                    throw new Error('Failed to get user data');
                }
            } catch (error) {
                console.error('Error fetching user data:', error);
                return null;
            }
        }

        function getCurrentUserEmail() {
            // This would need to be called asynchronously
            // For synchronous access, use one of the other approaches
            return null;
        }

        function getCurrentUserName() {
            // This would need to be called asynchronously
            return null;
        }

        async function contactSupport() {
            const userData = await getCurrentUserData();

            if (!userData || !userData.email || !userData.name) {
                alert('Please log in to contact support');
                return;
            }

            // Create form and submit
            const form = document.createElement('form');
            form.method = 'POST';
            form.action = `https://ticketsystem.flare99.com/api/auth/redirect/${userData.tenant

            const emailField = document.createElement('input');
            emailField.type = 'hidden';
            emailField.name = 'email';
            emailField.value = userData.email;
            form.appendChild(emailField);
```

```
            const nameField = document.createElement('input');
            nameField.type = 'hidden';
            nameField.name = 'name';
            nameField.value = userData.name;
            form.appendChild(nameField);

            const redirectField = document.createElement('input');
            redirectField.type = 'hidden';
            redirectField.name = 'redirect_url';
            redirectField.value = window.location.href;
            form.appendChild(redirectField);

            document.body.appendChild(form);
            form.submit();
        }
    </script>
</body>
</html>
```

Backend API Endpoint for AJAX Approach

Add this route to your `routes/api.php`:

```
Route::middleware('auth:sanctum')->get('/user/current', function (Request $request) {
    return response()->json([
        'email' => $request->user()->email,
        'name' => $request->user()->name,
        'tenant_slug' => $request->user()->tenant->slug ?? null
    ]);
});
```

E-commerce Platform

```
// After user login on e-commerce site
async function handleSupportRequest(userEmail, userName) {
    await redirectToTickets(
        userEmail,
        userName,
        'your-store',
        'https://yourstore.com/account/support'
    );
}
```

SaaS Application

```
// In user dashboard
public function contactSupport(Request $request) {
    return redirectToTickets(
        Auth::user()->email,
        Auth::user()->name,
        'your-saas',
        route('dashboard')
    );
}
```

WordPress Plugin

```
// WordPress integration
add_action('wp_ajax_create_ticket', 'handle_ticket_creation');
function handle_ticket_creation() {
    $user = wp_get_current_user();

    redirectToTickets(
        $user->user_email,
        $user->display_name,
        'your-wordpress-site',
        home_url('/support/thanks')
    );
}
```

Troubleshooting

Common Issues

1. **CORS Errors**: Enable CORS middleware on server
2. **Domain Not Found**: Verify DNS configuration
3. **SSL Errors**: Ensure HTTPS certificate is valid
4. **Token Expired**: Tokens valid for 2 minutes only
5. **Rate Limited**: Implement exponential backoff

Debug Tools

- Browser developer tools for redirect inspection
- Laravel logs for server-side debugging
- Test endpoints for isolated testing

Best Practices

1. **Always use HTTPS** for all requests
2. **Validate user input** before API calls
3. **Handle errors gracefully** with user-friendly messages
4. **Implement logging** for integration monitoring
5. **Use environment variables** for configuration
6. **Test thoroughly** before going live
7. **Monitor API usage** and set up alerts

Support Resources

- **Integration Hub**: `https://ticketsystem.flare99.com/integration`
- **API Documentation**: Repository README and inline comments
- **Test Tools**: Built-in test pages and endpoints
- **Example Code**: Multiple language examples provided

This integration guide is maintained with the codebase. Check for updates regularly.