BBTTCC Raid v4.8.1-ENHANCED - Updated Testing Guide

Overview

This testing guide covers the enhanced BBTTCC Raid module (v4.8.1-ENHANCED) designed for FoundryVTT v13+ and D&D5e v5.1.4+ compatibility. The module provides comprehensive raid planning, resource management, and execution mechanics.

System Requirements

- **FoundryVTT**: v13.0+ (tested on v13.348)
- **D&D5e System**: v5.1.4+ (tested on v5.1.4)
- **Module Dependencies**: BBTTCC Factions (recommended for full functionality)

Module Installation

1. Copy folder to FoundryVTT `Data/modules/` directory
2. Rename folder to exactly `bbttcc-raid` (no version suffix)
3. Enable module in FoundryVTT Module Management
4. Refresh FoundryVTT

Test Case 1: Installation & Module Initialization

Core Functionality Test

Expected Behavior:

- ✅ Module loads without critical errors
- ✅ Modern Application patterns for FoundryVTT v13+
- ✅ World flags-based data storage system
- ✅ Async/await patterns throughout
- ✅ Settings registration and API exposure

Installation Verification

```javascript
```

```
// Console test - check module status
const raidMod = game.modules.get('bbttcc-raid');
console.log('Module found:', !!raidMod);
console.log('Module active:', raidMod?.active);
console.log('API available:', !!raidMod?.api);

// Check system compatibility
console.log('Game version:', game.version);
console.log('System:', game.system.id, game.system.version);
```

Expected Results

- Console shows: "BBTTCC Raid v4.8.1-ENHANCED | System fully operational"
- No JavaScript errors in console
- Settings menu appears under "Configure Settings" → "Module Settings"
- API methods available via `game.modules.get('bbttcc-raid').api`

Test Case 2: Settings Configuration

Settings Registration Test

Objective: Verify all module settings are properly registered

Settings to Verify

1. **Enable Macro Integration** (Boolean, default: true)
2. **Debug Mode** (Boolean, default: false)
3. **Auto-Calculate Difficulty** (Boolean, default: true)

Test Steps

1. Navigate to Configure Settings → Module Settings
2. Locate "BBTTCC Raid v4.8.1-ENHANCED" section
3. Verify all 3 settings are present and functional
4. Test changing values and saving
5. Reload world and verify settings persist

Test Case 3: World Flag-Based Data Storage

Raid Data Structure Test

Objective: Verify raid data integrates with world flags storage

Test Steps

```javascript
// Check world flags structure
const worldRaids = game.world.getFlag('bbttcc-raid', 'raids') || {};
console.log('World raids data:', worldRaids);

// Check API availability
const api = game.modules.get('bbttcc-raid').api;
console.log('Available API methods:', Object.keys(api));
```

Expected Data Structure

```javascript
{
  "raid-id": {
    status: 'planning',
    name: 'Test Raid',
    type: 'assault',
    target: 'Enemy Base',
    objectives: ['Destroy defenses'],
    participants: ['faction-id'],
    resources: {
      violence: 10,
      nonLethal: 5,
      intrigue: 3,
      economy: 7
    },
    timeline: {
      preparation: 24,
      execution: 4,
      extraction: 2
    },
    difficulty: 'medium'
  }
}
```

Test Case 4: Raid Creation & Management

API-Based Raid Creation

Objective: Test core raid creation mechanics

Basic Raid Creation

```javascript
const api = game.modules.get('bbttcc-raid').api;

// Test raid creation
const testRaid = await api.createRaid({
    name: "Test Strike Mission",
    type: "assault",
    target: "Enemy Fortress",
    objectives: ["Neutralize defenses", "Capture commander"],
    participants: ["faction-id-here"], // Use actual faction ID
    resources: {
        violence: 15,
        nonLethal: 5,
        intrigue: 8,
        economy: 12
    }
});

console.log('Created raid:', testRaid);
```

Raid Types Testing

Test all available raid types:

- `assault` - Direct Military Attack
- `infiltration` - Stealth Operation
- `sabotage` - Disruption Mission
- `heist` - Resource Acquisition
- `rescue` - Extraction Mission
- `reconnaissance` - Information Gathering

Expected Results

- Raid creation returns complete raid object with generated ID

- Data persists in world flags
- Chat notification confirms creation
- No console errors during creation

Test Case 5: Difficulty Calculation System

Auto-Difficulty Testing

Objective: Test automatic difficulty calculation

Difficulty Factors Test

```javascript
const api = game.modules.get('bbttcc-raid').api;

// Test difficulty calculation
const testRaidData = {
  type: 'assault', // +3 modifier
  objectives: ['Obj1', 'Obj2', 'Obj3'], // +1.5 for 3 objectives
  participants: ['faction1'], // +1 for single participant
  resources: {
    violence: 5, // Low resources = +2
    nonLethal: 0,
    intrigue: 0,
    economy: 0
  }
};

const calculatedDifficulty = api.calculateDifficulty(testRaidData);
console.log('Calculated difficulty:', calculatedDifficulty);
```

Difficulty Levels to Test

- `trivial` (modifier: -2)
- `easy` (modifier: -1)
- `medium` (modifier: 0)
- `hard` (modifier: 1)
- `extreme` (modifier: 2)

Expected Behavior

- Difficulty increases with more objectives
- Difficulty increases with fewer resources
- Difficulty increases with fewer participants
- Different raid types have different base modifiers

Test Case 6: Raid Planner Interface

Modern UI Application Test

Objective: Test the RaidPlanner Application class

Open Raid Planner

```javascript
const api = game.modules.get('bbttcc-raid').api;

// Open new raid planner
const planner = new api.RaidPlanner();
planner.render(true);

// Or edit existing raid
const existingRaid = api.getRaidData('raid-id');
if (existingRaid) {
    const editPlanner = new api.RaidPlanner(existingRaid);
    editPlanner.render(true);
}
```

UI Features to Test

- **Tabbed Interface**: Basics, Resources, Participants, Timeline, Execution
- **Form Validation**: Required fields with proper error messages
- **Resource Controls**: +/- buttons for resource allocation
- **Objective Management**: Add/remove objectives dynamically
- **Participant Selection**: Multi-select from available factions
- **Timeline Planning**: Customizable phase durations
- **Auto-Calculate**: Difficulty calculation button

Expected Results

- All tabs render without errors
- Form validation prevents invalid submissions
- Resource controls update values correctly
- Participant selection shows available factions
- Timeline totals calculate correctly

Test Case 7: Raid Execution System

Execution & Outcome Test

Objective: Test raid execution and outcome generation

Execute Raid

```javascript
const api = game.modules.get('bbttcc-raid').api;

// First create a valid raid
const raid = await api.createRaid({
    name: "Test Execution",
    type: "heist",
    objectives: ["Steal supplies"],
    participants: ["faction-id"],
    resources: { violence: 10, intrigue: 15, economy: 5, nonLethal: 0 }
});

// Execute the raid
const result = await api.executeRaid(raid.id);
console.log('Execution result:', result);
console.log('Outcome:', result.outcome);
```

Outcome Elements to Verify

- **Success/Failure**: Based on resource adequacy and difficulty
- **Severity Levels**: Critical, Major, Minor (for both success/failure)
- **Roll Results**: Random roll vs calculated success chance
- **Rewards**: Appropriate rewards for successful raids

- **Consequences**: Penalties for failed raids
- **Description**: Narrative description of outcome

Success Chance Factors

- Base 60% success rate
- +1% per 5 resource points
- Difficulty modifier (±15% per modifier point)
- +5% per participating faction

Test Case 8: Cross-Module Integration

Faction Integration Test

Objective: Test integration with BBTTCC Factions module

Prerequisites

BBTTCC Factions module must be active with created factions

Integration Testing

```javascript
```

```
// Check for available factions
const factions = game.actors.filter(actor =>
    actor.flags['bbttcc-factions']?.isFaction === true
);
console.log('Available factions:', factions.length);

// Test raid creation with faction participants
const api = game.modules.get('bbttcc-raid').api;
const raid = await api.createRaid({
    name: "Integration Test",
    type: "reconnaissance",
    participants: [factions[0].id],
    objectives: ["Test integration"],
    resources: { intrigue: 10 }
});

// Execute and check reward application
const result = await api.executeRaid(raid.id);
if (result.outcome.success) {
    // Check if rewards were applied to faction
    const faction = factions[0];
    const updatedOPs = faction.getFlag('bbttcc-factions', 'ops');
    console.log('Updated faction OPs:', updatedOPs);
}
```

Expected Integration Features

- Raid planner shows available factions
- Successful raids apply OP rewards to participating factions
- Faction sheets show associated raids
- Cross-module data consistency

Test Case 9: Macro Integration

Modern Macro Test

Objective: Test the included CreateBBTTCCRaid-MODERN.js macro

Macro Execution

1. Import the macro from the module's macros folder

2. Execute the macro as GM
3. Verify enhanced dialog appears with:
    - Comprehensive form fields for all raid data
    - Faction selection dropdown
    - Resource allocation controls
    - Validation and error handling

Macro Features to Test

- **Enhanced Dialog**: Professional styling with validation
- **Comprehensive Form**: All raid creation fields
- **Error Handling**: Timeout protection and user feedback
- **Multiple Buttons**: Create Raid, Open Planner, Cancel
- **Success Dialog**: Detailed confirmation with next steps

Test Case 10: Data Persistence & Recovery

Persistence Testing

Objective: Test data reliability and recovery mechanisms

Data Persistence Test

javascript

```javascript
const api = game.modules.get('bbttcc-raid').api;

// Create multiple raids
const raids = [];
for (let i = 1; i <= 5; i++) {
  const raid = await api.createRaid({
    name: `Test Raid ${i}`,
    type: 'heist',
    objectives: [`Objective ${i}`],
    participants: [],
    resources: { economy: i * 5 }
  });
  raids.push(raid);
}

// Reload world and verify persistence
// (Manual step: F5 refresh)

// Check raids still exist
const worldRaids = game.world.getFlag('bbttcc-raid', 'raids');
console.log('Persisted raids:', Object.keys(worldRaids).length);
```

Recovery Testing

Test error handling for various edge cases:

- Invalid raid IDs
- Missing participant factions
- Corrupted raid data
- Network timeouts during operations

Test Case 11: Performance & Scalability

Performance Testing

javascript

```javascript
// Test bulk operations
const api = game.modules.get('bbttcc-raid').api;
const startTime = performance.now();

// Create multiple raids rapidly
const promises = [];
for (let i = 0; i < 10; i++) {
  promises.push(api.createRaid({
    name: `Performance Test ${i}`,
    type: 'reconnaissance',
    objectives: [`Test ${i}`],
    participants: [],
    resources: { intrigue: 5 }
  }));
}

await Promise.all(promises);
const endTime = performance.now();
console.log(`Created 10 raids in ${endTime - startTime}ms`);
```

Expected Performance Benchmarks

- Single raid creation: < 3 seconds
- Raid execution: < 2 seconds
- Bulk operations: < 10 seconds for 10 raids
- Memory usage: Stable with no significant leaks

Test Case 12: Error Handling & Edge Cases

Error Conditions to Test

Invalid Input Handling

javascript

```
const api = game.modules.get('bbttcc-raid').api;

// Test invalid raid creation
try {
    await api.createRaid(null);
} catch (error) {
    console.log('Properly caught error:', error.message);
}

try {
    await api.createRaid({ name: '' }); // Empty name
} catch (error) {
    console.log('Validation error caught:', error.message);
}

// Test invalid raid execution
try {
    await api.executeRaid('invalid-id');
} catch (error) {
    console.log('Invalid ID error caught:', error.message);
}
```

Boundary Testing

- Create raids with maximum resource allocations
- Test with empty objectives/participants arrays
- Execute raids with extreme difficulty modifiers
- Test update operations on non-existent raids

Test Case 13: Cleanup & Resource Management

Cleanup Testing

javascript

```javascript
const api = game.modules.get('bbttcc-raid').api;

// Create test raids
const testRaid = await api.createRaid({
    name: "Cleanup Test",
    type: "assault",
    objectives: ["Test cleanup"],
    participants: [],
    resources: { violence: 5 }
});

// Test deletion
const deleted = await api.deleteRaid(testRaid.id);
console.log('Deletion successful:', deleted);

// Verify removal from world flags
const worldRaids = game.world.getFlag('bbttcc-raid', 'raids');
console.log('Raid removed from world flags:', !worldRaids[testRaid.id]);
```

Integration Testing with Module Suite

Full Suite Compatibility

If other BBTTCC modules are present:

```javascript
```

```
// Check for other BBTTCC modules
const bbttccModules = game.modules.filter(m =>
    m.id.includes('bbttcc') && m.active
);
console.log('Active BBTTCC modules:', bbttccModules.map(m => m.id));

// Test cross-module API availability
const factionsAPI = game.modules.get('bbttcc-factions')?.api;
const territoryAPI = game.modules.get('bbttcc-territory')?.api;
const radiationAPI = game.modules.get('bbttcc-radiation')?.api;

console.log('Cross-module APIs:', {
    factions: !!factionsAPI,
    territory: !!territoryAPI,
    radiation: !!radiationAPI
});
```

Expected Success Criteria

Module Must:

- ✅ Load without console errors in FoundryVTT v13.348
- ✅ Integrate seamlessly with D&D5e v5.1.4+
- ✅ Store data reliably using world flags
- ✅ Provide functional RaidPlanner UI application
- ✅ Handle errors gracefully with user-friendly messages
- ✅ Support macro-based raid creation
- ✅ Calculate raid outcomes accurately
- ✅ Integrate with BBTTCC Factions for rewards

Performance Benchmarks

- Raid creation: < 3 seconds
- Raid execution: < 2 seconds for outcome calculation
- UI rendering: < 2 seconds for complex forms
- Memory usage: Stable with no memory leaks

Common Issues & Solutions

1. "Module not found" Errors

Cause: Incorrect folder naming **Solution**: Ensure folder is named exactly `bbttcc-raid`

2. "API not available" Errors

Cause: Module not fully initialized **Solution**: Use `await api.waitForReady()` before API calls

3. Data Not Persisting

Cause: World flag update failures **Solution**: Check GM permissions and world flag structure

4. Faction Integration Issues

Cause: BBTTCC Factions module not active **Solution**: Enable BBTTCC Factions module for full functionality

5. UI Not Rendering

Cause: Template loading issues **Solution**: Verify template files exist and are accessible

---

Diagnostic Commands

Quick Status Check

```javascript
// Comprehensive module status
const mod = game.modules.get('bbttcc-raid');
console.log({
  found: !!mod,
  active: mod?.active,
  hasAPI: !!mod?.api,
  isReady: mod?.api?.isReady?.() || false,
  raidTypes: mod?.api?.RAID_TYPES,
  difficulties: mod?.api?.RAID_DIFFICULTIES
});
```

World Data Status

```javascript
```

```javascript
const api = game.modules.get('bbttcc-raid').api;
const worldRaids = game.world.getFlag('bbttcc-raid', 'raids') || {};

console.log('World raids summary:', {
    totalRaids: Object.keys(worldRaids).length,
    raidStatuses: Object.values(worldRaids).map(r => r.status),
    raidTypes: Object.values(worldRaids).map(r => r.type)
});
```

Available Factions Check

```javascript
const factions = game.actors.filter(actor =>
    actor.flags['bbttcc-factions']?.isFaction === true
);

console.log('Available factions for raids:', {
    count: factions.length,
    names: factions.map(f => f.name),
    totalOPs: factions.map(f => {
        const ops = f.flags['bbttcc-factions']?.ops || {};
        return Object.values(ops).reduce((sum, val) => sum + val, 0);
    })
});
```

Updated for BBTTCC Raid v4.8.1-ENHANCED

Target Environment: FoundryVTT v13.348, D&D5e v5.1.4+