BBTTCC Factions v4.8.1-ENHANCED - User Acceptance Testing Guide

Overview

This UAT guide covers the enhanced BBTTCC Factions module (v4.8.1-ENHANCED) designed for FoundryVTT v13+ and D&D5e v5.1.4+ compatibility. The module has been completely modernized with current patterns, comprehensive diagnostics, and improved error handling.

System Requirements

- **FoundryVTT**: v13.0+ (verified on v13.348)
- **D&D5e System**: v5.1.4+ (compatible up to v5.4+)
- **Module Dependencies**: Standalone module with optional integrations
- **Browser**: Modern browser with ES6+ support

Pre-Test Setup Requirements

Installation Verification

1. **Correct Folder Structure**:

```
Data/modules/bbttcc-factions/
├── module.json
├── scripts/
│   ├── bbttcc-factions.js
│   ├── faction-sheet.js
│   └── faction-actor.js
├── templates/
│   └── faction-sheet.html
├── styles/
│   └── faction-sheet.css
└── lang/
    └── en.json
```

2. **Folder Name Validation**: Ensure folder is named exactly `bbttcc-factions` (no version suffixes)
3. **Module Activation**: Enable in FoundryVTT Module Management and refresh

---

Core Functionality Tests

Test 1: Module Initialization

Objective: Verify the module loads without errors and initializes properly

Console Commands:

```javascript
// Check module status
const mod = game.modules.get('bbttcc-factions');
console.log('Module found:', !!mod);
console.log('Module active:', mod?.active);
console.log('API available:', !!mod?.api);

// Check modern patterns
console.log('FactionSheet class available:', typeof FactionSheet !== 'undefined');
console.log('Modern API ready:', !!window.BBTTCC?.Factions);
```

Expected Results:

- ✅ Console shows "BBTTCC Factions v4.8.1-ENHANCED | Starting initialization..."
- ✅ Console shows "Module initialized" and "Module ready"
- ✅ No JavaScript errors in console
- ✅ All console checks return `true`

Test 2: Sheet Registration Verification

Objective: Confirm custom faction sheet is properly registered

Console Commands:

```javascript
```

```
// Check sheet registration
const sheets = CONFIG?.Actor?.sheetClasses?.npc;
console.log('Available NPC sheets:', Object.keys(sheets || {}));
console.log('BBTTCC sheet registered:', !!(sheets && sheets['bbttcc-factions.FactionSheet']));

// Check registration details
if (sheets && sheets['bbttcc-factions.FactionSheet']) {
  const sheetClass = sheets['bbttcc-factions.FactionSheet'];
  console.log('Sheet class:', sheetClass);
  console.log('Sheet label:', sheetClass.label);
}
```

Expected Results:

- ✅ "bbttcc-factions.FactionSheet" appears in available sheets
- ✅ Sheet class is properly defined
- ✅ Label shows "BBTTCC Faction Sheet"

Test 3: Modern API Functionality

Objective: Test the enhanced API system

Console Commands:

```javascript
// Test modern API
const api = game.modules.get('bbttcc-factions')?.api;
console.log('Modern API methods:', Object.keys(api || {}));
console.log('Factions API:', Object.keys(api?.factions || {}));
console.log('Events API:', Object.keys(api?.events || {}));
console.log('Utils API:', Object.keys(api?.utils || {}));

// Test API version info
console.log('API Version:', api?.version);
console.log('API Version:', api?.apiVersion);
console.log('Module ID:', api?.moduleId);
```

Expected Results:

- ✅ API contains: factions, events, utils, config sections
- ✅ Version shows "4.8.1-ENHANCED"

- ✅ API version shows "1.0"

Faction Creation Tests

Test 4: Modern Faction Creation via API

Objective: Test the enhanced faction creation system

Test Steps:

```javascript
// Test modern faction creation
async function testFactionCreation() {
  const startTime = performance.now();

  try {
    const api = game.modules.get('bbttcc-factions').api;
    const faction = await api.factions.create({
      name: "Test Faction Alpha",
      biography: "A test faction for UAT validation"
    });

    const endTime = performance.now();
    console.log(`Faction created in ${(endTime - startTime).toFixed(2)}ms`);
    console.log('Faction ID:', faction.id);
    console.log('Has OPs:', !!faction.getFlag('bbttcc-factions', 'ops'));
    console.log('Is Faction:', faction.getFlag('bbttcc-factions', 'isFaction'));

    return faction;
  } catch (error) {
    console.error('Faction creation failed:', error);
    throw error;
  }
}

// Run the test
const testFaction = await testFactionCreation();
```

Expected Results:

- ✅ Faction creates successfully in under 5 seconds
- ✅ Returns valid Actor object

- ✅ Has proper faction flags set
- ✅ Organization Points structure exists
- ✅ Success notification appears
- ✅ Sheet opens automatically

Test 5: Faction Data Structure Validation

Objective: Verify faction data integrity

Test Steps:

```javascript
// Test faction data validation
function validateFactionData(faction) {
    const moduleId = 'bbttcc-factions';
    const results = {};

    // Check basic flags
    results.isFaction = faction.getFlag(moduleId, 'isFaction');
    results.version = faction.getFlag(moduleId, 'version');
    results.hasOps = !!faction.getFlag(moduleId, 'ops');

    // Check OPs structure
    const ops = faction.getFlag(moduleId, 'ops');
    const expectedOPs = ['violence', 'nonlethal', 'intrigue', 'economy', 'softpower', 'diplomacy'];
    results.opsComplete = expectedOPs.every(op => ops && ops[op] && typeof ops[op].value === 'number');

    // Check arrays
    results.hasWarLog = Array.isArray(faction.getFlag(moduleId, 'warLog'));
    results.hasTerritories = Array.isArray(faction.getFlag(moduleId, 'territories'));
    results.hasBases = Array.isArray(faction.getFlag(moduleId, 'bases'));

    // Check sheet assignment
    results.sheetClass = faction.getFlag('core', 'sheetClass');

    console.log('Faction validation results:', results);
    return Object.values(results).every(r => r === true || r === 'bbttcc-factions.FactionSheet');
}

// Test with the created faction
const isValid = validateFactionData(testFaction);
console.log('Faction is valid:', isValid);
```

Expected Results:

- ✅ All validation checks return `true`
- ✅ Version shows "4.8.1-ENHANCED"
- ✅ Sheet class is "bbttcc-factions.FactionSheet"
- ✅ All 6 Organization Points exist with proper structure

Sheet Functionality Tests

Test 6: Faction Sheet Rendering

Objective: Verify the custom sheet renders properly

Test Steps:

1. Open the test faction created earlier
2. Verify sheet renders without blank sections
3. Check all tabs are present and functional
4. Verify data displays correctly

Manual Verification:

- ✅ Sheet opens with BBTTCC styling
- ✅ Header shows faction name and status
- ✅ Four tabs visible: "Organization Points", "Territories", "Warfare", "Details"
- ✅ Organization Points tab shows all 6 OPs with values
- ✅ +/- buttons are present and functional
- ✅ Roll buttons (d20 icons) are present
- ✅ Total OPs and Power Level display correctly

Test 7: Organization Points Management

Objective: Test OP adjustment and rolling system

Test Steps:

```javascript
```

```
// Test OP updates via API
async function testOPManagement(faction) {
  const api = game.modules.get('bbttcc-factions').api;

  try {
    // Test updating Violence OP
    const result = await api.factions.update(faction, 'violence', 5);
    console.log('Violence OP updated:', result);

    // Verify the change
    const ops = faction.getFlag('bbttcc-factions', 'ops');
    console.log('Current Violence OP:', ops.violence.value);

    // Test bounds checking (should clamp to max)
    await api.factions.update(faction, 'economy', 15);
    const economyOP = faction.getFlag('bbttcc-factions', 'ops').economy;
    console.log('Economy OP (should be clamped to 10):', economyOP.value);

    return true;
  } catch (error) {
    console.error('OP management test failed:', error);
    return false;
  }
}

// Run OP test
const opTestResult = await testOPManagement(testFaction);
console.log('OP management test passed:', opTestResult);
```

Manual UI Tests:

1. Click + button next to Violence OP several times
2. Click - button to decrease value
3. Try to exceed maximum value (should clamp to 10)
4. Click roll button next to Economy OP
5. Verify chat message appears with roll result

Expected Results:

- ✅ +/- buttons update values immediately
- ✅ Values constrained between 0 and max (10)

- ✅ Total OPs updates automatically
- ✅ Power Level recalculates (Emerging → Growing → etc.)
- ✅ Roll buttons generate proper chat messages
- ✅ Roll formula shows "1d20 + OP value"

Test 8: War Log and Base Management

Objective: Test dynamic content management

Manual Test Steps:

1. Switch to "Warfare" tab
2. Click "Add Entry" button under War Log
3. Fill out the dialog with test data:
   - Title: "Captured Northern Outpost"
   - Type: "Victory"
   - Description: "Successfully took control of strategic position"
4. Click "Add Entry"
5. Verify entry appears in war log
6. Click "Add Base" button
7. Add a test base:
   - Name: "Command Center Alpha"
   - Type: "Headquarters"
   - Description: "Primary operations base"
8. Verify base appears in list

Expected Results:

- ✅ War log entry dialog opens properly
- ✅ Entry appears immediately after adding
- ✅ Entry shows title, type badge, and description
- ✅ Delete button (trash icon) appears and functions
- ✅ Base entry dialog opens properly
- ✅ Base appears in bases section
- ✅ Base delete button functions correctly

Integration and Advanced Tests

Test 9: Data Persistence

Objective: Verify data survives session reload

Test Steps:

1. Note current faction data (OPs, war log entries, bases)
2. Close faction sheet
3. Refresh FoundryVTT (F5)
4. Reopen faction sheet
5. Verify all data is preserved

Expected Results:

- ✅ All OP values preserved exactly
- ✅ War log entries remain with correct data
- ✅ Bases list unchanged
- ✅ Sheet renders quickly after reload
- ✅ No data corruption or loss

Test 10: Performance Testing

Objective: Test module performance under load

Console Test:

```javascript
```

```
// Performance test - create multiple factions
async function performanceTest() {
  const startTime = performance.now();
  const api = game.modules.get('bbttcc-factions').api;
  const factions = [];

  try {
    // Create 5 factions concurrently
    const promises = Array.from({length: 5}, (_, i) =>
      api.factions.create({
        name: `Performance Test Faction ${i + 1}`,
        biography: `Test faction ${i + 1} for performance testing`
      })
    );

    const results = await Promise.all(promises);
    const endTime = performance.now();

    console.log(`Created ${results.length} factions in ${(endTime - startTime).toFixed(2)}ms`);
    console.log('Average time per faction:', ((endTime - startTime) / results.length).toFixed(2) + 'ms');

    // Cleanup
    for (const faction of results) {
      await faction.delete();
    }

    return true;
  } catch (error) {
    console.error('Performance test failed:', error);
    return false;
  }
}

// Run performance test
const perfResult = await performanceTest();
console.log('Performance test passed:', perfResult);
```

Expected Results:

- ✅ 5 factions create in under 15 seconds total
- ✅ Average creation time under 3 seconds per faction
- ✅ No memory leaks or performance degradation

- ✅ All factions have valid data structure

Diagnostic and Troubleshooting Tests

Test 11: Built-in Diagnostics

Objective: Test the module's diagnostic system

Console Command:

```javascript
// Run comprehensive diagnostics
const diagnostics = await game.modules.get('bbttcc-factions').api.runDiagnostics();
console.log('Diagnostic Results:', diagnostics);

// Check specific diagnostic areas
console.log('Tests run:', diagnostics.tests.length);
console.log('All tests passed:', diagnostics.tests.every(t => t.passed));
console.log('Failed tests:', diagnostics.tests.filter(t => !t.passed));
```

Expected Results:

- ✅ Diagnostics complete without errors
- ✅ All diagnostic tests pass
- ✅ Results include timestamp and version info
- ✅ Core functionality test passes
- ✅ API availability test passes
- ✅ Existing factions validation passes

Test 12: Error Recovery Testing

Objective: Test error handling and recovery mechanisms

Console Tests:

```javascript
```

```
// Test 1: Invalid faction creation
try {
    const api = game.modules.get('bbttcc-factions').api;
    await api.factions.create({ name: "" }); // Empty name should fail
} catch (error) {
    console.log('✅ Empty name properly rejected:', error.message);
}

// Test 2: Invalid OP update
try {
    const api = game.modules.get('bbttcc-factions').api;
    await api.factions.update(testFaction, 'invalidOP', 5);
} catch (error) {
    console.log('✅ Invalid OP type properly rejected:', error.message);
}

// Test 3: Validation and repair
const repairResult = await game.modules.get('bbttcc-factions').api.factions.repair(testFaction);
console.log('✅ Validation/repair completed:', repairResult);
```

Expected Results:

- ✅ Invalid operations throw appropriate errors
- ✅ Error messages are user-friendly
- ✅ Module continues functioning after errors
- ✅ Validation/repair system works correctly

Legacy Compatibility Tests

Test 13: Backward Compatibility

Objective: Ensure legacy API methods still work

Console Commands:

```javascript
```

```
// Test legacy API access
console.log('Legacy window.BBTTCCFactions:', !!window.BBTTCCFactions);
console.log('Legacy BBTTCC.Factions:', !!window.BBTTCC?.Factions);

// Test legacy creation method
if (window.BBTTCCFactions) {
  try {
    const legacyFaction = await window.BBTTCCFactions.createFaction({
      name: "Legacy Test Faction"
    });
    console.log('✅ Legacy creation method works:', !!legacyFaction);

    // Cleanup
    await legacyFaction.delete();
  } catch (error) {
    console.log('❌ Legacy creation failed:', error);
  }
}
```

Expected Results:

- ✅ Legacy global APIs are available
- ✅ Legacy creation method works
- ✅ Legacy methods produce same results as modern API

Success Criteria Summary

Module Must Pass All:

- ✅ **Initialization**: Loads without errors, proper console output
- ✅ **Sheet Registration**: Custom sheet available and functional
- ✅ **API Exposure**: Modern and legacy APIs working
- ✅ **Faction Creation**: Reliable creation under 5 seconds
- ✅ **Data Structure**: All required flags and arrays present
- ✅ **Sheet Rendering**: No blank sections, all tabs functional
- ✅ **OP Management**: +/- buttons, bounds checking, rolling
- ✅ **Content Management**: War log and bases CRUD operations
- ✅ **Data Persistence**: Survives session reload

- ✅ **Performance**: Multiple factions creation under 15 seconds
- ✅ **Diagnostics**: Built-in tests all pass
- ✅ **Error Handling**: Graceful error recovery
- ✅ **Compatibility**: D&D5e v5.1.4+ and v5.4+ support

Integration Requirements:

- ✅ **No Conflicts**: Works alongside standard D&D5e features
- ✅ **Sheet Selection**: Faction sheet appears in actor sheet options
- ✅ **Chat Integration**: Roll messages appear properly formatted
- ✅ **Flag System**: Reliable data storage using actor flags

Troubleshooting Quick Reference

Common Issues:

1. **Blank Sheet**: Check console for template loading errors
2. **Missing OPs**: Run validation/repair API command
3. **Sheet Not Listed**: Verify sheet registration in console
4. **Slow Performance**: Check for JavaScript errors, disable other modules
5. **Data Loss**: Verify flag-based storage is working

Emergency Commands:

```javascript
// Force re-registration
game.modules.get('bbttcc-factions').api.runDiagnostics();

// Repair faction data
game.modules.get('bbttcc-factions').api.factions.repair(actor);

// Check module status
console.log('Module Status:', {
  loaded: !!game.modules.get('bbttcc-factions'),
  active: game.modules.get('bbttcc-factions')?.active,
  apiReady: !!game.modules.get('bbttcc-factions')?.api
});
```

Updated for BBTTCC Factions v4.8.1-ENHANCED

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

Last Updated: [Current Date]