

Integration Guide - Power Components

Learn how to use Power Redact Plugin v2.0 and Power Canvas Plugin together for enhanced workflows and secure visual content creation.

Overview

The Power Components suite is designed with integration in mind. Both plugins can work independently, but when used together, they create powerful workflows for secure visual content creation, privacy-protected diagrams, and confidential documentation.

Integration Scenarios


1. Secure Visual Documentation

Use Case: Create diagrams with sensitive information that needs redaction before sharing.

Workflow:

1. **Create Canvas:** Design your diagram with all information
2. **Export to Text:** Extract text elements from canvas
3. **Apply Redaction:** Use Power Redact to identify and hide sensitive data
4. **Update Canvas:** Replace original text with redacted versions
5. **Export Securely:** Generate final diagram with protected information

Example:

```
Original Diagram: Network topology with IP addresses and passwords
 Power Redact Processing
Redacted Diagram: Network topology with [REDACTED] placeholders
```

2. Privacy-First Mind Mapping

Use Case: Create mind maps containing personal or confidential information.

Workflow:

1. **Brain Dump:** Create comprehensive mind map with all details
2. **Pattern Detection:** Let Power Redact identify sensitive patterns
3. **Selective Redaction:** Choose which information to protect
4. **Version Control:** Maintain both full and redacted versions
5. **Secure Sharing:** Export redacted version for collaboration

3. Compliance Documentation

Use Case: Technical documentation that must meet regulatory requirements.

Workflow:

1. **Document Creation:** Build technical diagrams and flowcharts
2. **Compliance Scan:** Use Power Redact to identify regulated data
3. **Automated Redaction:** Apply compliance-specific patterns
4. **Audit Trail:** Maintain logs of redaction operations
5. **Certified Export:** Generate compliant documentation

Technical Integration

Cross-Plugin Communication

Both plugins use a shared event system for seamless integration:

```
// Event bus for cross-plugin communication
interface PowerComponentsEventBus {
  // Redact content before canvas processing
  redactAndCanvas(content: string, patterns: string[]): Promise<CanvasData>;

  // Extract text from canvas for redaction
  extractCanvasText(canvasId: string): Promise<string[]>;

  // Update canvas with redacted content
  updateCanvasText(canvasId: string, redactedText: string[]): Promise<void>;

  // Export canvas with redaction metadata
  exportSecureCanvas(canvasId: string, format: ExportFormat): Promise<Blob>;
}
```

Shared Data Structures

Redaction Metadata

```
interface RedactionMetadata {
  id: string;
  originalText: string;
  redactedText: string;
  pattern: string;
  style: RedactionStyle;
  timestamp: number;
  canvasElementId?: string;
}
```

Canvas Integration Data

```
interface CanvasIntegrationData {
  canvasId: string;
  elementId: string;
  textContent: string;
  redactionApplied: boolean;
  redactionMetadata?: RedactionMetadata[];
}
```

Setup and Configuration

Enable Integration Features

1. Install Both Plugins:

- Ensure both Power Redact and Power Canvas are installed
- Enable both plugins in Obsidian settings

2. Configure Integration:

```
json
{
```

```

    "powerComponents": {
      "enableIntegration": true,
      "crossPluginEvents": true,
      "sharedStorage": true,
      "syncSettings": false
    }
  }
}

```

3. Set Integration Preferences:

- Settings → Power Components → Integration
- Enable desired integration features
- Configure default behaviors

Integration Settings

Automatic Redaction

```

{
  "autoRedaction": {
    "enabled": true,
    "triggerOnCanvasExport": true,
    "patterns": ["ssn", "email", "phone"],
    "style": "blackout",
    "confirmBeforeRedaction": true
  }
}

```

Canvas Text Extraction

```

{
  "textExtraction": {
    "includeShapes": true,
    "includeAnnotations": true,
    "preserveFormatting": false,
    "extractHiddenText": false
  }
}

```

Workflow Examples

Example 1: Secure Network Diagram

Scenario: Create a network diagram with IP addresses and credentials that needs to be shared with external consultants.

Step-by-Step Process

1. Create Network Diagram:

Power Canvas:

- Draw network topology
- Add server boxes with IP addresses
- Include connection details
- Add credential annotations

2. Extract Text for Analysis:

```
```javascript
// Extract all text elements from canvas
const textElements = await PowerCanvas.extractText(canvasId);

// Results:
[
 "Server: 192.168.1.100",
 "Database: 10.0.0.50",
 "Admin: admin@company.com",
 "Password: SecurePass123"
]
```
```

1. Apply Redaction Patterns:

```
```javascript
// Configure patterns for network security
const patterns = [
 'ip-address', // 192.168.1.100 → [REDACTED-IP]
 'email', // admin@company.com → [REDACTED-EMAIL]
 'password' // SecurePass123 → [REDACTED-PWD]
];

const redacted = await PowerRedact.processText(textElements, patterns);
```
```

1. Update Canvas with Redacted Content:

```
javascript
// Replace original text with redacted versions
await PowerCanvas.updateTextElements(canvasId, redacted);
```

2. Export Secure Version:

```
javascript
// Export with redaction metadata
const secureExport = await PowerCanvas.exportSecure(canvasId, {
  format: 'png',
  includeMetadata: false,
  watermark: 'CONFIDENTIAL'
});
```

Result

- **Original:** Full network diagram with real IPs and credentials
- **Redacted:** Same diagram with sensitive data replaced by placeholders
- **Audit Trail:** Complete log of what was redacted and when

Example 2: Personal Project Planning

Scenario: Create a project timeline with personal information, financial details, and contact information.

Workflow Implementation

1. Create Comprehensive Timeline:

```
Power Canvas Elements:
```

- Project milestones with dates
- Budget information (\$50,000 total)
- Team member contacts (john@email.com)
- Personal notes and reminders

2. Smart Pattern Detection:

```
javascript
// Power Redact automatically detects:
const detectedPatterns = {
  financial: ['$50,000', '$15,000', '$8,500'],
  email: ['john@email.com', 'sarah@company.com'],
  phone: ['555-123-4567', '800-555-0199'],
  personal: ['My birthday: 03/15/1985']
};
```

3. Selective Redaction:

```
```javascript
// Choose what to redact for different audiences
const publicVersion = await PowerRedact.processSelective(content, {
 redact: ['financial', 'personal'],
 preserve: ['email', 'phone'] // Keep for team collaboration
});

const clientVersion = await PowerRedact.processSelective(content, {
 redact: ['financial', 'personal', 'phone'],
 preserve: ['email'] // Keep business emails only
});
```
```

1. Multi-Version Export:

```
javascript
// Generate different versions for different audiences
await PowerCanvas.exportVersions(canvasId, [
  { name: 'internal', redactionLevel: 'none' },
  { name: 'team', redactionLevel: 'personal' },
  { name: 'client', redactionLevel: 'full' }
]);
```

Example 3: Educational Content with Privacy

Scenario: Create educational diagrams using real student data that must be anonymized for sharing.

Privacy-First Educational Workflow

1. Create Learning Materials:

```
Canvas Content:
- Student performance charts
- Individual progress tracking
- Contact information for parents
- Assessment scores and grades
```

2. FERPA Compliance Redaction:

```
```javascript
// Educational privacy patterns
```

```

const educationPatterns = {
 studentId: /STU\d{6}/g,
 grades: /[A-F][+-]?|\d{1,3}%/g,
 parentEmail: /parent.\w+@\w+.\w+/g,
 ssn: /\d{3}-\d{2}-\d{4}/g
};

await PowerRedact.addPatterns(educationPatterns);
...

```

### 1. Anonymization Process:

```

javascript
// Replace with anonymous identifiers
const anonymized = await PowerRedact.anonymize(content, {
 studentNames: 'Student A, Student B, Student C',
 grades: '[GRADE REDACTED]',
 contacts: '[CONTACT REDACTED]'
});

```

### 2. Compliant Export:

```

javascript
// Export with compliance metadata
await PowerCanvas.exportCompliant(canvasId, {
 standard: 'FERPA',
 anonymizationLevel: 'full',
 auditTrail: true,
 certification: true
});

```

## Advanced Integration Features

### Automated Workflows

#### Trigger-Based Redaction

```

// Automatically redact when exporting canvas
PowerCanvas.on('before-export', async (canvasData) => {
 if (canvasData.containsSensitiveData) {
 const redacted = await PowerRedact.processCanvas(canvasData);
 return redacted;
 }
 return canvasData;
});

```

## Smart Content Analysis

```
// Analyze canvas content for sensitivity
const analysis = await PowerRedact.analyzeCanvas(canvasId);

if (analysis.sensitivityScore > 0.7) {
 // High sensitivity - require redaction
 await PowerRedact.enforceRedaction(canvasId);
} else if (analysis.sensitivityScore > 0.3) {
 // Medium sensitivity - suggest redaction
 await PowerRedact.suggestRedaction(canvasId);
}
```

## Batch Processing Integration

### Process Multiple Canvases

```
// Batch process all canvases in a folder
const canvases = await PowerCanvas.getCanvasesInFolder('Confidential');

for (const canvas of canvases) {
 // Extract text content
 const textContent = await PowerCanvas.extractText(canvas.id);

 // Apply redaction
 const redacted = await PowerRedact.processText(textContent);

 // Update canvas
 await PowerCanvas.updateText(canvas.id, redacted);

 // Export secure version
 await PowerCanvas.exportSecure(canvas.id, 'png');
}
```

### Compliance Batch Processing

```
// Process all educational materials for FERPA compliance
await PowerComponents.batchProcess({
 source: 'Education/',
 patterns: ['student-data', 'grades', 'parent-info'],
 compliance: 'FERPA',
 outputFormat: 'png',
 auditTrail: true
});
```

## Visual Integration Examples

### Redaction Visualization

#### Before/After Comparison

```
// Create side-by-side comparison
const comparison = await PowerCanvas.createComparison({
 original: originalCanvasId,
 redacted: redactedCanvasId,
 layout: 'side-by-side',
 annotations: true
});
```

#### Redaction Heatmap

```
// Visualize redaction density
const heatmap = await PowerCanvas.createRedactionHeatmap(canvasId, {
 colorScale: 'red-to-green',
 showLegend: true,
 includeStats: true
});
```

### Interactive Redaction Controls

#### Toggle Redaction View

```
// Add interactive controls to canvas
await PowerCanvas.addRedactionControls(canvasId, {
 toggleButton: true,
 sliderControl: true,
 patternSelector: true
});
```

#### Progressive Disclosure

```
// Reveal information gradually
await PowerCanvas.addProgressiveDisclosure(canvasId, {
 levels: ['public', 'internal', 'confidential'],
 authentication: true,
 auditLog: true
});
```



## Integration Analytics

### Usage Tracking

#### Redaction Statistics

```
// Track redaction usage across canvases
const stats = await PowerComponents.getIntegrationStats();

console.log({
 canvasesWithRedaction: stats.redactedCanvases,
 totalRedactions: stats.redactionCount,
 mostUsedPatterns: stats.topPatterns,
 averageRedactionsPerCanvas: stats.avgRedactions
});
```

### Compliance Reporting

```
// Generate compliance reports
const report = await PowerComponents.generateComplianceReport({
 timeRange: 'last-30-days',
 standards: ['GDPR', 'HIPAA', 'FERPA'],
 includeAuditTrail: true
});
```

### Performance Monitoring

#### Integration Performance

```
// Monitor integration performance
const performance = await PowerComponents.getPerformanceMetrics();

console.log({
 averageRedactionTime: performance.redactionTime,
 canvasUpdateTime: performance.updateTime,
 exportTime: performance.exportTime,
 memoryUsage: performance.memoryUsage
});
```

## Troubleshooting Integration

### Common Integration Issues

#### Plugins Not Communicating

**Symptoms:** Features don't work together, no cross-plugin functionality

**Solutions:**

1. **Check Plugin Versions:** Ensure both plugins are up to date
2. **Verify Integration Settings:** Enable cross-plugin communication
3. **Restart Obsidian:** Reload plugins to establish connections
4. **Check Console:** Look for integration errors in developer console

#### Performance Issues

**Symptoms:** Slow processing when using both plugins together

**Solutions:**

1. **Reduce Canvas Complexity:** Simplify canvases before redaction
2. **Limit Redaction Patterns:** Use only necessary patterns
3. **Batch Process Smaller Groups:** Process fewer files at once
4. **Increase Memory Allocation:** Adjust Obsidian memory settings

**Export Problems**

**Symptoms:** Failed exports, corrupted files when using integration

**Solutions:**

1. **Check Export Settings:** Verify format compatibility
2. **Test Individual Plugins:** Ensure each plugin exports correctly
3. **Clear Cache:** Reset both plugin caches
4. **Update Dependencies:** Ensure all required libraries are current

**Debug Integration****Enable Debug Mode**

```
{
 "powerComponents": {
 "debug": {
 "integration": true,
 "crossPluginEvents": true,
 "performanceLogging": true,
 "verboseLogging": true
 }
 }
}
```

**Debug Commands**

```
// Check integration status
PowerComponents.getIntegrationStatus();

// Test cross-plugin communication
PowerComponents.testCommunication();

// Validate shared data
PowerComponents.validateSharedData();

// Performance diagnostics
PowerComponents.runDiagnostics();
```

**Best Practices****Security Best Practices**

1. **Verify Redaction:** Always review redacted content before sharing
2. **Use Strong Patterns:** Ensure patterns catch all sensitive data
3. **Maintain Originals:** Keep unredacted versions in secure locations
4. **Audit Regularly:** Review redaction logs and patterns
5. **Test Exports:** Verify exported content doesn't leak information

## Performance Best Practices

1. **Optimize Canvas Size:** Use appropriate dimensions for your needs
2. **Limit Text Elements:** Reduce number of text elements for faster processing
3. **Cache Results:** Enable caching for repeated operations
4. **Batch Similar Operations:** Group similar redaction tasks
5. **Monitor Memory:** Watch memory usage during large operations

## Workflow Best Practices

1. **Plan Integration:** Design workflows with both plugins in mind
2. **Version Control:** Maintain multiple versions of sensitive content
3. **Document Processes:** Record your integration workflows
4. **Train Users:** Ensure team members understand integration features
5. **Regular Updates:** Keep both plugins updated for best compatibility

## Future Integration Features

---

### Planned Enhancements

#### AI-Powered Redaction

- **Smart Pattern Detection:** AI identifies sensitive content automatically
- **Context-Aware Redaction:** Understands when information should be protected
- **Learning System:** Improves pattern detection based on user feedback

#### Advanced Collaboration

- **Real-time Collaboration:** Multiple users working on redacted canvases
- **Permission-Based Viewing:** Different redaction levels for different users
- **Collaborative Redaction:** Team-based redaction decision making

#### Enhanced Export Options

- **Multi-Format Batch Export:** Export to multiple formats simultaneously
- **Conditional Exports:** Different exports based on audience
- **Automated Distribution:** Send appropriate versions to different recipients

## Integration Support

---

### Getting Help

1. **Documentation:** Check plugin-specific guides first
2. **GitHub Issues:** Report integration-specific bugs
3. **Community Discord:** Ask questions and share workflows
4. **Video Tutorials:** Watch integration workflow examples

### Contributing

1. **Share Workflows:** Document your integration use cases
  2. **Report Issues:** Help improve integration features
  3. **Suggest Features:** Propose new integration capabilities
  4. **Test Beta Features:** Try new integration features early
-

## Quick Integration Reference

---

### Essential Integration Commands

```
Extract Canvas Text: PowerCanvas.extractText(canvasId)
Redact Text: PowerRedact.processText(text, patterns)
Update Canvas: PowerCanvas.updateText(canvasId, redactedText)
Export Secure: PowerCanvas.exportSecure(canvasId, options)
```

### Common Integration Patterns

```
// Basic integration workflow
const text = await PowerCanvas.extractText(canvasId);
const redacted = await PowerRedact.processText(text);
await PowerCanvas.updateText(canvasId, redacted);
await PowerCanvas.exportSecure(canvasId, 'png');
```

### Integration Settings

```
{
 "enableIntegration": true,
 "autoRedaction": true,
 "crossPluginEvents": true,
 "sharedStorage": true
}
```

---

**Ready to integrate?** Start with the [Power Redact Guide](#) (POWER\_REDACT\_GUIDE.md) and [Power Canvas Guide](#) (POWER\_CANVAS\_GUIDE.md) to understand each plugin's capabilities, then return here to combine their power.