TAMBO MCP Integration Suite - User Guide

🚀 Getting Started

First-Time Setup

1. **Open the Application**
   - Navigate to the TAMBO Routing Console Pro interface
   - The system will initialize with default settings
   - Ensure you have the necessary permissions for your role

2. **Understand Your Dashboard**

```
┌─────────────────────────────────────────┐
    │  Mode Selection: [Route Request ▼]     │
    ├─────────────────────────────────────────┤
    │  User Tier: [Pro ▼]                    │
    │  Environment: [Development ▼]          │
    ├─────────────────────────────────────────┤
    │  Request Input Area                     │
    │  [Type your request here...]            │
    ├─────────────────────────────────────────┤
    │  [Submit Request]                       │
    └─────────────────────────────────────────┘
```

🎯 Core Operations

1. Request Routing

Purpose: Route user requests to the appropriate agent using AI intelligence

Steps:
1. Set **Mode** to "Route Request"
2. Select your **User Tier** (Standard/Pro/Enterprise)
3. Choose **Environment** (Development/Production)
4. Enter your request in the payload field
5. Click **Submit Request**

Example Requests:

```
✅ Good Examples:
- "I'm having trouble logging into my account"
- "Need help with upgrading my subscription"
- "Found a bug in the navigation component"
- "Want to publish a new blog post about AI"

❌ Avoid:
- Single words without context
- Requests with sensitive information
- Extremely long requests (>500 characters)
```

Expected Response:

```
{
  "agent": "TriageAgent",
  "tier": "Pro",
  "intent": "support_request",
  "route": "/triage",
  "notes": "Detected login issue; routed to support.",
  "confidence": 0.95,
  "processing_time_ms": 127
}
```

2. Component Updates

Purpose: Modify existing components using natural language instructions

Steps:
1. Set **Mode** to “Component Update”
2. Enter **Component ID** (e.g., “nav-header”, “sidebar-menu”)
3. Write clear **update instructions**
4. Submit the request

Component ID Examples:
- `nav-header` - Main navigation header
- `sidebar-menu` - Left sidebar navigation
- `button-primary` - Primary action buttons
- `card-component` - Card layout components

Update Instruction Examples:

```
✅ Clear Instructions:
- "Make the button larger and change color to blue"
- "Add responsive design for mobile screens"
- "Include loading spinner when processing"
- "Add dark mode support with proper contrast"

❌ Unclear Instructions:
- "Make it better"
- "Fix the styling"
- "Update everything"
```

3. Agent Diagnostics

Purpose: Analyze agent performance and health status

Steps:
1. Set **Mode** to “Agent Diagnostics”
2. Enter **Agent Name** (e.g., “TriageAgent”, “ContentRouterAgent”)
3. Select **Diagnostic Scope**:
- **Fast**: Basic health check (~30 seconds)
- **Detailed**: Performance analysis (~2 minutes)
- **Comprehensive**: Full diagnostic report (~5 minutes)

Available Agents:
- `TriageAgent` - Support and error handling

- `ContentRouterAgent` - Content management
- `FeedbackMinerAgent` - Feedback analysis
- `PricingIntelligenceAgent` - Pricing optimization
- `AuditAgent` - Logging and compliance

🛠️ Advanced Features

Environment Switching

Development Mode:
- Safe testing environment
- Mock data responses
- No real component modifications
- Full logging enabled

Production Mode:
- Live system operations
- Real API calls
- Actual component changes
- Reduced logging

Real-time Features

Live Updates:
- Component changes reflect immediately
- Real-time sync across tools
- Automatic conflict resolution
- Change history tracking

Collaboration:
- Multiple users can work simultaneously
- Changes are broadcast to all sessions
- Role-based permissions enforced
- Conflict resolution alerts

👥 Role-Based Usage

🎭 CEO/Demo User

Full Access Capabilities:
- All operation modes available
- Can override safety restrictions
- Access to all diagnostic levels
- Complete component modification rights
- Real-time analytics dashboard

Demo Scenarios:

```
1. Customer Support Demo:
   - Input: "Customer can't complete checkout"
   - Expected: TriageAgent routing with confidence score

2. Content Management Demo:
   - Input: "Need to publish urgent product announcement"
   - Expected: ContentRouterAgent with fast-track routing

3. Component Update Demo:
   - Component: "checkout-button"
   - Instruction: "Make more prominent with animation"
   - Expected: Safe update with preview
```

👨‍💻 Developer

Available Operations:
- Component updates (non-protected)
- Route request testing
- Basic diagnostics
- Cross-tool searches

Common Workflows:

```
1. Component Debugging:
   Mode: Agent Diagnostics
   Agent: Component-specific agent
   Scope: Detailed

2. Feature Updates:
   Mode: Component Update
   Target: Feature components
   Safety: Always enabled

3. Integration Testing:
   Mode: Route Request
   Environment: Development
   Tier: Pro (for testing)
```

📝 Content Manager

Focused Capabilities:
- Content routing analysis
- Feedback processing
- Basic component viewing
- Content-related searches

Typical Tasks:

```
1. Content Routing:
   - "Need to schedule social media posts"
   - "Blog post needs SEO optimization"
   - "Customer testimonial requires approval"

2. Feedback Analysis:
   - Agent: FeedbackMinerAgent
   - Scope: Fast analysis
   - Focus: Content feedback trends
```

🆘 Support Agent

Support-Focused Tools:
- Triage agent access
- Error diagnostic tools
- Customer issue routing
- Basic system status

Standard Procedures:

```
1. Customer Issue Routing:
   Input: Customer's problem description
   Expected: Appropriate agent assignment
   Action: Follow routing recommendation

2. Error Diagnosis:
   Agent: TriageAgent
   Scope: Fast (for quick resolution)
   Purpose: Identify root cause quickly
```

🔍 Understanding Responses

Routing Response Breakdown

```
{
  "agent": "TriageAgent",          // Which agent will handle this
  "tier": "Pro",                   // User's access level
  "intent": "support_request",     // What the system thinks you want
  "route": "/triage",              // Technical endpoint for processing
  "notes": "Reasoning explanation", // Why this decision was made
  "confidence": 0.95,              // How sure the AI is (0-1)
  "metadata": {
    "keywords_matched": ["error", "bug"],  // Words that influenced decision
    "processing_time_ms": 127              // How long it took to decide
  }
}
```

Confidence Score Interpretation:
- **0.9-1.0**: High confidence, trust the routing
- **0.7-0.89**: Good confidence, likely correct
- **0.5-0.69**: Moderate confidence, double-check
- **Below 0.5**: Low confidence, manually review

Component Update Response

```json
{
  "analysis": "Update appears safe and feasible",
  "recommendations": [
    "Test on mobile devices",
    "Verify color contrast for accessibility",
    "Update documentation"
  ],
  "estimated_time": "15-30 minutes",
  "complexity": "Medium",
  "risks": ["Potential layout shift on smaller screens"]
}
```

Diagnostic Response

```json
{
  "status": "Healthy",
  "performance_metrics": {
    "average_response_time": 245,
    "success_rate": 0.97,
    "recent_errors": 2
  },
  "recommendations": [
    "Monitor memory usage trends",
    "Consider optimizing query performance"
  ],
  "alerts": []
}
```

⚠️ Safety & Best Practices

Protected Components

These components cannot be modified to ensure system stability:
- `core-layout` - Main application structure
- `main-navigation` - Primary navigation system
- `app-shell` - Application wrapper
- `error-boundary` - Error handling wrapper

Input Validation

The system automatically checks for:
- Dangerous code patterns
- Malicious scripts
- Invalid component references
- Rate limit compliance

Error Prevention

```
✅ Do:
- Test in Development first
- Use clear, specific instructions
- Check component exists before updating
- Monitor rate limits

❌ Don't:
- Make multiple rapid requests
- Try to modify protected components
- Use vague or ambiguous language
- Ignore error messages
```

📊 Monitoring Your Usage

Rate Limits Dashboard

```
Current Status:
├─ Requests this minute: 15/30
├─ Component updates today: 8/50
├─ Search queries this hour: 5/20
└─ Next reset: 42 seconds
```

Activity Log

- All requests are automatically logged
- View history in the diagnostics panel
- Export logs for compliance
- Track performance trends

🆘 Quick Troubleshooting

Common Issues

“Rate limit exceeded”
- **Wait**: Limits reset automatically
- **Check**: Current quota in status panel
- **Solution**: Space out requests

“Component is protected”
- **Cause**: Trying to modify core system components
- **Solution**: Target non-protected components
- **Alternative**: Request admin assistance

“Low confidence score”
- **Cause**: Unclear or ambiguous input
- **Solution**: Rewrite with more specific language
- **Example**: “Fix button” → “Make login button larger and blue”

“Component not found”
- **Check**: Component ID spelling

- **Verify**: Component exists in system
- **Solution**: Use search to find correct ID

Getting Help

1. **In-App Support**: Click the help icon for contextual guidance
2. **Documentation**: Reference this guide and technical docs
3. **Team Support**: Contact your team lead or admin
4. **System Status**: Check the status dashboard for outages

For technical details, see the Technical Implementation Guide (02_TECHNICAL.md)