SEO WORKS INTEGRATION GUIDE 🔧

COMPLETE INTEGRATION SETUP

OVERVIEW

This guide provides step-by-step instructions for integrating SEO WORKS with the Rylie SEO Hub platform, including webhook setup, task management, and automated workflows.

PART 1: INITIAL SETUP

1. PLATFORM CONFIGURATION

SEO WORKS Platform Setup:

1. **Create Integration Account:** `Integration Name: Rylie SEO Hub Type: Webhook Integration Status: Active`
2. **Generate API Credentials:** `API Key: seoworks_api_key_12345 Secret Key: seoworks_secret_67890 Webhook Token: webhook_token_abcdef`
3. **Configure Webhook Endpoints:** `Task Assignment: https://seoworks.com/api/tasks/assign Status Updates: https://rylie-hub.com/api/seoworks/webhook Health Check: https://seoworks.com/api/health`

Rylie Hub Configuration:

1. **Navigate to Super Admin Panel** → Integrations → SEO WORKS

2. **Enter SEO WORKS Credentials:** `API Endpoint: https://seoworks.com/api API Key: seoworks_api_key_12345 Secret Key: seoworks_secret_67890 Webhook URL: https://rylie-hub.com/api/seoworks/webhook`

3. **Test Connection:**

4. Click "Test Connection"

5. Verify successful authentication

6. Confirm webhook delivery

7. Save configuration

2. WEBHOOK SETUP

Outgoing Webhooks (Rylie Hub → SEO WORKS):

```
{
  "event": "task.assigned",
  "task_id": "task_12345",
  "dealership_id": "abc_auto_dealership",
  "agency_id": "rylie_seo",
  "task_type": "seo_audit",
  "priority": "high",
  "package": "GOLD",
  "details": {
    "title": "Monthly SEO Optimization",
    "description": "Comprehensive SEO audit and optimization",
    "target_keywords": ["Toyota dealer", "Honda cars", "auto financing"],
    "target_locations": ["Main City", "Nearby Town"],
    "estimated_hours": 10,
    "deadline": "2024-02-15T00:00:00Z"
  },
  "contact": {
    "business_name": "ABC Auto Dealership",
    "website": "https://abcauto.com",
    "contact_email": "manager@abcauto.com",
    "phone": "(555) 123-4567"
  }
}
```

Incoming Webhooks (SEO WORKS → Rylie Hub):

```
{
  "event": "task.completed",
  "task_id": "task_12345",
  "status": "completed",
  "completion_date": "2024-01-15T10:30:00Z",
  "deliverables": [
    {
      "type": "report",
      "title": "SEO Audit Report",
      "url": "https://seoworks.com/reports/audit_12345.pdf",
      "description": "Comprehensive SEO analysis and recommendations"
    },
    {
      "type": "document",
      "title": "Keyword Research",
      "url": "https://seoworks.com/reports/keywords_12345.xlsx",
      "description": "Target keyword analysis and strategy"
    }
  ],
  "completion_notes": "Completed comprehensive audit, identified 15
optimization opportunities",
  "quality_score": 5,
  "actual_hours": 8.5,
  "next_steps": [
    "Implement on-page optimizations",
    "Create content calendar",
    "Monitor ranking improvements"
  ]
}
```

PART 2: TASK MANAGEMENT WORKFLOWS

1. AUTOMATIC TASK CREATION

Package-Based Task Templates:

PLATINUM PACKAGE ($2,500/month):

```
{
  "initial_tasks": [
    {
      "type": "seo_audit",
      "title": "Comprehensive SEO Audit",
      "priority": "high",
      "estimated_hours": 8,
      "deadline_days": 7
    },
    {
      "type": "keyword_research",
      "title": "Advanced Keyword Research",
      "priority": "high",
      "estimated_hours": 6,
      "deadline_days": 5
    },
    {
      "type": "gbp_optimization",
      "title": "Google Business Profile Setup",
      "priority": "medium",
      "estimated_hours": 4,
      "deadline_days": 10
    },
    {
      "type": "content_calendar",
      "title": "Monthly Content Calendar",
      "priority": "medium",
      "estimated_hours": 6,
      "deadline_days": 14
    },
    {
      "type": "ppc_setup",
      "title": "PPC Campaign Setup",
      "priority": "high",
      "estimated_hours": 10,
      "deadline_days": 14
    }
  ],
  "recurring_tasks": [
    {
      "type": "blog_post",
      "frequency": "weekly",
      "estimated_hours": 3,
      "priority": "medium"
    },
    {
      "type": "performance_report",
      "frequency": "monthly",
      "estimated_hours": 2,
      "priority": "low"
    }
  ]
}
```


GOLD PACKAGE ($1,500/month):

```
{
  "initial_tasks": [
    {
      "type": "seo_audit",
      "title": "Standard SEO Audit",
      "priority": "medium",
      "estimated_hours": 6,
      "deadline_days": 10
    },
    {
      "type": "keyword_research",
      "title": "Keyword Research & Strategy",
      "priority": "medium",
      "estimated_hours": 4,
      "deadline_days": 7
    },
    {
      "type": "gbp_optimization",
      "title": "Google Business Profile Setup",
      "priority": "medium",
      "estimated_hours": 3,
      "deadline_days": 14
    }
  ],
  "recurring_tasks": [
    {
      "type": "blog_post",
      "frequency": "bi-weekly",
      "estimated_hours": 3,
      "priority": "medium"
    },
    {
      "type": "performance_report",
      "frequency": "monthly",
      "estimated_hours": 2,
      "priority": "low"
    }
  ]
}
```


SILVER PACKAGE ($750/month):

```
{
  "initial_tasks": [
    {
      "type": "seo_audit",
      "title": "Basic SEO Audit",
      "priority": "low",
      "estimated_hours": 4,
      "deadline_days": 14
    },
    {
      "type": "gbp_setup",
      "title": "Google Business Profile Basic Setup",
      "priority": "medium",
      "estimated_hours": 2,
      "deadline_days": 10
    }
  ],
  "recurring_tasks": [
    {
      "type": "blog_post",
      "frequency": "monthly",
      "estimated_hours": 3,
      "priority": "low"
    },
    {
      "type": "performance_report",
      "frequency": "quarterly",
      "estimated_hours": 2,
      "priority": "low"
    }
  ]
}
```

2. TASK ASSIGNMENT RULES

Priority Queue Assignment:

```
function assignTaskPriority(task) {
  const priorityRules = {
    'PLATINUM': {
      queue: 'high_priority',
      sla_hours: 24,
      team: 'senior_specialists'
    },
    'GOLD': {
      queue: 'standard_priority',
      sla_hours: 48,
      team: 'specialists'
    },
    'SILVER': {
      queue: 'basic_priority',
      sla_hours: 72,
      team: 'junior_specialists'
    }
  }

  return priorityRules[task.package] || priorityRules['SILVER']
}
```

Task Routing Logic:

1. **Receive Task** from Rylie Hub
2. **Determine Priority** based on package
3. **Assign to Queue** (High/Standard/Basic)
4. **Select Team Member** based on availability and expertise
5. **Send Notification** to assigned team member
6. **Track SLA** for completion timeline

PART 3: WEBHOOK IMPLEMENTATION

1. WEBHOOK SECURITY

Authentication:

```
// Verify webhook signature
function verifyWebhookSignature(payload, signature, secret) {
  const expectedSignature = crypto
    .createHmac('sha256', secret)
    .update(payload)
    .digest('hex')

  return crypto.timingSafeEqual(
    Buffer.from(signature),
    Buffer.from(expectedSignature)
  )
}
```

Request Validation:

```
// Validate incoming webhook
function validateWebhook(req) {
  const signature = req.headers['x-webhook-signature']
  const payload = JSON.stringify(req.body)

  if (!verifyWebhookSignature(payload, signature, WEBHOOK_SECRET)) {
    throw new Error('Invalid webhook signature')
  }

  return true
}
```

2. WEBHOOK HANDLERS

Task Assignment Handler (SEO WORKS):

```
app.post('/api/tasks/assign', async (req, res) => {
  try {
    // Validate webhook
    validateWebhook(req)

    const task = req.body

    // Determine priority and assignment
    const assignment = assignTaskPriority(task)

    // Create task in SEO WORKS system
    const seoworksTask = await createTask({
      id: task.task_id,
      type: task.task_type,
      priority: assignment.queue,
      dealership: task.dealership_id,
      details: task.details,
      deadline: calculateDeadline(assignment.sla_hours),
      assigned_team: assignment.team
    })

    // Send confirmation back to Rylie Hub
    await sendWebhook('https://rylie-hub.com/api/seoworks/webhook', {
      event: 'task.assigned',
      task_id: task.task_id,
      seoworks_task_id: seoworksTask.id,
      assigned_to: seoworksTask.assigned_team_member,
      estimated_completion: seoworksTask.deadline
    })

    res.json({ success: true, task_id: seoworksTask.id })

  } catch (error) {
    console.error('Task assignment failed:', error)
    res.status(500).json({ error: error.message })
  }
})
```

Status Update Handler (Rylie Hub):

```
app.post('/api/seoworks/webhook', async (req, res) => {
  try {
    // Validate webhook
    validateWebhook(req)

    const update = req.body

    // Update task status in database
    await prisma.seoworksTask.upsert({
      where: { id: update.task_id },
      update: {
        status: update.status,
        completionDate: update.completion_date ? new
Date(update.completion_date) : null,
        completionNotes: update.completion_notes,
        qualityScore: update.quality_score,
        actualHours: update.actual_hours,
        deliverables: JSON.stringify(update.deliverables),
        payload: JSON.stringify(update)
      },
      create: {
        id: update.task_id,
        taskType: update.task_type || 'unknown',
        status: update.status,
        dealershipId: update.dealership_id,
        completionDate: update.completion_date ? new
Date(update.completion_date) : null,
        completionNotes: update.completion_notes,
        payload: JSON.stringify(update)
      }
    })

    // Update related order status
    if (update.status === 'completed') {
      await updateOrderStatus(update.task_id, 'completed')
      await notifyAgencyAndDealership(update)
      await generateCompletionReport(update)
    }

    res.json({ success: true })

  } catch (error) {
    console.error('Webhook processing failed:', error)
    res.status(500).json({ error: error.message })
  }
})
```

PART 4: MONITORING AND TROUBLESHOOTING

1. HEALTH MONITORING

System Health Checks:

```
// Health check endpoint
app.get('/api/seoworks/health', async (req, res) => {
  try {
    const health = {
      status: 'healthy',
      timestamp: new Date().toISOString(),
      checks: {
        database: await checkDatabase(),
        webhook_delivery: await checkWebhookDelivery(),
        task_processing: await checkTaskProcessing(),
        api_connectivity: await checkAPIConnectivity()
      }
    }

    const allHealthy = Object.values(health.checks).every(check => check.status
=== 'ok')

    res.status(allHealthy ? 200 : 503).json(health)

  } catch (error) {
    res.status(503).json({
      status: 'unhealthy',
      error: error.message,
      timestamp: new Date().toISOString()
    })
  }
})
```

Performance Metrics:

```
 // Track webhook performance
const webhookMetrics = {
  delivery_success_rate: 0.99,
  average_response_time: 150, // milliseconds
  failed_deliveries_24h: 2,
  retry_success_rate: 0.95
}

// Track task processing
const taskMetrics = {
  average_completion_time: {
    'PLATINUM': 18, // hours
    'GOLD': 36,     // hours
    'SILVER': 60    // hours
  },
  quality_scores: {
    'PLATINUM': 4.8,
    'GOLD': 4.6,
    'SILVER': 4.4
  },
  sla_compliance: 0.96
}
```

2. ERROR HANDLING

Webhook Retry Logic:

```
async function sendWebhookWithRetry(url, payload, maxRetries = 3) {
  for (let attempt = 1; attempt <= maxRetries; attempt++) {
    try {
      const response = await fetch(url, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          'X-Webhook-Signature': generateSignature(payload)
        },
        body: JSON.stringify(payload),
        timeout: 10000 // 10 second timeout
      })

      if (response.ok) {
        return response
      }

      throw new Error(`HTTP ${response.status}: ${response.statusText}`)

    } catch (error) {
      console.error(`Webhook attempt ${attempt} failed:`, error)

      if (attempt === maxRetries) {
        // Log to dead letter queue for manual review
        await logFailedWebhook(url, payload, error)
        throw error
      }

      // Exponential backoff
      await sleep(Math.pow(2, attempt) * 1000)
    }
  }
}
```

Common Issues and Solutions:

Webhook Delivery Failures:

```
// Check webhook endpoint accessibility
async function diagnoseWebhookFailure(url, payload) {
  const diagnosis = {
    url_accessible: false,
    ssl_valid: false,
    response_time: null,
    error_details: null
  }

  try {
    const start = Date.now()
    const response = await fetch(url, { method: 'HEAD' })
    diagnosis.response_time = Date.now() - start
    diagnosis.url_accessible = response.ok

  } catch (error) {
    diagnosis.error_details = error.message

    if (error.code === 'CERT_HAS_EXPIRED') {
      diagnosis.ssl_valid = false
    }
  }

  return diagnosis
}
```

Task Assignment Failures:

```
// Validate task data before assignment
function validateTaskAssignment(task) {
  const required = ['task_id', 'task_type', 'dealership_id', 'package']
  const missing = required.filter(field => !task[field])

  if (missing.length > 0) {
    throw new Error(`Missing required fields: ${missing.join(', ')}`)
  }

  const validTypes = ['seo_audit', 'keyword_research', 'blog_post',
'gbp_optimization', 'ppc_setup']
  if (!validTypes.includes(task.task_type)) {
    throw new Error(`Invalid task type: ${task.task_type}`)
  }

  const validPackages = ['PLATINUM', 'GOLD', 'SILVER']
  if (!validPackages.includes(task.package)) {
    throw new Error(`Invalid package: ${task.package}`)
  }

  return true
}
```

PART 5: TESTING AND VALIDATION

1. INTEGRATION TESTING

Test Webhook Delivery:

```
# Test outgoing webhook (Rylie Hub → SEO WORKS)
curl -X POST https://seoworks.com/api/tasks/assign \
  -H "Content-Type: application/json" \
  -H "X-Webhook-Signature: sha256=..." \
  -d '{
    "event": "task.assigned",
    "task_id": "test_task_123",
    "dealership_id": "test_dealership",
    "task_type": "seo_audit",
    "package": "GOLD"
  }'
```

```
# Test incoming webhook (SEO WORKS → Rylie Hub)
curl -X POST https://rylie-hub.com/api/seoworks/webhook \
  -H "Content-Type: application/json" \
  -H "X-Webhook-Signature: sha256=..." \
  -d '{
    "event": "task.completed",
    "task_id": "test_task_123",
    "status": "completed",
    "completion_date": "2024-01-15T10:30:00Z"
  }'
```

End-to-End Testing:

1. **Create Test Dealership** in Rylie Hub
2. **Trigger Task Creation** via AI chat or manual creation
3. **Verify Webhook Delivery** to SEO WORKS
4. **Simulate Task Completion** in SEO WORKS
5. **Verify Status Update** webhook delivery
6. **Check Final State** in Rylie Hub dashboard

2. PERFORMANCE TESTING

Load Testing:

```
// Simulate high webhook volume
async function loadTestWebhooks(concurrency = 10, duration = 60) {
  const startTime = Date.now()
  const results = []

  while (Date.now() - startTime < duration * 1000) {
    const promises = []

    for (let i = 0; i < concurrency; i++) {
      promises.push(sendTestWebhook())
    }

    const batchResults = await Promise.allSettled(promises)
    results.push(...batchResults)

    await sleep(100) // Brief pause between batches
  }

  return analyzeResults(results)
}
```

PART 6: DEPLOYMENT CHECKLIST

PRE-DEPLOYMENT:

- [ ] API credentials configured
- [ ] Webhook endpoints tested
- [ ] SSL certificates valid
- [ ] Database migrations applied
- [ ] Environment variables set
- [ ] Monitoring alerts configured

DEPLOYMENT:

- [ ] Deploy Rylie Hub updates
- [ ] Deploy SEO WORKS integration
- [ ] Test webhook connectivity

- [ ] Verify task assignment flow
- [ ] Test status update flow
- [ ] Monitor error rates

POST-DEPLOYMENT:

- [ ] Monitor webhook delivery rates
- [ ] Check task processing times
- [ ] Verify data consistency
- [ ] Test error handling
- [ ] Monitor performance metrics
- [ ] Validate business workflows

The SEO WORKS integration is now complete and ready for production use! 🚀