Orq.ai Customer Support Guidelines

Version 1.0 | October 2025

Table of Contents

Introduction

About This Guide

This document provides comprehensive guidelines for customer support teams assisting Orq.ai users. Orq.ai is a Generative AI Collaboration Platform where software teams build, ship, and optimize LLM applications at scale.

Support Philosophy

- **User-centric**: Focus on understanding the customer's use case and goals
- **Educational**: Help users understand the platform's capabilities and best practices
- **Proactive**: Anticipate common issues and provide preventative guidance
- **Collaborative**: Work with technical and non-technical team members alike

Key Platform Benefits

- No-code interface for cross-functional teams
- Unified platform for prompt engineering, experimentation, deployment, and monitoring
- Support for multiple LLM providers and models
- Built-in RAG capabilities with Knowledge Bases
- SOC 2-certified, GDPR-compliant, and EU AI Act-ready

Platform Overview

Core Modules

1. Model Garden

- Central hub for browsing and activating available LLM models
- Supports models from major providers (OpenAI, Anthropic, Cohere, Google, AWS,

etc.)
- Filter by model type (text generation, embedding, vision, etc.)

Support Tip: Users must activate models in the Model Garden before using them in Deployments or Experiments.

2. Playground

- Quick testing environment for prompts and model configurations
- Ideal for rapid prototyping and one-off tests
- No evaluation or batch processing capabilities

When to Use: Single prompt testing, quick model comparisons, initial exploration

3. Experiments

- Scalable testing environment for multiple model configurations
- Run evaluations across datasets with built-in metrics
- Compare performance across different LLMs and parameter settings
- Supports LLM-as-a-Judge and RAGAS evaluators

When to Use: A/B testing, model evaluation, quality assurance before deployment

4. Deployments

- Production-ready API endpoints for LLM applications
- Version control and instant updates without code changes
- Integrated with Knowledge Bases for RAG applications
- Support for streaming responses and function calling

When to Use: Production applications, API integration, live user-facing features

5. Knowledge Bases

- Managed RAG infrastructure with vector database
- Upload documents (PDF, TXT, DOCX, CSV, XML)
- Configure chunking strategies, embedding models, and retrieval settings
- Support for Agentic RAG to improve retrieval quality

When to Use: Domain-specific applications, FAQ bots, document Q&A, customer support automation

6. Logs & Traces

- Real-time monitoring of deployment usage
- Track requests, responses, latency, and costs
- View conversation traces and debug issues
- Monitor evaluator scores and quality metrics

Common Support Topics

Getting Started

Account Setup

Issue: New user needs to get started **Resolution Steps**:

1. Verify user has access to Orq.ai Studio workspace
2. Guide to Model Garden to activate required models
3. Walk through creating first Deployment
4. Demonstrate API key generation in Workspace Settings
5. Provide SDK installation instructions (Node.js or Python)

First API Call

Sample Code Snippets:

Node.js:

```
import { Orq } from "@orq-ai/node";

const client = new Orq({
  apiKey: process.env.ORQ_API_KEY,
  environment: "production"
});

const response = await client.deployments.invoke({
  key: "your-deployment-key",
  context: { environments: [] },
  metadata: { "source": "customer-support" }
});
```

Python:

```
import os
from orq_ai_sdk import Orq

client = Orq(
    api_key=os.environ.get("ORQ_API_KEY"),
    environment="production"
)

response = client.deployments.invoke(
    key="your-deployment-key",
    context={"environments": []},
    metadata={"source": "customer-support"}
)
```

Authentication Issues

Problem: "API key not working"

Troubleshooting:

1. Verify API key is correctly copied (no extra spaces)
2. Check key hasn't been revoked in Workspace Settings > API Keys
3. Confirm HTTPS is being used (HTTP will fail)
4. Verify proper authentication header format: `Authorization: Bearer <API_KEY>`

Problem: "403 Forbidden errors"

Resolution:

- Check user has appropriate workspace permissions
- Verify API key belongs to correct workspace
- Contact admin to review role-based access controls

Deployment Issues

Problem: "Model not available in Deployment"

Resolution:

1. Navigate to Model Garden
2. Find desired model and toggle activation
3. Return to Deployment and refresh model list
4. If model still missing, check workspace permissions

Problem: "Deployment not updating with new prompt changes"

Resolution:

1. Ensure user clicked **Deploy** button (top-right) to save changes
2. Verify deployment is using correct version
3. Check API calls are targeting correct deployment key
4. Clear any client-side caching if applicable

Problem: "Slow response times"

Troubleshooting:

1. Check Logs to identify latency sources (retrieval vs. generation)
2. For RAG applications, review Knowledge Base chunk limits
3. Consider using faster models (e.g., GPT-3.5 vs GPT-4)
4. Implement streaming for better user experience
5. Review concurrent request limits

Knowledge Base & RAG

Problem: "RAG returning irrelevant results"

Resolution Steps:

1. **Review chunking strategy**: Adjust chunk_size and strategy (semantic vs. fixed)
2. **Check embedding model**: Ensure appropriate model for domain (e.g., cohere/embed-english-v3.0)

3. **Adjust retrieval settings**:
    - Increase similarity threshold to be more selective
    - Adjust chunk_limit to retrieve more/fewer results
    - Enable reranking models for better relevance
4. **Verify source documents**: Ensure uploaded files contain relevant information
5. **Enable Agentic RAG**: Improves retrieval quality and performance

Problem: "Unable to upload document to Knowledge Base"

Troubleshooting:

1. Verify file format is supported (TXT, PDF, DOCX, CSV, XML)
2. Check file size limits
3. Ensure document has extractable text (not scanned images without OCR)
4. Review error messages in upload interface

Problem: "Knowledge Base not connected to Deployment"

Resolution:

1. Open Deployment configuration
2. Navigate to variant settings
3. Ensure Knowledge Base variable is selected in system prompt
4. Variable should appear in dark blue if properly configured
5. Save and Deploy changes

SDK & Integration Issues

Problem: "Module not found" errors

Resolution:

```
# Node.js
npm install @orq-ai/node zod

# Python
pip install orq-ai-sdk
```

Problem: "Streaming not working"

Sample Implementation:

```
const result = await client.deployments.stream({
  key: "deployment-key"
});

for await (const event of result) {
  console.log(event);
}
```

Problem: "Contact ID not tracking correctly"

Resolution:

```
const client = new Orq({
  apiKey: process.env.ORQ_API_KEY,
  environment: "production",
  contactId: "unique-user-identifier"  // Add this
});
```

Evaluation & Testing

Problem: "Need to evaluate model performance"

Guidance:

1. Create an Experiment (not Playground)
2. Upload test dataset or use existing one
3. Add evaluators:
    - **RAGAS metrics**: For RAG quality (context precision, recall, faithfulness)
    - **LLM-as-a-Judge**: For custom evaluation criteria
    - **Custom code evaluators**: For specific business logic
4. Run experiment across multiple models/configurations
5. Review heatmap visualization to identify outliers
6. Export results for detailed analysis

Problem: "How to use evaluators as guardrails"

Configuration:

1. In Deployment settings, add evaluators
2. Set evaluators to run on:
    - **Input**: Block inappropriate user inputs
    - **Output**: Block problematic model responses
3. Define threshold scores for blocking
4. Test thoroughly before production deployment

Troubleshooting Guide

Diagnostic Checklist

When a user reports an issue, gather this information:

1. **Environment Details**:
    - Workspace name
    - Deployment key or Knowledge Base ID
    - SDK version (Node.js/Python)
    - Production vs. staging environment
2. **Error Information**:
    - Exact error message

    - HTTP status code (if applicable)
    - Timestamp of occurrence
    - Frequency (one-time vs. recurring)
3. **Recent Changes**:
    - New deployments or updates
    - Model configuration changes
    - Knowledge Base modifications
    - SDK updates

Common Error Codes

Code	Meaning	Resolution
401	Unauthorized	Check API key validity and format
403	Forbidden	Verify permissions and access controls
404	Not Found	Confirm deployment key, prompt version, or contact ID exists
429	Rate Limited	Review usage limits; consider upgrading plan
500	Internal Error	Check platform status; escalate if persists

Performance Optimization

Latency Issues

1. Use appropriate model for task (smaller models for simple tasks)
2. Implement streaming for real-time feedback
3. Optimize Knowledge Base retrieval (chunk limits, embedding models)
4. Consider caching frequently requested responses
5. Monitor concurrent request patterns

Cost Optimization

1. Use smaller, efficient models where appropriate
2. Implement smart prompt caching
3. Set reasonable token limits (max_tokens parameter)
4. Monitor usage in Logs dashboard
5. Use evaluators to ensure quality without excessive testing

Quality Optimization

1. Use Experiments to test before deployment
2. Implement multiple evaluators (RAGAS + LLM-as-a-Judge)
3. Collect user feedback systematically
4. Iterate on prompts based on real usage data
5. A/B test significant changes

Best Practices

For Customer Support Teams

Communication

- **Use clear, jargon-free language** when possible
- **Provide context** for technical recommendations
- **Share relevant documentation links** from docs.orq.ai
- **Set realistic expectations** about resolution times
- **Follow up** to ensure issues are fully resolved

Documentation

- **Log all issues** in support system with full details
- **Tag by category** (authentication, deployment, RAG, etc.)
- **Create internal knowledge base** of common resolutions
- **Update this guide** with new patterns and solutions

Collaboration

- **Escalate complex issues** promptly (see Escalation Procedures)
- **Involve technical team** for architecture questions
- **Share customer feedback** with product team
- **Coordinate with sales** on enterprise feature requests

For End Users

Development Workflow

1. **Start in Playground** for quick experiments
2. **Move to Experiments** for systematic testing
3. **Deploy to staging** environment first
4. **Monitor with Logs** before production rollout
5. **Iterate based on data** from production usage

Security & Compliance

- **Never share API keys** in public repositories or client-side code
- **Use environment variables** for sensitive credentials
- **Implement PII masking** for sensitive data
- **Choose appropriate hosting region** (US vs. EU)
- **Review role-based access controls** regularly

Prompt Engineering

- **Be specific and clear** in instructions
- **Use examples** (few-shot learning) when possible
- **Set appropriate temperature** (lower for factual, higher for creative)
- **Define output format** explicitly
- **Test edge cases** thoroughly

RAG Implementation

- **Curate quality sources** for Knowledge Bases
- **Test chunking strategies** for your document types
- **Monitor retrieval relevance** regularly
- **Update knowledge sources** as information changes
- **Use Agentic RAG** for complex retrieval needs

Escalation Procedures

When to Escalate

Escalate to Technical Team when:

- Platform outage or widespread issues
- Data integrity concerns
- Security vulnerabilities
- Custom integration requirements
- Enterprise feature requests
- Bugs in SDK or API
- Performance degradation across multiple users

Escalation Process

Level 1: Support Team (Response: 4-24 hours)

- Handle common issues using this guide
- Provide workarounds when available
- Gather comprehensive diagnostic information

Level 2: Technical Team (Response: 1-3 business days)

- Deep technical debugging
- SDK/API bug fixes
- Platform configuration issues
- Integration guidance

Level 3: Engineering Team (Response: 3-5 business days)

- Core platform bugs
- Feature development
- Architecture consultations
- Performance optimization

Escalation Template

```
**Subject**: [ESCALATION] Brief description

**Priority**: High/Medium/Low

**Affected User(s)**:
- Workspace: [name]
- Contact: [name/email]
- Account Type: [Enterprise/Standard]

**Issue Summary**:
[Clear description of problem]

**Business Impact**:
```

```
[How this affects user's operations]

**Reproduction Steps**:
1. [Step by step]
2. ...

**Diagnostic Information**:
- Environment: [Production/Staging]
- Deployment Key: [if applicable]
- Error Messages: [exact text]
- Timestamps: [when occurred]
- SDK Version: [if applicable]

**Attempted Solutions**:
[What has been tried already]

**Urgency Justification**:
[Why this needs priority attention]
```

FAQs

Platform & Features

Q: What's the difference between Playground and Experiments? A: Playground is for quick, one-off tests. Experiments allow batch testing across datasets with evaluation metrics, ideal for comparing multiple configurations before deployment.

Q: Can I use my own API keys for LLM providers? A: Yes, navigate to the Integrations tab to configure your own provider API keys.

Q: How do I update a prompt without redeploying my application? A: Make changes in the Deployment configuration, click the Deploy button, and changes are live immediately. No code changes needed in your application.

Q: Is there a rate limit on API calls? A: Yes, rate limits vary by plan. Check your plan details or contact support for specific limits.

Q: Can I use Orq.ai on-premises? A: Yes, Orq.ai supports cloud, VPC, and on-premises deployment options for enterprise customers.

Knowledge Bases & RAG

Q: What embedding models are available? A: Multiple models from OpenAI, Cohere, and other providers. View available models in Model Garden filtered by "Embedding" type. Format: `supplier/model_name` (e.g., `cohere/embed-english-v3.0`).

Q: Can I connect my own vector database? A: Yes, Orq.ai supports integration with thirdparty vector databases like Pinecone, Qdrant, and Weaviate while using Orq for orchestration.

Q: How do I improve RAG accuracy? A:

1. Optimize chunking strategy (semantic vs. fixed)
2. Use domain-appropriate embedding models
3. Adjust similarity threshold
4. Enable reranking models
5. Turn on Agentic RAG
6. Curate high-quality source documents

Q: Can Knowledge Bases handle images or audio? A: Currently supports text-based documents (PDF, TXT, DOCX, CSV, XML). For PDFs with images, text extraction is performed.

Evaluation & Monitoring

Q: What evaluators are available? A:

- **RAGAS**: Context precision, recall, faithfulness, answer relevance
- **LLM-as-a-Judge**: Custom evaluation criteria using LLMs
- **Custom Code**: Your own evaluation logic

Q: How do I use evaluators as guardrails? A: Configure evaluators in Deployment settings to run on inputs or outputs with defined threshold scores that block unwanted content.

Q: Where can I see usage costs? A: Navigate to Logs dashboard to view request costs, token usage, and aggregate spending per deployment.

Q: Can I export logs for analysis? A: Yes, logs can be exported for external analysis and monitoring.

Integration & Development

Q: Which SDKs are available? A: Official SDKs for Node.js (`@orq-ai/node`) and Python (`orq-ai-sdk`).

Q: Does Orq.ai support streaming responses? A: Yes, use the `deployments.stream()` method in the SDK for server-sent events streaming.

Q: How do I track individual users? A: Pass a `contactId` when initializing the SDK client. This tracks all usage for that contact in the platform.

Q: Can I use Orq.ai with serverless functions? A: Yes, Orq.ai works well with serverless architectures (AWS Lambda, Vercel Functions, etc.).

Security & Compliance

Q: Is Orq.ai SOC 2 certified? A: Yes, Orq.ai is SOC 2-certified, GDPR-compliant, and EU AI Act-ready.

Q: How is data handled for EU customers? A: You can choose EU-based model hosting to keep data within European infrastructure.

Q: Does Orq.ai support PII masking? A: Yes, built-in PII and response masking features are available to protect sensitive data.

Q: Who can access my data? A: Role-based access controls allow you to define permissions at organization, workspace, and project levels.

Additional Resources

Documentation

- Main Documentation: https://docs.orq.ai
- API Reference: https://docs.orq.ai/reference
- Quick Start Guide: https://docs.orq.ai/docs/quick-start

Contact Information

- General Support: https://support.claude.com (for platform inquiries)
- Sales & Enterprise: [email protected]
- Documentation: https://docs.claude.com

SDK Resources

- Node.js SDK: https://github.com/orq-ai/orq-node
- Python SDK: Available via pip
- Code Examples: https://github.com/orq-ai/orq-cookbooks

Community & Updates

- LinkedIn: https://www.linkedin.com/company/orqai
- Platform Updates: Check in-app notifications
- Blog: https://orq.ai/blog

Appendix

Model Configuration Parameters

Common parameters available across Playground, Experiments, and Deployments:

Parameter	Description	Typical Range
`temperature`	Controls randomness (lower = more focused)	0.0 - 1.0
`max_tokens`	Maximum response length	1 - 4096+
`top_p`	Nucleus sampling threshold	0.0 - 1.0
`frequency_penalty`	Reduces repetition	0.0 - 2.0
`presence_penalty`	Encourages topic diversity	0.0 - 2.0
`stop`	Sequences where generation stops	Array of strings

Knowledge Base Configuration

Setting	Description	Recommendation
`chunk_size`	Size of text chunks	256-512 tokens for most use cases
`strategy`	Chunking method	"semantic" for natural boundaries
`threshold`	Similarity threshold	0.7-0.8 for balanced retrieval
`chunk_limit`	Max chunks retrieved	3-5 for focused context
`embedding_model`	Model for vectorization	Domain-specific when available

Troubleshooting Checklist

Before escalating, verify:

- [ ] API key is valid and correctly formatted
- [ ] Model is activated in Model Garden
- [ ] Deployment has been saved with Deploy button
- [ ] Correct deployment key is being used
- [ ] All required parameters are provided
- [ ] Network connectivity to Orq.ai API
- [ ] SDK is latest version
- [ ] Environment variables are properly set
- [ ] HTTPS (not HTTP) is being used
- [ ] Recent platform status is normal

Document Version: 1.0
Last Updated: October 2025
Maintained By: Customer Support Team
Review Cycle: Quarterly

For questions about this guide, contact the Support Team Lead.