Agent Commands

```bash
```

~~~
# Ask (QA mode - $0.01)
ask-qa what happened

# Ask (Summary mode - $0.05)
ask-sum elaborate on the themes

# Ask (Auto mode - $0.05, intelligently picks mode)
ask what were the key insights

# Upload (<100MB: FREE, 100MB+: $0.05)
upload /path/to/file.pdf
upload ./interviews/large_video.mp4

# Exit
exit
````# Interview RAG + ATXP — Quick Reference Card

Print this or keep it open while building! 🚀

---

## 3-Terminal Setup
```bash
# Terminal 1: FastAPI RAG (Port 8000)
cd interview-rag
uvicorn main:app --reload --port 8000

# Terminal 2: Python MCP Server (Port 8001)
cd interview-rag-atxp/mcp-server
source venv/bin/activate
python mcp_server.py

# Terminal 3: Node.js ATXP Agent (Interactive)
cd interview-rag-atxp/atxp-agent
npm run dev
~~~

## MCP Server Tools

Tool	Cost	Command	Time
````
````

```
| `ask_rag` (QA) | $0.01 | `ask-qa <question>` | 2-5s |
| `ask_rag` (Summary) | $0.05 | `ask-sum <question>` | 5-15s |
| `ask_rag` (Auto) | $0.05 | `ask <question>` | 5-15s |
| `upload_file` | Dynamic | `upload <path>` | 10-60s |

---

## Upload Pricing
````
File Size | Price | Example
0-99 MB | FREE | interview.pdf (50 MB)
100+ MB | $0.05 | conference.mp4 (120 MB)

Agent Commands

```bash
# Search
search what are the main topics

# Ask (auto mode)
ask what were the key insights

# Ask (QA mode - concise)
ask-qa what happened

# Ask (Summary mode - detailed)
ask-sum elaborate on the themes

# Upload
upload /path/to/file.pdf
upload ./interviews/interview_large.mp4

# Exit
exit
```

Key File Locations

```
interview-rag-atxp/
├── mcp-server/
```

```
│   ├── venv/               # Python virtual env
│   ├── mcp_server.py           # MCP server code
│   ├── requirements.txt        # Python deps
│   └── .env                # (optional) config
│
├── atxp-agent/
│   ├── node_modules/           # NPM packages
│   ├── agent.ts                # ATXP agent code
│   ├── package.json            # Node config
│   ├── tsconfig.json           # TypeScript config
│   ├── .env                # ATXP connection string
│   └── .env.example            # Template
```

Critical Configuration

MCP Server (`mcp_server.py`)

```python
RAG_BASE_URL = "http://localhost:8000"  # Match your FastAPI port
```

ATXP Agent (`.env`)

```
ATXP_CONNECTION_STRING=your_connection_string_here
```

Troubleshooting Quick Fixes

Problem	Solution
"Connection refused"	FastAPI not running? Check Terminal 1
"Unknown tool"	MCP server crashed? Check Terminal 2 output
"ATXP_CONNECTION_STRING not set"	Create `.env` in atxp-agent folder
Slow responses	Gemini API is slow on first call. Normal.
Payment failed	Check ATXP account has funds
File upload stuck	Large file? May take 1-2 mins to process

Performance Expectations

```
Operation           | Time      | Note
search              | 2-5s      | Fast
ask (small context)  | 5-10s    | Depends on Gemini
ask (large context)  | 10-20s   | More tokens = slower
upload (small, <10MB) | 5-10s   | Quick
upload (100MB)       | 30-60s   | Whisper transcription is slow
upload (200MB)       | 60-90s   | Could take longer
```

Testing Checklist

- [ ] All 3 services running (check all terminals)
- [ ] FastAPI responds to `http://localhost:8000/`
- [ ] MCP server prints "Server ready" message
- [ ] ATXP agent shows `🤖 >` prompt
- [ ] Can run: `search what`
- [ ] Can run: `ask what`
- [ ] Can run: `upload ./test_file.pdf`
- [ ] Payment goes through without error
- [ ] Retrieved chunks show up in results
- [ ] Gemini answer is relevant

Demo Commands (Copy-Paste Ready)

```bash
# QA test ($0.01)
ask-qa what are the main topics discussed

# Summary test ($0.05)
ask-sum summarize the key insights

# Upload test (need a real file <100MB for free)
upload ~/Downloads/sample_interview.pdf

# Upload large test ($0.05)
upload ~/Downloads/large_video_120mb.mp4
```

Important URLs

Service	URL	Purpose
FastAPI Docs	`http://localhost:8000/docs`	API testing
Swagger UI	`http://localhost:8000/redoc`	API reference
ATXP Docs	`https://docs.atxp.ai`	ATXP reference
ATXP Account	`https://docs.atxp.ai/client/create_an_account`	Get connection string

Friction Log Sections

When documenting, cover:

1. **Setup Time** — How long each step took
2. **Pain Points** — Specific things that were confusing
3. **What Worked** — Positive surprises
4. **Suggestions** — 1-3 concrete improvements
5. **Would You Use It?** — 1-10 rating + reasoning

Agent Interactive Mode Examples

```
🤖 > search interview insights
🔍 Searching RAG for: "interview insights"
💰 Cost: $0.01
Result: ...

🤖 > ask what were the main themes discussed
💬 Asking RAG: "what were the main themes..."
💰 Cost: $0.05
Result: ...

🤖 > upload /Users/jane/interview_150mb.mp4
📤 Uploading file: "/Users/jane/interview_150mb.mp4"
💰 Cost: Dynamic based on file size
Result: ✅ File uploaded! Cost: $0.50
```

```
🤖 > exit
Goodbye! 👋
```

Environment Variables

```bash
# MCP Server (optional)
RAG_BASE_URL=http://localhost:8000
RAG_TIMEOUT=120

# ATXP Agent (required)
ATXP_CONNECTION_STRING=your_string_here

# Debug mode (optional)
DEBUG=true
```

Next Steps After Testing

1. ✅ Document pain points in friction log
2. ✅ Note what was smooth
3. ✅ Write 3 specific suggestions
4. ✅ Time each step for your log
5. ✅ Test all 3 tools at least once
6. ✅ Attempt a large file upload (200+ MB) to test pricing

Key Insights to Share with Founder

Show you understand:

- ✅ Why tiered pricing makes sense (cost scales with file size)
- ✅ Cross-platform challenges (Python + Node communication)
- ✅ Payment validation is critical (needs good error handling)
- ✅ MCP protocol is powerful but has learning curve

Demonstrate:

- ✅ You can build working integrations
- ✅ You think about real-world monetization
- ✅ You document friction and suggest fixes
- ✅ You tested thoroughly and found edge cases

Time Estimates

Activity	Time
Read setup guide	5 min
Install dependencies	10 min
Copy code files	5 min
Start all 3 services	5 min
Test search	2 min
Test ask	3 min
Test small upload	5 min
Test large upload	10 min
Document friction log	15 min
Total	**~60 min**

Good luck! You've got this! 🚀

Questions? Check the setup guide or ATXP docs first.