How to Build an Agent for IBM i

Jesse Gorzinski, IBM
Business Architect, Quantum and AI

Session Agenda

1. Quick intro to agentic AI
2. What Makes IBM i Special for Agents (the why)
3. Building Blocks & Practical Patterns (the how)
4. How MCP fits into the picture
5. Some of many approaches
6. Securing agents
7. Closing thoughts

1. Quick intro to agentic AI

The foundations

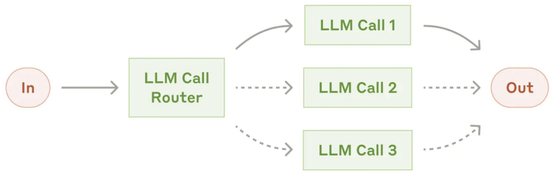
Evolving LLMs

Single-LLM Features

Summarization, classification, extraction

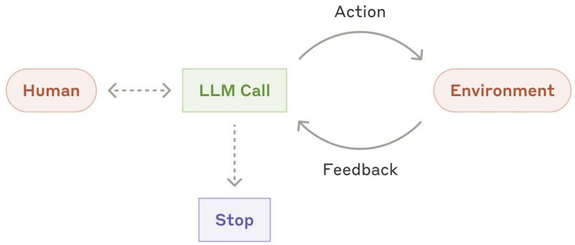
Workflows

LLMs orchestrated by code



Agents

LLMs deciding their own trajectories



???

Agency?

Capability?

Self-healing?

Intro to Agentic AI

Agentic AI refers to AI systems that use large language models to **reason through problems iteratively** — deciding what information to gather, which tools to call, and how to interpret results — in order to accomplish a **goal**.

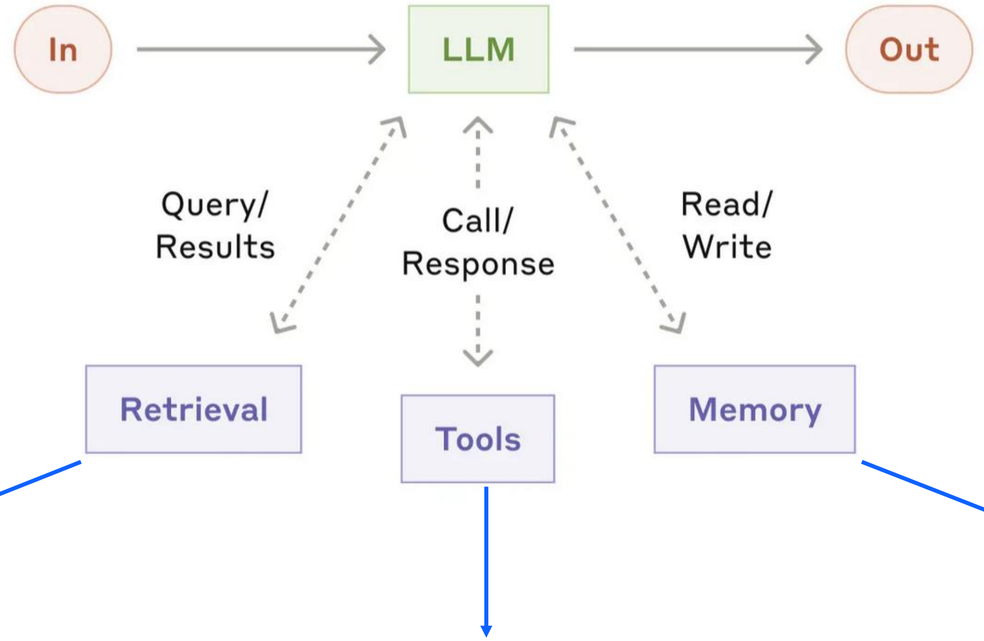
Unlike a chatbot that only generates text responses, an agent can take action through structured tools (query databases, check system status, execute commands) exposed via standard protocols like MCP. The human defines the objective and the boundaries. The agent figures out the steps.

What this looks like to me

```
env = Environment()
tools = Tools(env)
system_prompt = "Goals, contraints, how to act"
task = "Peform task X, Y, Z"

while True:
  action = llm.run(task, system_prompt + env.state)
  env.state = tools.run(action)
```

Building block: The augmented LLM

- **Basic building block:** an LLM enhanced with augmentations such as retrieval, tools, and memory



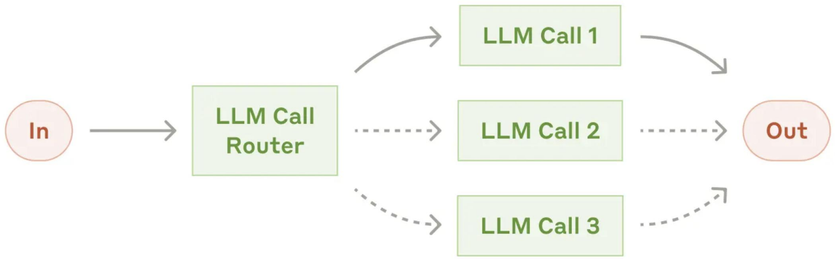
- Fetch system information
- Gather relevant information from the database
- Fetch relevant documentation

- `Run SQL` tool
- generate SQL statements given relevant information and context
- Call SQL services

- Conversation history
- Agent learning
- Retain system meta-data (database structure, coding conventions, etc)

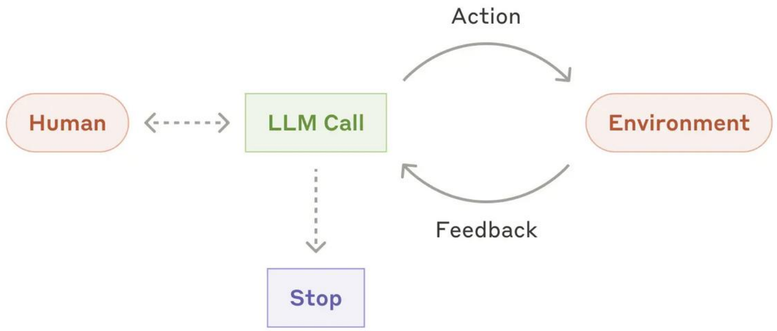
Workflows

- Systems where LLMs use tools and are **orchestrated** though pre-defined code paths
- “deterministic” scope
- workflows offer predictability and consistency for well-defined tasks



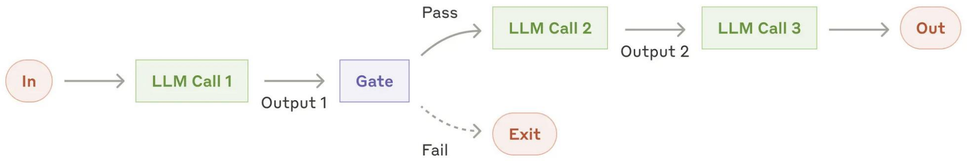
Agents

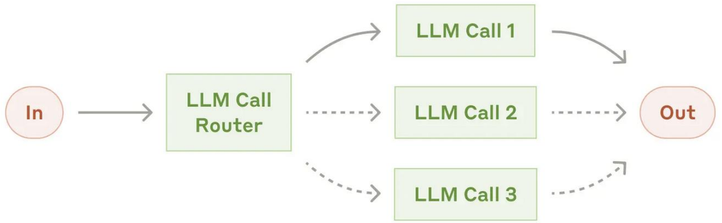
- Systems where LLMs **dynamically** direct their own processes and tool usage to accomplish a task
- LLMs “decide” which steps to take in a given context
- Are better when flexibility and decision making are needed at scale

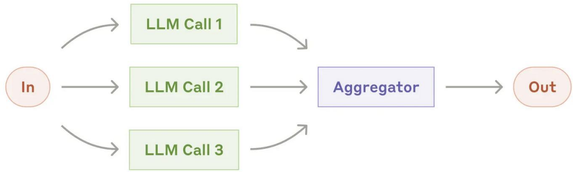


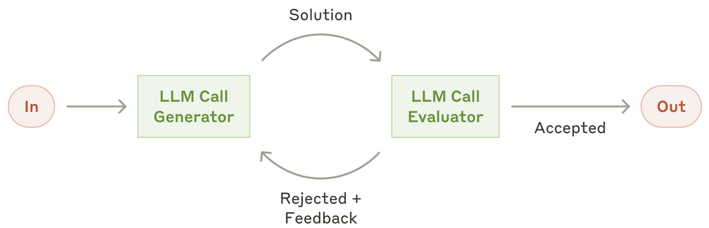
Agent Patterns

- **Prompt Chaining:** Sequential steps
- **Routing:** Classify and delegate
- **Parallelization:** simultaneous execution
- **Orchestrator-Workers:** Dynamic task breakdown
- **Evaluator-Optimizer:** Iterative refinement



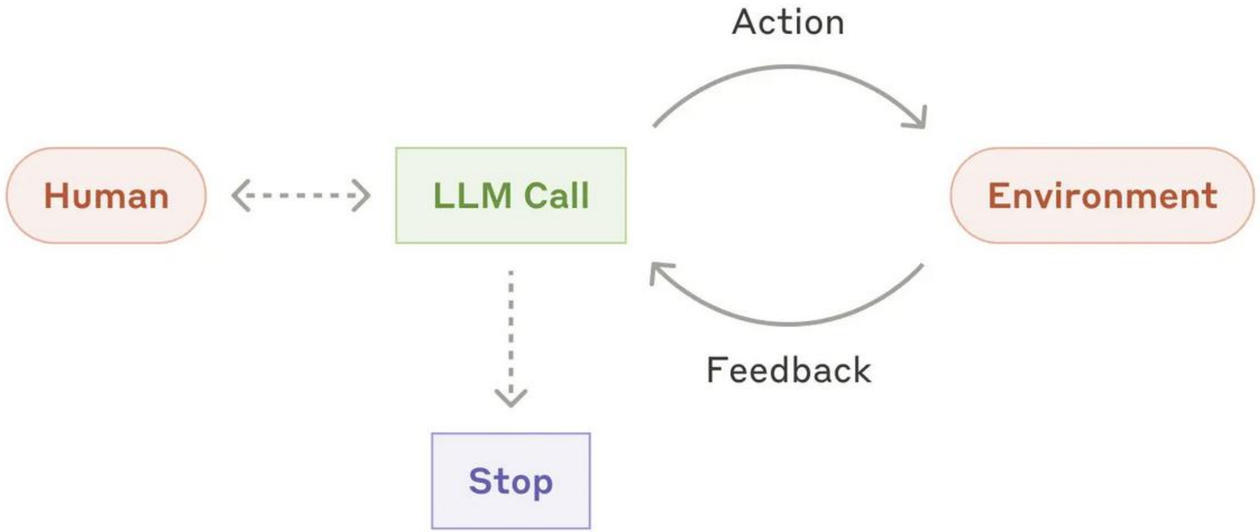






General Agents

- Start with a command or interactive discussion
- Once task is clear: Plan and operate independently
- Gather ground truth, add human feedback
- **“LLMs using tools based on environmental feedback in a loop”**



When to use agents: open-ended problems where it’s difficult or impossible to predict the required number of steps, and where you can’t hardcode a fixed path

2. What Makes IBM i Special for Agents

(The why)

Three Core reasons

1. Agents augment your best people

   Example: Your senior admin's security audit methodology becomes an agent's instructions. That audit now runs across every system, every day, at the same standard of quality.

2. IBM i systems deserve 24/7 intelligent attention

   Example: Performance degradation detected and analyzed at 2 AM, with admin receiving actionable insights when they arrive

3. The interface to your system is already agent-ready

   Example: "Why did Order 12345 fail?" gets investigated across jobs, logs, and database in seconds vs. hours

5 Requirements of an Enterprise AI OS

1. Governed Data Access	Access business data without crossing system boundaries
2. Object-Level Enforcement	Enforce what an agent can do at the object level
3. Intrinsic Audit Trails	Every action traceable to a decision, identity, and policy
4. Hardware Abstraction	Absorb new hardware without rewriting core applications
5. Bounded Autonomy	Agents operate within autonomy the system enforces

IBM i answers YES to all five -- by design.

Other “systems”

- Multiple distributed services and connections
- Monitoring tools, DB, security model all separate
- Each individual service has its own API, SDKs, usage guidelines

IBM i

- Single Interface: SQL
- Security and access control built in
- Audit trails built in

Less complexity = more reliable agents = faster development

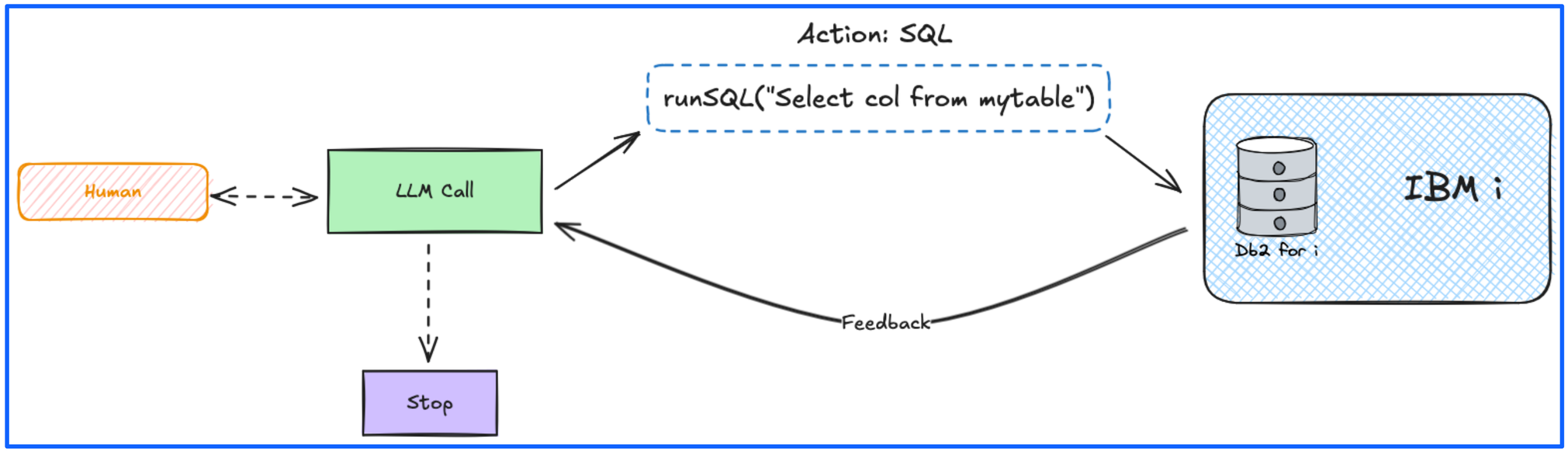
IBM i: The first agentic-native operating system

IBM i's greatest strength for the AI age may be architectural decisions made decades before the current AI revolution

Three Pillars

1. Integrated architecture by design (OS+DB)
2. Command Driven environment
3. Security and Role-based access

The universal agentic interface: SQL



Key Takeaway: SQL is the universal interface

IBM® Db2® for i Services

Health Center Procedures

QSYS2.HEALTH_ACTIVITY
QSYS2.HEALTH_DATABASE_OVERVIEW
QSYS2.HEALTH_DESIGN_LIMITS
QSYS2.HEALTH_ENVIRONMENTAL_LIMITS
QSYS2.HEALTH_SIZE_LIMITS
QSYS2.RESET_ENVIRONMENTAL_LIMITS

Performance Services

QSYS2.ACTIVE_QUERY_INFO – UDTF
QSYS2.ADD_QUERY_THRESHOLD – PROCEDURE
QSYS2.CONDENSEDINDEXADVICE – VIEW
QSYS2.DATABASE_MONITOR_INFO – VIEW
QSYS2.MTI_INFO – UDTF
QSYS2.QUERY_SUPERVISOR – VIEW
QSYS2.REMOVE_QUERY_THRESHOLD – PROCEDURE
QSYS2.RESET_TABLE_INDEX_STATISTICS – PROCEDURE
QSYS2.SYSIXADV – TABLE
SYSIBMADM.QIBM_SYSIXADV_BY_DAYS – GLOBAL VARIABLE
SYSTOOLS.ACT_ON_INDEX_ADVICE – PROCEDURE
SYSTOOLS.HARVEST_INDEX_ADVICE – PROCEDURE
SYSTOOLS.REMOVE_INDEXES – PROCEDURE

Plan Cache Procedures

QSYS2.CHANGE_PLAN_CACHE_SIZE
QSYS2.CLEAR_PLAN_CACHE
QSYS2.DUMP_PLAN_CACHE
QSYS2.DUMP_PLAN_CACHE_PROPERTIES
QSYS2.DUMP_PLAN_CACHE_TOPN
QSYS2.DUMP_SNAP_SHOT_PROPERTIES
QSYS2.END_ALL_PLAN_CACHE_EVENT_MONITORS
QSYS2.END_PLAN_CACHE_EVENT_MONITOR
QSYS2.IMPORT_PC_EVENT_MONITOR
QSYS2.IMPORT_PC_SNAPSHOT
QSYS2.REMOVE_PC_EVENT_MONITOR
QSYS2.REMOVE_PC_SNAPSHOT
QSYS2.REMOVE_PERFORMANCE_MONITOR
QSYS2.START_PLAN_CACHE_EVENT_MONITOR

IBM® Db2® for i Services

Utility Services

QSYS2.ANALYZE_CATALOG – UDTF
QSYS2.CANCEL_SQL – PROCEDURE
QSYS2.COMPARE_FILE – UDTF
QSYS2.DUMP_SQL_CURSORS – PROCEDURE
QSYS2.END_IDLE_SQE_THREADS – PROCEDURE
QSYS2.EXTRACT_STATEMENTS – PROCEDURE
QSYS2.FIND_AND_CANCEL_QSQSRVR_SQL – PROCEDURE
QSYS2.FIND_QSQSRVR_JOBS – PROCEDURE
QSYS2.GENERATE_SQL – PROCEDURE
QSYS2.GENERATE_SQL_OBJECTS – PROCEDURE
QSYS2.RESTART_IDENTITY – PROCEDURE
QSYS2.SWAP_DYNUSRPRF – PROCEDURE
SYSTOOLS.CHECK_SYSCST – PROCEDURE
SYSTOOLS.CHECK_SYSROUTINE – PROCEDURE
SYSTOOLS.RELATED_OBJECTS – UDTF
SYSTOOLS.VALIDATE_DATA – UDTF

Application Services

QSYS2.DELIMIT_NAME – UDF
QSYS2.OVERRIDE_QAQQINI – PROCEDURE
QSYS2.OVERRIDE_TABLE – PROCEDURE
QSYS2.PARSE_STATEMENT – UDTF
QSYS2.SQLSTATE_INFO – TABLE
QSYS2.SQL_ERRORT – TABLE
QSYS2.SQL_ERROR_LOG – VIEW
SYSIBMADM.QIBM_SELF_BY_DAYS – GLOBAL VARIABLE
SYSIBMADM.SELFCODES – GLOBAL VARIABLE
SYSIBMADM.VALIDATE_SELF – UDF
SYSPROC.WLM_SET_CLIENT_INFO – PROCEDURE
SYSTOOLS.SQLCODE_INFO – UDTF

IBM® i Services

Spool Services

QSYS2.OUTPUT_QUEUE_ENTRIES – UDTF & VIEW
QSYS2.OUTPUT_QUEUE_ENTRIES_BASIC – VIEW
QSYS2.OUTPUT_QUEUE_INFO – VIEW
QSYS2.SPOOLED_FILE_INFO – UDTF
SYSTOOLS.DELETE_OLD_SPOOLED_FILES – PROCEDURE
SYSTOOLS.GENERATE_PDF – UDF
SYSTOOLS.PRINTER_FILE_INFO – VIEW
SYSTOOLS.SPOOLED_FILE_DATA – UDTF

Journal Services

QSYS2.ASSOCIATE_JOURNAL_RECEIVER – UDTF
QSYS2.AUDIT_JOURNAL_DATA_MART_INFO – VIEW
QSYS2.DISPLAY_JOURNAL – UDTF
QSYS2.JOURNAL_INFO – VIEW
QSYS2.JOURNAL_CODE_INFO – VIEW
QSYS2.JOURNAL_RECEIVER_INFO – VIEW
QSYS2.JOURNALED_OBJECTS – VIEW
QSYS2.MANAGE_AUDIT_JOURNAL_DATA_MART – PROCEDURE
QSYS2.REMOTE_JOURNAL_INFO – VIEW
QSYS2.SMAPP_ACCESS_PATHS – VIEW
SYSTOOLS.AUDIT_JOURNAL_nn – UDTFs
SYSTOOLS.DELETE_OLD_JOURNAL_RECEIVERS – PROCEDURE

Message Handling Services

QSYS2.HISTORY_LOG_INFO – UDTF
QSYS2.JOBLOG_INFO – UDTF
QSYS2.MESSAGE_FILE_DATA – VIEW
QSYS2.MESSAGE_QUEUE_INFO – UDTF & VIEW
QSYS2.REPLY_LIST_INFO – VIEW
QSYS2.SEND_MESSAGE – PROCEDURE
SYSTOOLS.REPLY_INQUIRY_MESSAGES – UDF

Problem Management Services

SYSTOOLS.PROBLEM_INFO – VIEW

Backup and Recovery Services

QSYS2.MEDIA_LIBRARY_INFO – VIEW
QSYS2.SAVE_FILE_INFO – VIEW
QSYS2.SAVE_FILE_OBJECTS – UDTF & VIEW
QSYS2.TAPE_CARTRIDGE_INFO – VIEW

IBM® i Services

Security Services

QSYS2.AUTHORITY_COLLECTION – VIEW
QSYS2.AUTHORITY_COLLECTION_DLO – VIEW
QSYS2.AUTHORITY_COLLECTION_FSOBJ – VIEW
QSYS2.AUTHORITY_COLLECTION_IFS – VIEW
QSYS2.AUTHORITY_COLLECTION_LIBRARIES – VIEW
QSYS2.AUTHORITY_COLLECTION_OBJECT – VIEW
QSYS2.AUTHORIZATION_LIST_INFO – VIEW
QSYS2.AUTHORIZATION_LIST_USER_INFO – VIEW
QSYS2.CERTIFICATE_INFO – UDTF
QSYS2.CERTIFICATE_USAGE_INFO – VIEW
QSYS2.CHANGE_TOTP_KEY – UDTF
QSYS2.CHECK_PASSWORD – UDTF
QSYS2.CHECK_TOTP – UDF
QSYS2.DRDA_AUTHENTICATION_ENTRY_INFO – VIEW
QSYS2.FUNCTION_INFO – VIEW
QSYS2.FUNCTION_USAGE – VIEW
QSYS2.GROUP_PROFILE_ENTRIES – VIEW
QSYS2.KERBEROS_KEYTAB_ENTRIES – UDTF
QSYS2.OBJECT_OWNERSHIP – VIEW
QSYS2.OBJECT_PRIVILEGES – UDTF & VIEW
QSYS2.SECURITY_INFO – VIEW
QSYS2.SQL_CHECK_AUTHORITY – UDF
QSYS2.SQL_CHECK_FUNCTION_USAGE – UDF
QSYS2.SQL_CHECK_SPECIAL_AUTHORITY – UDF
QSYS2.USER_INFO – VIEW
QSYS2.USER_INFO_BASIC – VIEW

Security Services (continued)

SYSPROC.SET_COLUMN_ATTRIBUTE – PROCEDURE
SYSTOOLS.ADD_VALIDATION_LIST_ENTRY – UDF
SYSTOOLS.CHANGE_USER_PROFILE – UDTF
SYSTOOLS.CHANGE_VALIDATION_LIST_ENTRY – UDF
SYSTOOLS.REMOVE_VALIDATION_LIST_ENTRY – UDF
SYSTOOLS.SPECIAL_AUTHORITY_DATA_MART – TABLE
SYSTOOLS. USER_DRDA_AUTHENTICATION_ENTRIES – UDTF

PTF Services

QSYS2.ELECTRONIC_SERVICE_AGENT_INFO – VIEW
QSYS2.GROUP_PTF_INFO – VIEW
QSYS2.PTF_INFO – VIEW
SYSTOOLS.DEFECTIVE_PTF_CURRENCY – VIEW
SYSTOOLS.FIRMWARE_CURRENCY – VIEW
SYSTOOLS.GROUP_PTF_CURRENCY – VIEW
SYSTOOLS.GROUP_PTF_DETAILS – VIEW
SYSTOOLS.PTF_COVER_LETTER – UDTF

IBM® i Services

Application Services

QSYS2.ACTIVATION_GROUP_INFO – UDTF
QSYS2.ADD_USER_INDEX_ENTRY – PROCEDURE
QSYS2.BINDING_DIRECTORY_INFO – VIEW
QSYS2.BOUND_MODULE_INFO – VIEW
QSYS2.BOUND_SRVPGM_INFO – VIEW
QSYS2.CHANGE_USER_SPACE – PROCEDURE
QSYS2.CHANGE_USER_SPACE_ATTRIBUTES – PROCEDURE
QSYS2.CLEAR_DATA_QUEUE – PROCEDURE
QSYS2.COMMAND_INFO – VIEW
QSYS2.CREATE_USER_INDEX – PROCEDURE
QSYS2.CREATE_USER_SPACE – PROCEDURE
QSYS2.DATA_AREA_INFO – UDTF & VIEW
QSYS2.DATA_QUEUE_ENTRIES – UDTF
QSYS2.DATA_QUEUE_INFO – VIEW
QSYS2.DB_TRANSACTION_INFO – VIEW
QSYS2.DB_TRANSACTION_JOURNAL_INFO – UDTF
QSYS2.DB_TRANSACTION_OBJECT_INFO – UDTF
QSYS2.DB_TRANSACTION_RECORD_INFO – UDTF
QSYS2.ENVIRONMENT_VARIABLE_INFO – VIEW
QSYS2.EXIT_POINT_INFO – VIEW
QSYS2.EXIT_PROGRAM_INFO – VIEW
QSYS2.PROGRAM_EXPORT_IMPORT_INFO – VIEW
QSYS2.PROGRAM_INFO – VIEW
QSYS2.PROGRAM_RESOLVED_ACTIVATIONS – UDTF
QSYS2.PROGRAM_RESOLVED_IMPORTS – UDTF

Application Services (continued)

QSYS2.QCMDEXC – PROCEDURE & UDF
QSYS2.RECEIVE_DATA_QUEUE – UDTF
QSYS2.REMOVE_USER_INDEX_ENTRY – UDTF
QSYS2.SEND_DATA_QUEUE – PROCEDURE
QSYS2.SERVICES_INFO – TABLE
QSYS2.SET_PASE_SHELL_INFO – PROCEDURE
QSYS2.STACK_INFO – UDTF
QSYS2.USER_INDEX_ENTRIES – UDTF
QSYS2.USER_INDEX_INFO – VIEW
QSYS2.USER_SPACE – UDTF
QSYS2.USER_SPACE_INFO – VIEW
QSYS2.VERIFY_NAME – UDF
QSYS2.WATCH_DETAIL – UDTF
QSYS2.WATCH_INFO – VIEW
SYSTOOLS.ERRNO_INFO – UDF
SYSTOOLS.EVEN – UDF
SYSTOOLS.GENERATE_SPREADSHEET – UDF
SYSTOOLS.GETENV – UDF
SYSTOOLS.LPRINTF – PROCEDURE
SYSTOOLS.ODD – UDF
SYSTOOLS.OVERRIDE_INFO – VIEW
SYSTOOLS.PUTENV – UDF
SYSTOOLS.SEND_EMAIL – UDF
SYSTOOLS.SPLIT – UDTF

IBM® i Services

Work Management Services

QSYS2.ACTIVE_JOB_INFO – UDTF
QSYS2.ADD_TRACKED_JOB_QUEUE – PROCEDURE
QSYS2.AUTOSTART_JOB_INFO – VIEW
QSYS2.CLEAR_TRACKED_JOB_INFO – PROCEDURE
QSYS2.COMMUNICATIONS_ENTRY_INFO – VIEW
QSYS2.GET_JOB_INFO – UDTF
QSYS2.JOB_DESCRIPTION_INFO – VIEW
QSYS2.JOB_INFO – VIEW
QSYS2.JOB_LOCK_INFO – UDTF
QSYS2.JOB_QUEUE_INFO – VIEW
QSYS2.MEMORY_POOL – UDTF
QSYS2.MEMORY_POOL_INFO – VIEW
QSYS2.OBJECT_LOCK_INFO – VIEW
QSYS2.OPEN_FILES – UDTF
QSYS2.PRESTART_JOB_INFO – VIEW
QSYS2.PRESTART_JOB_STATISTICS – UDTF
QSYS2.RECORD_LOCK_INFO – VIEW
QSYS2.REMOVE_TRACKED_JOB_QUEUE – UDTF
QSYS2.ROUTING_ENTRY_INFO – VIEW
QSYS2.SCHEDULED_JOB_INFO – VIEW
QSYS2.SUBSYSTEM_INFO – VIEW
QSYS2.SUBSYSTEM_POOL_INFO – VIEW
QSYS2.SYSTEM_ACTIVITY_INFO – UDTF
QSYS2.SYSTEM_STATUS – UDTF
QSYS2.SYSTEM_STATUS_INFO – VIEW
QSYS2.SYSTEM_STATUS_INFO_BASIC – VIEW

Work Management Services (continued)

QSYS2.SYSTEM_VALUE_INFO – VIEW
QSYS2.TRACKED_JOB_INFO – UDTF
QSYS2.TRACKED_JOB_QUEUES – VIEW
QSYS2.WORKLOAD_GROUP_INFO – VIEW
QSYS2.WORKSTATION_INFO – VIEW
SYSTOOLS.END_JOBS – PROCEDURE
SYSTOOLS.ENDED_JOB_INFO – UDTF
SYSTOOLS.JOB_QUEUE_ENTRIES – VIEW
SYSTOOLS.POWER_SCHEDULE_INFO – VIEW

IBM® i Services

Product Services

QSYS2.LICENSE_INFO – VIEW
QSYS2.SOFTWARE_PRODUCT_INFO – VIEW
SYSTOOLS.CHECK_PRODUCT_OPTIONS – UDTF
SYSTOOLS.LICENSE_EXPIRATION_CHECK – PROCEDURE

Storage Services

QSYS2.ADD_DEVICE_LOCKING_POLICY – PROCEDURE
QSYS2.ASP_INFO – VIEW
QSYS2.ASP_JOB_INFO – VIEW
QSYS2.ASP_VARY_INFO – VIEW
QSYS2.CHANGE_DEVICE_LOCKING_POLICY – PROCEDURE
QSYS2.CHANGE_DISK_PATHS – PROCEDURE
QSYS2.CREATE_LOCKING_POLICY – PROCEDURE
QSYS2.DELETE_LOCKING_POLICY – PROCEDURE
QSYS2.FACTORY_RESET_DEVICE – PROCEDURE
QSYS2.LOCKING_POLICY_INFO – VIEW
QSYS2.NVME_INFO – VIEW
QSYS2.REMOVE_DEVICE_LOCKING_POLICY – PROCEDURE
QSYS2.SYSDISKSTAT – UDTF & VIEW
QSYS2.SYSTMPSTG – VIEW
QSYS2.UNLOCK_DEVICE – PROCEDURE
QSYS2.USER_STORAGE – VIEW

Performance Services

QSYS2.COLLECTION_SERVICES_INFO – VIEW

System Health Services

QSYS2. PROCESS_SYSTEM_LIMITS_ALERTS – PROCEDURE
QSYS2.SYSLIMITS – VIEW
QSYS2.SYSLIMITS_BASIC – VIEW
QSYS2.SYSLIMTBL – TABLE

Configuration Services

QSYS2.ADD_ISCSI_TARGET – PROCEDURE
QSYS2.CHANGE_IOP – PROCEDURE
QSYS2.CHANGE_ISCSI_TARGET – PROCEDURE
QSYS2.CHANGE_SERVICE_TOOLS_SERVER – PROCEDURE
QSYS2.HARDWARE_RESOURCE_INFO – UDTF & VIEW
QSYS2.ISCSI_INFO – VIEW
QSYS2.REMOVE_ISCSI_TARGET – PROCEDURE
QSYS2.SERVICE_TOOLS_SERVER_INFO – VIEW
QSYS2.SFP_TRANSCEIVER_INFO – VIEW
SYSTOOLS.CONFIGURATION_STATUS – VIEW

IBM® i Services

IFS Services

QSYS2.COMPARE_IFS – UDTF
QSYS2.IFS_JOB_INFO – UDTF
QSYS2.IFS_OBJECT_LOCK_INFO – UDTF
QSYS2.IFS_OBJECT_PRIVILEGES – UDTF
QSYS2.IFS_OBJECT_REFERENCES_INFO – UDTF
QSYS2.IFS_OBJECT_STATISTICS – UDTF
QSYS2.IFS_READ – UDTF
QSYS2.IFS_WRITE – PROCEDURE
QSYS2.SERVER_SHARE_INFO – VIEW
SYSTOOLS.IFS_ACCESS – UDF
SYSTOOLS.IFS_PATH – UDF
SYSTOOLS.IFS_RENAME – UDF
SYSTOOLS.IFS_UNLINK – UDF

Java Services

QSYS2.JVM_INFO – UDTF & VIEW
QSYS2.SET_JVM – PROCEDURE

Librarian Services

QSYS2.JOURNAL_INHERIT_RULES – VIEW
QSYS2.LIBRARY_INFO – UDTF
QSYS2.LIBRARY_LIST_INFO – VIEW
QSYS2.OBJECT_STATISTICS – UDTF
QSYS2.SYSTEM_OBJECT_TYPES – TABLE

Communication Services

QSYS2.DNS_LOOKUP_IP – UDF
QSYS2.NETSTAT_JOB_INFO – VIEW
QSYS2.NETSTAT_ROUTE_INFO – VIEW
QSYS2.NETWORK_ATTRIBUTE_INFO – VIEW
QSYS2.RDB_ENTRY_INFO – VIEW
QSYS2.SERVER_SBS_ROUTING – VIEW
QSYS2.SET_SERVER_SBS_ROUTING – PROCEDURE
QSYS2.TELNET_SERVER_ATTRIBUTES – VIEW
QSYS2.TIME_PROTOCOL_INFO – VIEW
SYSIBMADM.ENV_SYS_INFO – VIEW
SYSTOOLS.PING – UDTF

3. Building Blocks & Practical Patterns

(The how)

Anatomy of an Agent

Core Components:

1. Tools
2. Memory
3. Skills/instructions
4. Guardrails

- Keep agents ”atomic”
- Focus tools and capabilities*
- Use SQL services as a guideline
- MCP server as the connector

```python
# ============================================================================
# Tools
# ============================================================================

tools = [
    MCPTools(
        url=MCP_URL,
        transport="streamable-http",
        timeout_seconds=30,
        include_tools=get_toolsets("ifs_operations", "service_discovery") + SQL_TOOLS,
        requires_confirmation_tools=SQL_CONFIRMATION_TOOLS,
    ),
    exa_tools(),
]

# ============================================================================
# Agent Instance
# ============================================================================

ifs_agent = Agent(
    id=AGENT_ID,
    name=NAME,
    model=get_model(model_id=DEFAULT_MODEL_ID),
    description=DESCRIPTION,
    instructions=INSTRUCTIONS,
    tools=tools,
    db=PostgresDb(id="agno-storage", db_url=db_url),
    learning=get_agent_learning("nav"),
    **_AGENT_DEFAULTS,
)
```

Considerations

1. Security & Guardrails
    Agents need clear boundaries on what they can/can't do
    IBM i strength: Built-in role-based access enforced automatically
    Consideration: Define agent permissions as carefully as human permissions

2. Trust & Transparency
    Teams need to understand what agents are doing
    Best practice: Always show agent reasoning and actions taken
    Start: Read-only analysis agents before action-taking agents

3. Learning Curve
    Shift from "how do I use this tool" to "how do I describe my goal"
    Reality: Natural language is easy, but precise prompting is learned skill
    Approach: Start with well-defined use cases, expand as team gains confidence

4. Integration with Existing Workflows
    Agents augment humans, not replace overnight
    Strategy: Begin with time-consuming investigative tasks
    Avoid: Don't force agents where deterministic workflows work well

Framework for building IBM i agents: 3 easy steps (*without MCP)

Step 1: What problem are you trying to solve?

Step 2: What IBM i data do you have access to?

Step 3: How do you want your agent to behave or interact with your environment (scope of actions)

Step 1: What Problem are we solving?

Imagine: You are a system admin who need to check system performance throughout the day.

Instead: of checking dashboards or running queries manually, you want to simply ask: *“What’s the current system performance?”*

Our Goal: Build an agent that can retrieve and analyze system performance data

Step 2: What Data do we have to make tools?

The universal agentic interface: SQL

- The true brilliance of the IBM i architecture: All system information is available via SQL
- For performance monitoring:

```
SELECT * FROM TABLE(QSYS2.SYSTEM_STATUS(RESET_STATISTICS=>'YES',DETAILED_INFO=>'ALL')) X

SELECT * FROM TABLE(QSYS2.SYSTEM_ACTIVITY_INFO())
```

The “bridge”

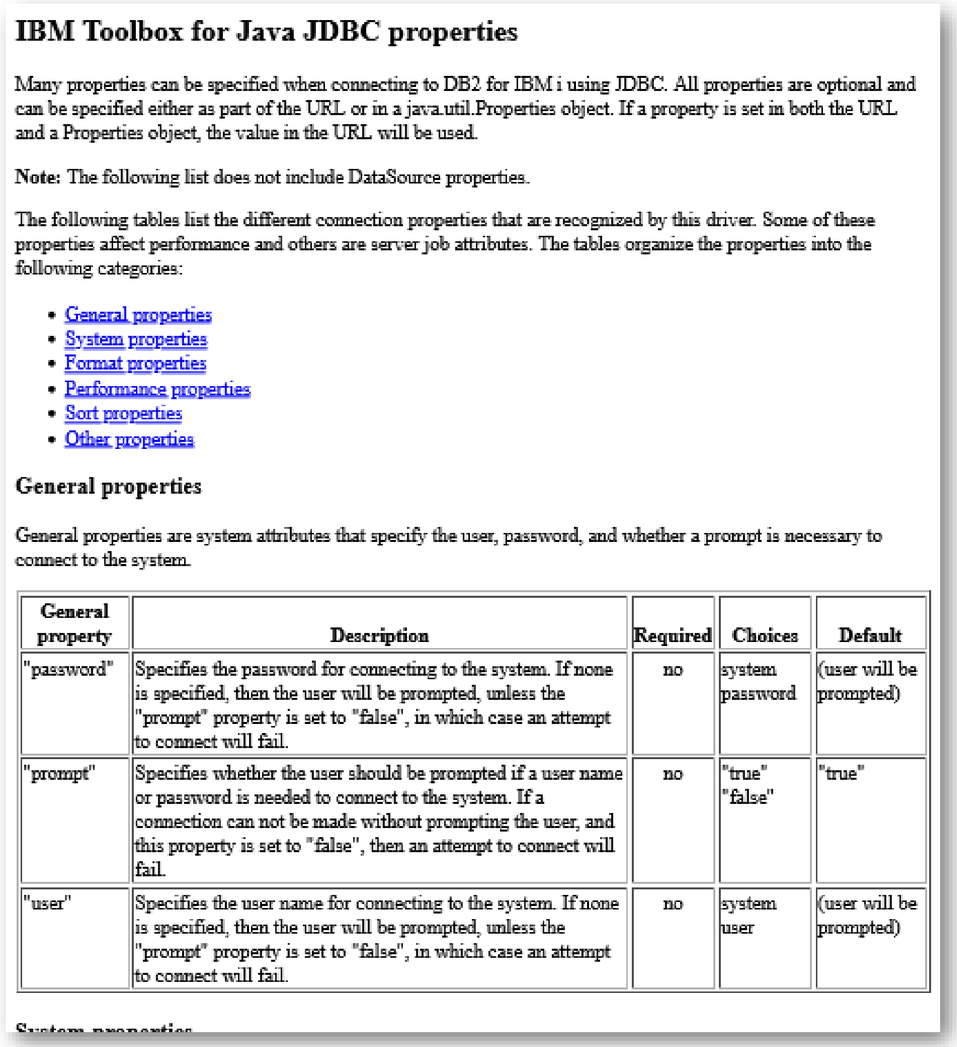
- To allow our agent to run SQL, we need a secure “client” method for executing SQL statements
- Fully encrypted communication
- This single function handles connection management and returns the result set as a string

```python
def run_sql_statement(
    sql: str,
    parameters: Optional[QueryParameters] = None,
    creds: Optional[Dict[str, Any]] = None,
) -> str:
    """
    Execute SQL statement on IBM i and return formatted results.

    Args:
        sql: SQL statement to execute
        parameters: Optional parameters for prepared statements
        creds: Database connection credentials (defaults to environment credentials)

    Returns:
        Formatted string with SQL results or error message
    """
    if creds is None:
        creds = get_ibmi_credentials()

    with connect(creds) as conn:
        with conn.execute(sql, parameters=parameters) as cur:
            if cur.has_results:
                result = cur.fetchall()
                return str(result["data"])
            else:
                return "SQL executed successfully. No results returned."
```

https://mapepire-ibmi.github.io/

The “bridge”

- Supports all the common JDBC properties
  - Of note: “access=read only”
- Can be governed with all the classic techniques
  - SET_SERVER_SBS_ROUTING
    - Memory pools
    - Workload capping groups
  - Exit point user governance



IBM Toolbox for Java JDBC properties

Many properties can be specified when connecting to DB2 for IBM i using JDBC. All properties are optional and can be specified either as part of the URL or in a java.util.Properties object. If a property is set in both the URL and a Properties object, the value in the URL will be used.

Note: The following list does not include DataSource properties.

The following tables list the different connection properties that are recognized by this driver. Some of these properties affect performance and others are server job attributes. The tables organize the properties into the following categories:

- General properties
- System properties
- Format properties
- Performance properties
- Sort properties
- Other properties

General properties

General properties are system attributes that specify the user, password, and whether a prompt is necessary to connect to the system.

General property	Description	Required	Choices	Default
"password"	Specifies the password for connecting to the system. If none is specified, then the user will be prompted, unless the "prompt" property is set to "false", in which case an attempt to connect will fail.	no	system password	(user will be prompted)
"prompt"	Specifies whether the user should be prompted if a user name or password is needed to connect to the system. If a connection can not be made without prompting the user, and this property is set to "false", then an attempt to connect will fail.	no	"true" "false"	"true"
"user"	Specifies the user name for connecting to the system. If none is specified, then the user will be prompted, unless the "prompt" property is set to "false", in which case an attempt to connect will fail.	no	system user	(user will be prompted)

System properties

https://mapepire-ibmi.github.io/

Converting SQL into agent tools

Key Concept: wrap each SQL statement in a “tool” that the agent can discover

Tools: functions with descriptions attached

- the agent reads these descriptions to decide when to use each tool

system_status

```
system_status = "SELECT * FROM
TABLE(QSYS2.SYSTEM_STATUS(RESET_STATISTICS=>'YES',DETAILED_INFO=>'ALL')) X"

@tool(
    name="Get System Status",
    description="Retrieve overall system performance statistics",
)
def get_system_status() -> str:
    """Retrieve overall system performance statistics"""
    return run_sql_statement(system_status)
```

system_activity

```
system_activity = "SELECT * FROM TABLE(QSYS2.SYSTEM_ACTIVITY_INFO())"

@tool(
    name="Get System Activity",
    description="Retrieve current system activity information",
)
def get_system_activity() -> str:
    """Retrieve current system activity information"""
    return run_sql_statement(system_activity)
```

Step 3: How should our agent behave?

Goal: Define the agent itself

- ➢ Define the design-making layer
- ➢ Define guardrails and custom instructions
- ➢ Pick an Agent Framework

Agent Breakdown:

- ❖ Framework: Agno
- ❖ Model: Claude sonnet 4
- ❖ Instructions: agent's behavior guide. It tells the agent its role, how to act, and what approach to take

Performance Metrics Assistant

```
from agno.agent import Agent
from agno.models.anthropic import Claude

agent = Agent(
    name="Performance Metrics Assistant",
    model=Claude(id="claude-sonnet-4-20250514"),
    tools=[
        get_system_status,
        get_system_activity,
    ],
    instructions=dedent("""
        You are an expert IBM i performance metrics assistant. Your role is to help users
        retrieve and analyze performance-related data from the IBM i system using SQL queries.

        When asked about system performance:
        - Use the available tools to gather current data
        - Explain the metrics in business-friendly terms
        - Highlight any concerning values
    """),
    markdown=True,
)
```

uv

```
(base) ~/Documents/IBM/sandbox/oss/ai/db2i-ai/db2i-agents/examples/agents-infra [main]
$ uv run cli.py --stream --agent metrics --model-id anthropic:claude-haiku-4-5

================================================================================
Starting: Performance Metrics Assistant
Model: anthropic:claude-haiku-4-5
Storage: SQLite (tmp/agents.db)
Debug: False
Stream: True
================================================================================

😎 User :
```

uv

```
(base) ~/Documents/IBM/sandbox/oss/ai/db2i-ai/db2i-agents/examples/agents-infra [main]
$ uv run cli.py --stream  --agent metrics

================================================================================
Starting: Performance Metrics Assistant
Model: openai:gpt-4o
Storage: SQLite (tmp/agents.db)
Debug: False
Stream: True
================================================================================

😎 User :
```

“Core Business Data” Agent

1. Working with Core business data can be complex and challenging, especially for new developers
2. I have access to the SAMPLE database with employee and department tables
3. I want to create comprehensive salary reports using natural language

*Instead of using Agno, lets use the **prompt chaining pattern** to build an agent

Step 1: What problem are you trying to solve?

Step 2: What IBM i data do you have access to?

Step 3: How do you want your agent to behave or interact with your environment (scope of actions)

Core Business Data agent: SQL tools

- **Schema:** SAMPLE
- **Tables:** Department, and Employee

Task: Build an AI agent that can safely query information from IBM i systems

```
SELECT
    e.EMPNO,
    e.FIRSTNME,
    e.LASTNAME,
    e.JOB,
    e.EDLEVEL,
    e.HIREDATE,
    e.SALARY,
    e.BONUS,
    e.COMM,
    d.DEPTNAME,
    d.LOCATION
FROM SAMPLE2.EMPLOYEE e
LEFT JOIN SAMPLE2.DEPARTMENT d ON e.WORKDEPT = d.DEPTNO
WHERE e.SALARY IS NOT NULL
ORDER BY e.SALARY DESC;
```

Mapepire AI Demo: Salary Insights Analyzer

This notebook demonstrates how to integrate **Mapepire** (IBM i database connector) with **AI** to analyze organizational data.

What This Demo Does

1. Connects to IBM i using Mapepire
2. Queries employee and department data
3. Uses Ollama (local AI) to analyze salary patterns and generate insights
4. Displays both raw data and AI-generated analysis

Prerequisites

- IBM i system with SAMPLE2 database tables (EMPLOYEE, DEPARTMENT)
- Ollama running locally with a model installed (e.g., `llama3.2`)
- Environment variables set: `IBMI_HOST`, `IBMI_USER`, `IBMI_PASSWORD`

Section 1: Setup and Imports

```python
import os
from dotenv import load_dotenv
import pandas as pd
from mapepire_python.client import SQLJob
from mapepire_python import connect
from mapepire_python.data_types import DaemonServer
import ollama

# Load environment variables from .env file (if present)
load_dotenv()

print("✓ All imports successful")
```

[ ]

Python

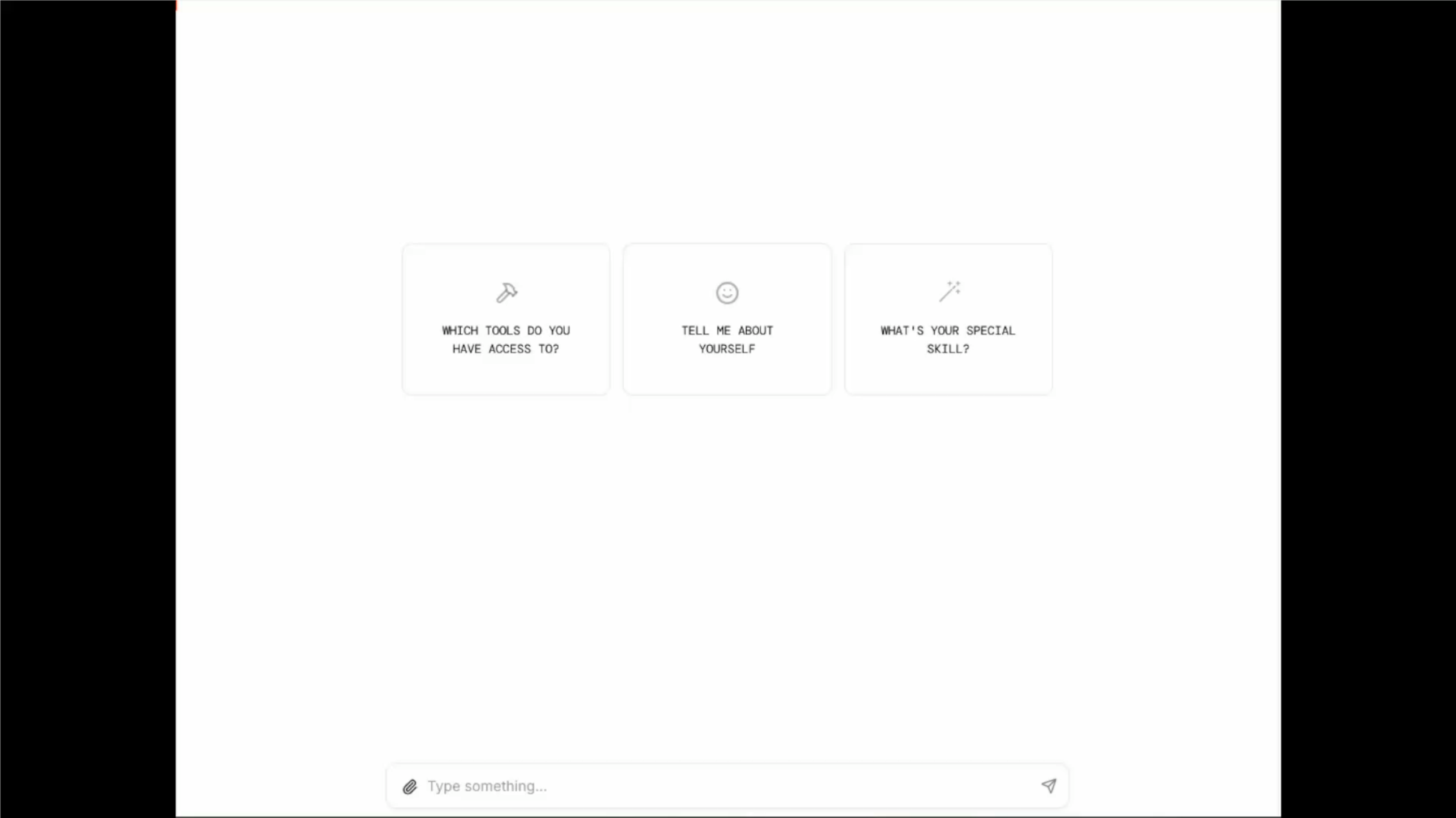
“Security” Agent

1. Protect the profiles on my system from impersonations attacks
2. I have access to `QSYS2.object_privaleges` for querying profile info
3. I want to find all exposed profiles on my system, then secure these profiles using a chat interface
   - **Best Practice:** To prevent an impersonation attack, user profiles should be configured with *PUBLIC set to *EXCLUDE

Step 1: What problem are you trying to solve?

Step 2: What IBM i data do you have access to?

Step 3: How do you want your agent to behave or interact with your environment (scope of actions)


WHICH TOOLS DO YOU HAVE ACCESS TO?
TELL ME ABOUT YOURSELF
WHAT'S YOUR SPECIAL SKILL?
Type something...

SQL Services = MCP Tools

Agentic Category: IBM i Security

Topic: IBM i User Profile (*USRPRF) attack vector(s)

Best Practice: To prevent an impersonation attack, user profiles should be configured with *PUBLIC set to *EXCLUDE

```sql
--
-- How many *USRPRF's do not have *PUBLIC set to *EXCLUDE?
--
SELECT COUNT(*) AS exposed_profile_count
FROM qsys2.object_privileges
WHERE system_object_schema = 'QSYS'
  AND object_type = '*USRPRF'
  AND object_name NOT IN ('QDBSHR', 'QDBSHRDO', 'QDOC', 'QTMPLPD')
  AND user_name = '*PUBLIC'
  AND object_authority <> '*EXCLUDE';
```

```sql
--
-- Which *USRPRF's do not have *PUBLIC set to *EXCLUDE?
--
SELECT object_name AS user_name,
       object_authority
FROM qsys2.object_privileges
WHERE system_object_schema = 'QSYS'
  AND object_type = '*USRPRF'
  AND object_name NOT IN ('QDBSHR', 'QDBSHRDO', 'QDOC', 'QTMPLPD')
  AND user_name = '*PUBLIC'
  AND object_authority <> '*EXCLUDE'
ORDER BY object_name;
```

```sql
--
-- Which *USRPRF's do not have *PUBLIC set to *EXCLUDE?
-- Include a query that corrects the exposure
--
SELECT object_name AS user_name,
       object_authority,
       'SELECT qsys2.qcmdexc(''GRTOBJAUT OBJ(QSYS/' || object_name ||
       ') OBJTYPE(*USRPRF) USER(*PUBLIC) AUT(*EXCLUDE)'') FROM sysibm.sysdummy1'
       AS corrective_query
FROM qsys2.object_privileges
WHERE system_object_schema = 'QSYS'
  AND object_type = '*USRPRF'
  AND object_name NOT IN ('QDBSHR', 'QDBSHRDO', 'QDOC', 'QTMPLPD')
  AND user_name = '*PUBLIC'
  AND object_authority <> '*EXCLUDE'
ORDER BY object_name;
```

Creating Tools for agent

Python function to run an SQL statement

Metadata about the Tool

Python function that runs the SQL to count the number of exposed profiles

```python
def run_sql_statement(
    sql: str,
    parameters: Optional[QueryParameters] = None,
    creds: Dict[str, Any] = credentials,
) -> str:
    """Execute SQL statement and return results"""
    with connect(creds) as conn:
        with conn.execute(sql, parameters=parameters) as cur:
            if cur.has_results:
                result = cur.fetchall()
                return str(result["data"])
            else:
                return "SQL executed successfully. No results returned."

@tool(
    name="count_exposed_profiles",
    description="Count how many user profiles don't have *PUBLIC set to *EXCLUDE",
    show_result=True,
    stop_after_tool_call=False,
)
def count_exposed_profiles() -> str:
    sql = """
        SELECT COUNT(*) AS exposed_profile_count
        FROM qsys2.object_privileges
        WHERE system_object_schema = 'QSYS'
            AND object_type = '*USRPRF'
            AND object_name NOT IN ('QDBSHR', 'QDBSHRDO', 'QDOC', 'QTMPLPD')
            AND user_name = '*PUBLIC'
            AND object_authority <> '*EXCLUDE'
    """

    result = run_sql_statement(sql)
    return result

# Create the simple agent
agent = Agent(
    model=OpenAIChat(),
    tools=[count_exposed_profiles],
    show_tool_calls=True,
    markdown=True,
)
```

Agentic AI with IBM i

Samples published here: https://github.com/ajshedivy/db2i-agents

📊 Agent Framework Comparison

Framework	Languages	Implementation Status	Db2i Access Method	Supports MCP	Description
🔗 LangChain	Python ✅	Complete ✅	Mapepire 🔌	Yes ✅	Popular framework for developing applications powered by LLMs
🏗️ MCP	Python ✅ TypeScript ✅	Complete ✅	Mapepire 🔌	Yes ✅	An open protocol that standardizes how applications provide context to LLMs.
🧩 Agno	Python ✅	Complete ✅	Mapepire 🔌	Yes ✅	A lightweight library for building Agents with memory, knowledge, tools and reasoning.
🐝 BeeAI	Python ✅ TypeScript ✅	Coming soon ⏳	Mapepire 🔌	Yes ✅	An open-source ecosystem that empowers developers to discover, run, and compose AI agents from any framework.
👥 CrewAI	Python ✅	Coming soon ⏳	Mapepire 🔌	No ❌	Fast and flexible Python Multi-Agent automation framework

Agent skills

Encapsulation of how to perform a task

Consists of:

- Instructions
- Metadata
- Scripts or other resources if needed

```
---
name: attack-vectors
description: >
  Procedures for investigating IBM i privilege escalation attacks. Load when:
  (1) investigating trigger attacks, rename attacks, or adopted authority
  exploitation, (2) correlating authority exposure with exploitable attack paths,
  (3) assessing compound attack chains that escalate individual findings.
---

# Attack Vectors & Privilege Escalation

Procedures for investigating how authority weaknesses chain into active
privilege escalation. These investigations answer: "How can an attacker
use these exposures to gain elevated access?"

## Quick Dispatch

| Vector | MCP Tool | Severity |
|--------|----------|----------|
| Trigger Attack | `list_db_files_exposed_to_trigger_attack` | CRITICAL |
| Rename Attack | `list_files_exposed_to_rename_attack` | CRITICAL |
| Adopted Authority | `list_adopted_authority_programs_with_public_access` | HIGH-CRITICAL |
| Command Authority | `list_public_authority_on_attack_vector_commands` | HIGH |

Load `attack-procedures.md` for full investigation steps per vector.

## Critical Gotchas

- NAMING(*SQL) defaults USRPRF(*OWNER) for BOTH static AND dynamic SQL — many
  programs adopt authority unknowingly. Always check DYNUSRPRF settings.
- Trigger programs execute under the adopted authority of the *accessing* program,
  NOT the trigger creator — this is the core escalation mechanism.
- Rename attack creates a view that looks identical to the original table. Even
  read-only access triggers malicious code in the WHERE clause.
- Heritage Navigator is known to be exposed to trigger attacks.

## Compound Risk Patterns

Watch for combinations that escalate severity. See `compound-risk-patterns.md`
for the 6 named chains (Identity Takeover, Library List + Trigger, Trigger +
Adopted Authority, Rename + View Substitution, SQL Injection + Command Execution,
Service Account Exposure).

When you detect a compound pattern, call it out explicitly and escalate severity.

```

```
---
name: attack-vectors
description: >
  Procedures for investigating IBM i privilege escalation attacks. Load when:
  (1) investigating trigger attacks, rename attacks, or adopted authority
  exploitation, (2) correlating authority exposure with exploitable attack paths,
  (3) assessing compound attack chains that escalate individual findings.
---

# Attack Vectors & Privilege Escalation

Procedures for investigating how authority weaknesses chain into active
privilege escalation. These investigations answer: "How can an attacker
use these exposures to gain elevated access?"

## Quick Dispatch

| Vector | MCP Tool | Severity |
|--------|----------|----------|
| Trigger Attack | `list_db_files_exposed_to_trigger_attack` | CRITICAL |
| Rename Attack | `list_files_exposed_to_rename_attack` | CRITICAL |
| Adopted Authority | `list_adopted_authority_programs_with_public_access` | HIGH-CRITICAL |
```

Quick Dispatch

Vector	MCP Tool	Severity
Trigger Attack	`list_db_files_exposed_to_trigger_attack`	CRITICAL
Rename Attack	`list_files_exposed_to_rename_attack`	CRITICAL
Adopted Authority	`list_adopted_authority_programs_with_public_access`	HIGH-CRITICAL
Command Authority	`list_public_authority_on_attack_vector_commands`	HIGH

Load `attack-procedures.md` for full investigation steps per vector.

Critical Gotchas

- NAMING(*SQL) defaults USRPRF(*OWNER) for BOTH static AND dynamic SQL — many programs adopt authority unknowingly. Always check DYNUSRPRF settings.
- Trigger programs execute under the adopted authority of the *accessing* program, NOT the trigger creator — this is the core escalation mechanism.
- Rename attack creates a view that looks identical to the original table. Even read-only access triggers malicious code in the WHERE clause.
- Heritage Navigator is known to be exposed to trigger attacks.

Compound Risk Patterns

- NAMING(*SQL) defaults USRPRF(*OWNER) for BOTH static AND dynamic SQL — many programs adopt authority unknowingly. Always check DYNUSRPRF settings.
- Trigger programs execute under the adopted authority of the *accessing* program, NOT the trigger creator — this is the core escalation mechanism.
- Rename attack creates a view that looks identical to the original table. Even read-only access triggers malicious code in the WHERE clause.
- Heritage Navigator is known to be exposed to trigger attacks.

Compound Risk Patterns

Watch for combinations that escalate severity. See `compound-risk-patterns.md` for the 6 named chains (Identity Takeover, Library List + Trigger, Trigger + Adopted Authority, Rename + View Substitution, SQL Injection + Command Execution, Service Account Exposure).

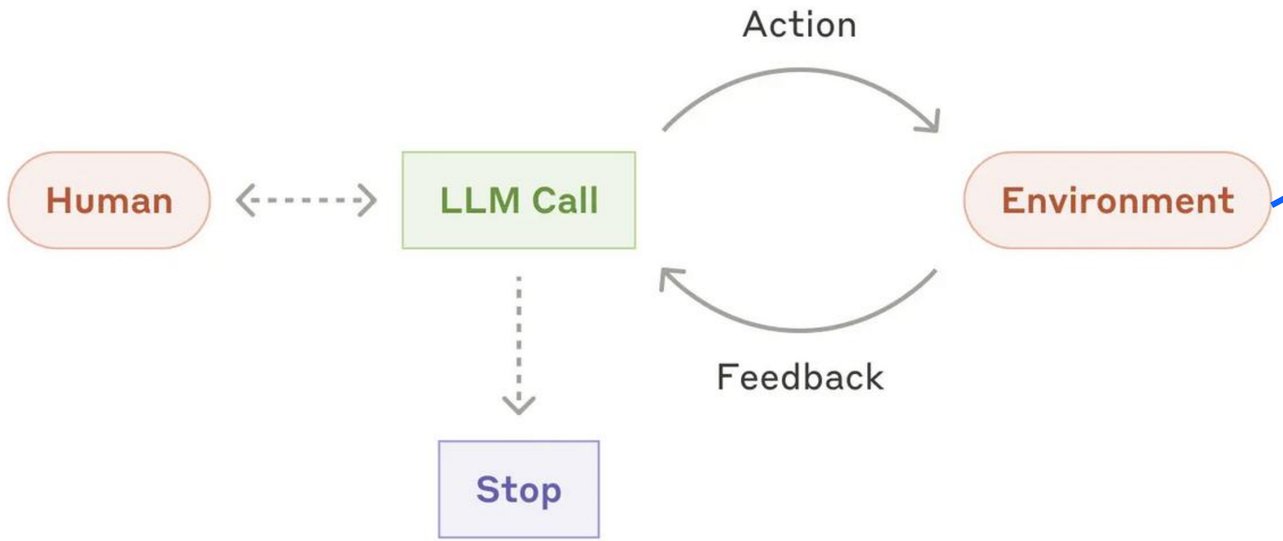
When you detect a compound pattern, call it out explicitly and escalate severity.

4. How MCP fits into the picture

What about MCP?

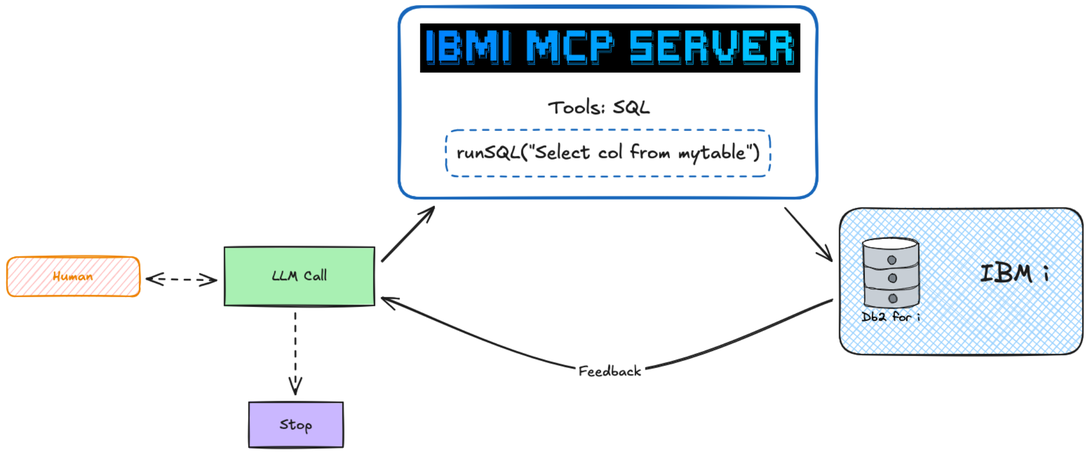
```
env = Environment()
tools = Tools(env)
system_prompt = "Goals, contraints, how to act"
task = "Peform task X, Y, Z"

while True:
  action = llm.run(task, system_prompt + env.state)
  env.state = tools.run(action)
```

MCP Server



Agent core component: Tools

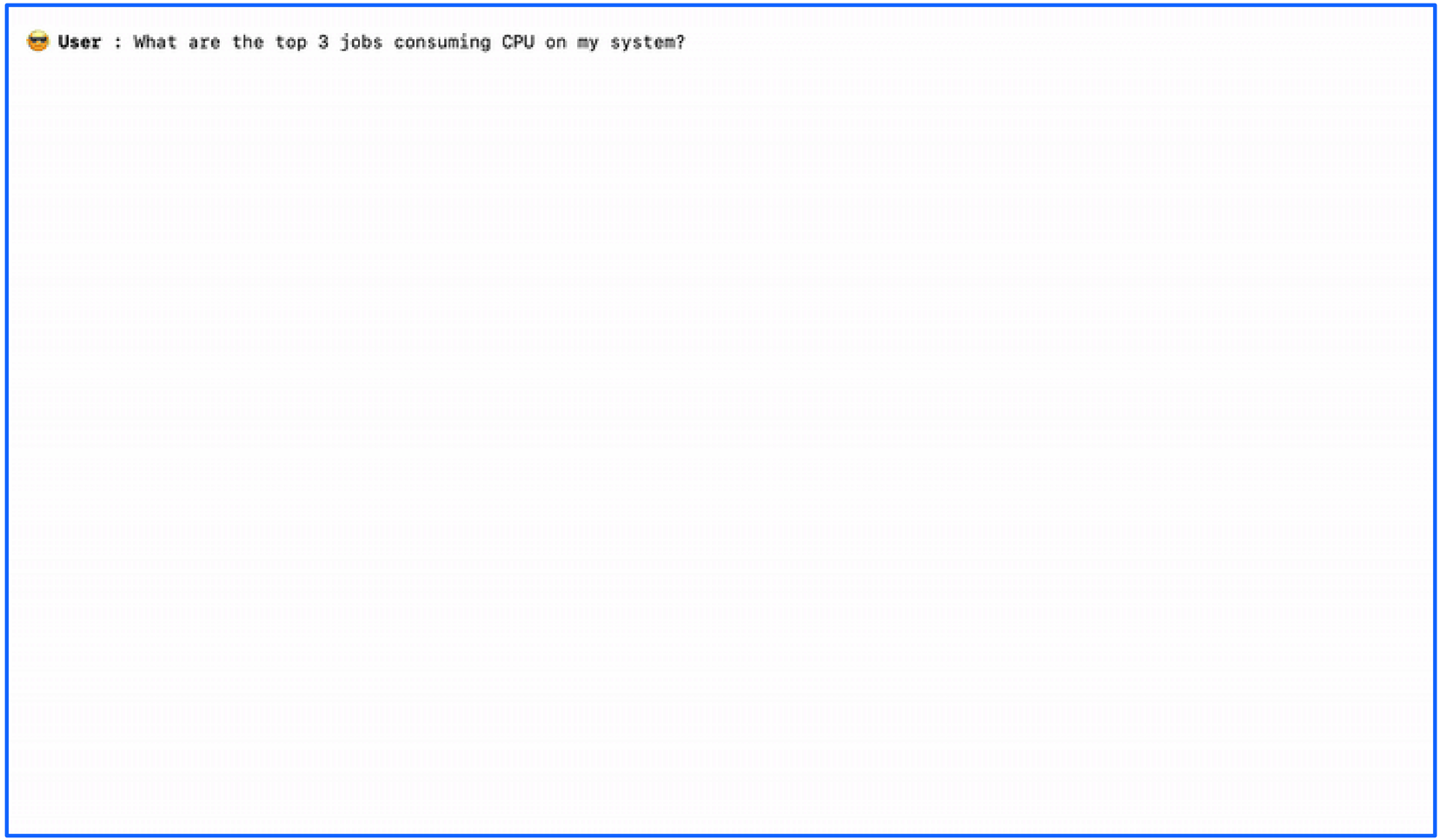
– Build tools for each SQL service category: Security, performance, developer tools, etc



sql-tools-config.json

```yaml
sources:
  ibmi-system:
    host: ${DB2i_HOST}
    user: ${DB2i_USER}
    password: ${DB2i_PASS}
    port: 8076
    ignore-unauthorized: true

tools:
  whoami:
    source: ibmi-system
    description: Get current ibm i user
    statement: values(current_user)

  related_objects:
    source: ibmi-system
    description: Get related objects for a given database object
    statement: |-
      SELECT * FROM TABLE(
        SYSTOOLS.RELATED_OBJECTS(LIBRARY_NAME => :library,
        FILE_NAME => :object_name)
      )
    parameters:
      - name: library
        type: string
        description: The name of the library where the object is defined
        required: true
      - name: object_name
        type: string
        description: The name of the database object
        required: true

toolsets:
  dev_tools:
    tools:
      - whoami
      - related_objects
```

How to build an MCP tool for the IBM i MCP server



Step 1: Start with SQL

```
select CPU_TIME, A.* FROM
table(QSYS2.ACTIVE_JOB_INFO(
    SUBSYSTEM_LIST_FILTER => 'QUSRWRK,QSYSWRK')
) A
ORDER BY CPU_TIME DESC
LIMIT 3;
```

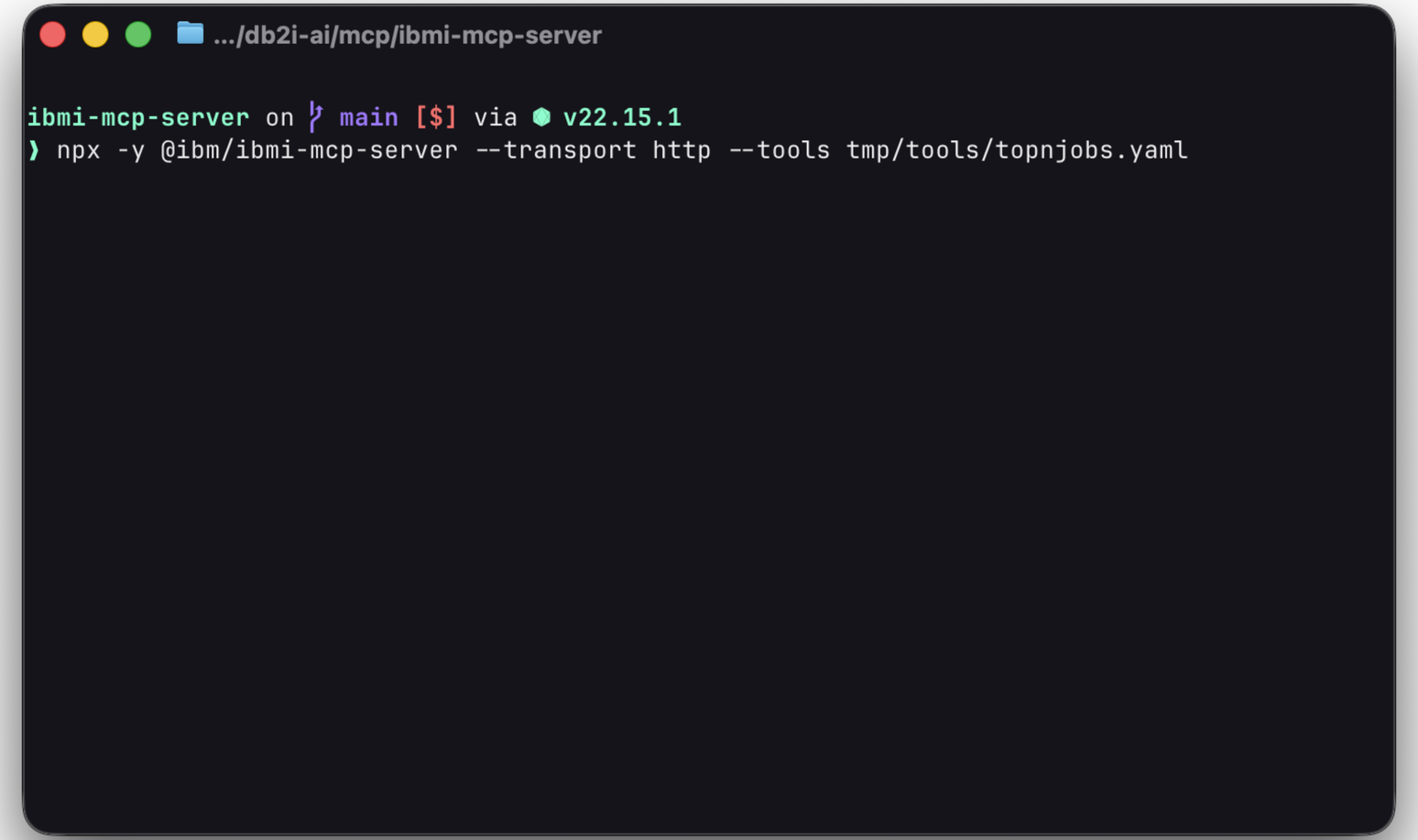
Step 2: Build YAML tool file

Tool Metadata:

- `source:` the IBM i
- `description`: Human-readable summary
- `parameters` (optional)
  - limit(int)
  - default = 10
- `statement:` the SQL

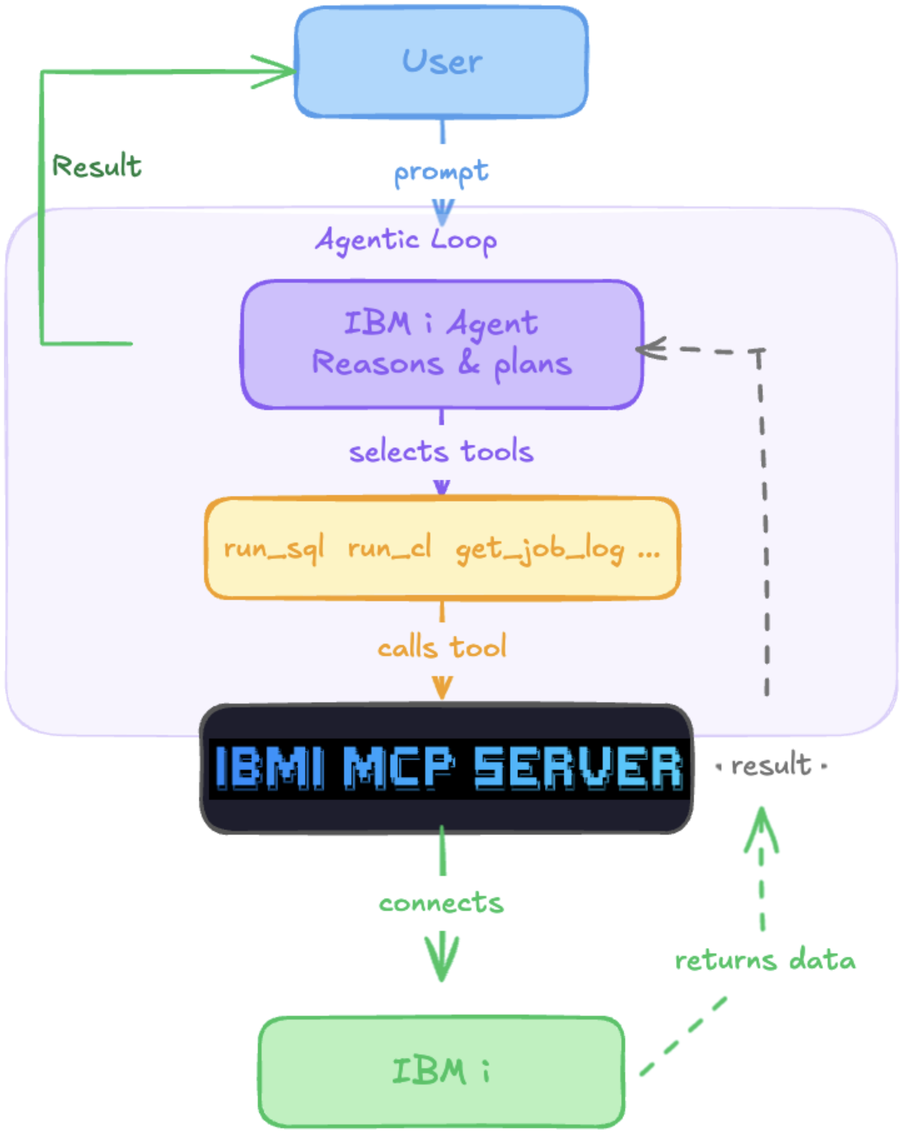
```yaml
sources:
  ibmi-system:
    host: ${DB2i_HOST}
    user: ${DB2i_USER}
    password: ${DB2i_PASS}
    port: 8076
    ignore-unauthorized: true

tools:
  active_job_info:
    source: ibmi-system
    description: "Find the top 10 consumers of CPU in the QUSRWRK and QSYSWRK subsystems"
    parameters:
      - name: limit
        type: integer
        default: 10
        description: "Number of top CPU consumers to return"
    statement: |
      SELECT CPU_TIME, A.* FROM
      TABLE(QSYS2.ACTIVE_JOB_INFO(
          SUBSYSTEM_LIST_FILTER => 'QUSRWRK,QSYSWRK')
      ) A
      ORDER BY CPU_TIME DESC
      FETCH FIRST :limit ROWS ONLY

toolsets:
  performance:
    tools:
      - active_job_info
```

Run the Server with the Tool file

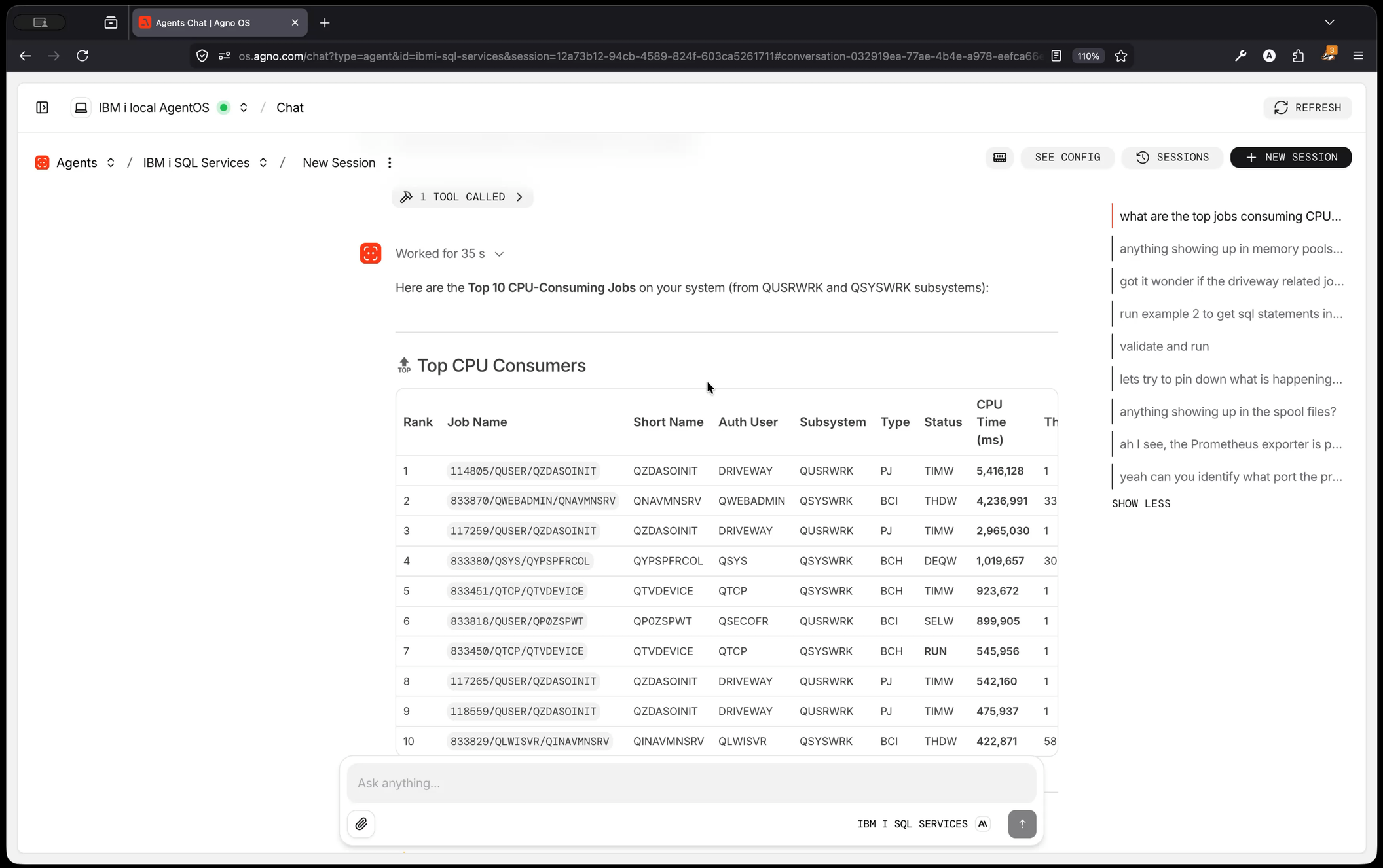


**Remember to setup up your .env file!*

IBM i Agent + MCP Server Flow



```python
"""Create an IBM i AI agent built with Agno + MCP."""

from agno.agent import Agent
from agno.models.anthropic import Claude
from agno.tools.mcp import MCPTools

agent = Agent(
    name="IBM i Assistant",
    model=Claude(id="claude-sonnet-4-5"),
    description="You are an expert IBM i database and systems assistant.",
    instructions=[
        "Use SQL tools to explore schemas, validate queries, and run SQL on Db2 for i.",
        "Always validate SQL with `validate_query` before executing.",
        "Use fully qualified names like SCHEMA.TABLE (e.g. SAMPLE.EMPLOYEE).",
        "Present results in formatted markdown tables.",
    ],
    tools=[
        MCPTools(
            url="http://localhost:3010/mcp",
            transport="streamable-http",
            include_tools=[
                "list_tables_in_schema",
                "validate_query",
                "execute_sql",
                "sample_rows",
            ],
        )
    ],
    markdown=True,
)

if __name__ == "__main__":
    agent.print_response("What tables are in the SAMPLE schema?", stream=True)

```


Agents Chat | Agno OS
os.agno.com/chat?type=agent&id=ibmi-sql-services&session=12a73b12-94cb-4589-824f-603ca5261711#conversation-032919ea-77ae-4b4e-a978-eefca66
110%
IBM i local AgentOS
Chat
REFRESH
Agents
IBM i SQL Services
New Session
SEE CONFIG
SESSIONS
NEW SESSION
1 TOOL CALLED
Worked for 35 s
Here are the Top 10 CPU-Consuming Jobs on your system (from QUSRWRK and QSYSWRK subsystems):
Top CPU Consumers
Rank | Job Name | Short Name | Auth User | Subsystem | Type | Status | CPU Time (ms) | Th
1 | 114805/QUSER/QZDASOINIT | QZDASOINIT | DRIVEWAY | QUSRWRK | PJ | TIMW | 5,416,128 | 1
2 | 833870/QWEBADMIN/QNAVMNSRV | QNAVMNSRV | QWEBADMIN | QSYSWRK | BCI | THDW | 4,236,991 | 33
3 | 117259/QUSER/QZDASOINIT | QZDASOINIT | DRIVEWAY | QUSRWRK | PJ | TIMW | 2,965,030 | 1
4 | 833380/QSYS/QYPSPFRCOL | QYPSPFRCOL | QSYS | QSYSWRK | BCH | DEQW | 1,019,657 | 30
5 | 833451/QTCP/QTVDEVICE | QTVDEVICE | QTCP | QSYSWRK | BCH | TIMW | 923,672 | 1
6 | 833818/QUSER/QP0ZSPWT | QP0ZSPWT | QSECOFR | QUSRWRK | BCI | SELW | 899,905 | 1
7 | 833450/QTCP/QTVDEVICE | QTVDEVICE | QTCP | QSYSWRK | BCH | RUN | 545,956 | 1
8 | 117265/QUSER/QZDASOINIT | QZDASOINIT | DRIVEWAY | QUSRWRK | PJ | TIMW | 542,160 | 1
9 | 118559/QUSER/QZDASOINIT | QZDASOINIT | DRIVEWAY | QUSRWRK | PJ | TIMW | 475,937 | 1
10 | 833829/QLWISVR/QINAVMNSRV | QINAVMNSRV | QLWISVR | QSYSWRK | BCI | THDW | 422,871 | 58
what are the top jobs consuming CPU...
anything showing up in memory pools...
got it wonder if the driveway related jo...
run example 2 to get sql statements in...
validate and run
lets try to pin down what is happening...
anything showing up in the spool files?
ah I see, the Prometheus exporter is p...
yeah can you identify what port the pr...
SHOW LESS
Ask anything...
IBM I SQL SERVICES

5. Some of many approaches

IBM watsonx Orchestrate

Empower customers and employees through simple, intuitive and guided conversations

Customers

Employees

Subject matter experts

Agent Catalog

Agents for HR

Agents for Sales

Agents for Procurement

Agents for Customer Care

Custom built Agents

Orchestrator Agent
AI Agent supervises and manages how work is executed across assistants, agents, and skills

Supervising

Routing

Reasoning

Planning

Boost productivity with AI and automation

AI Agents
Uses tools and other agents to plan and act, automating complex tasks

Tools
APIs and automations that can be used by agents to accomplish tasks.

Accelerate time to value with pre-built capabilities or build your own

Studio
Discover, create and manage agents and tools that orchestrate tasks and workflows.

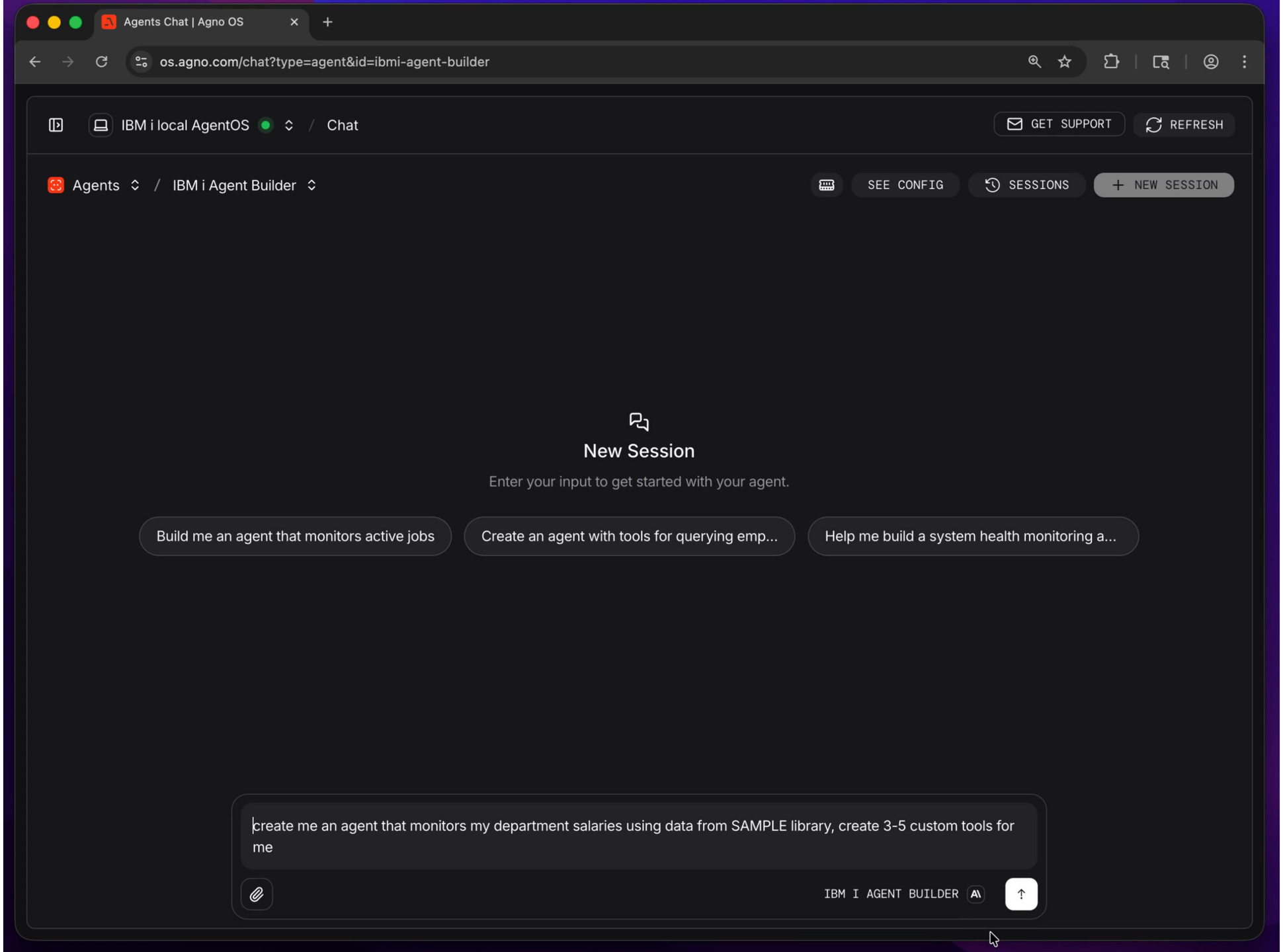
Prebuilt tools

Custom tools – build new or discovery existing

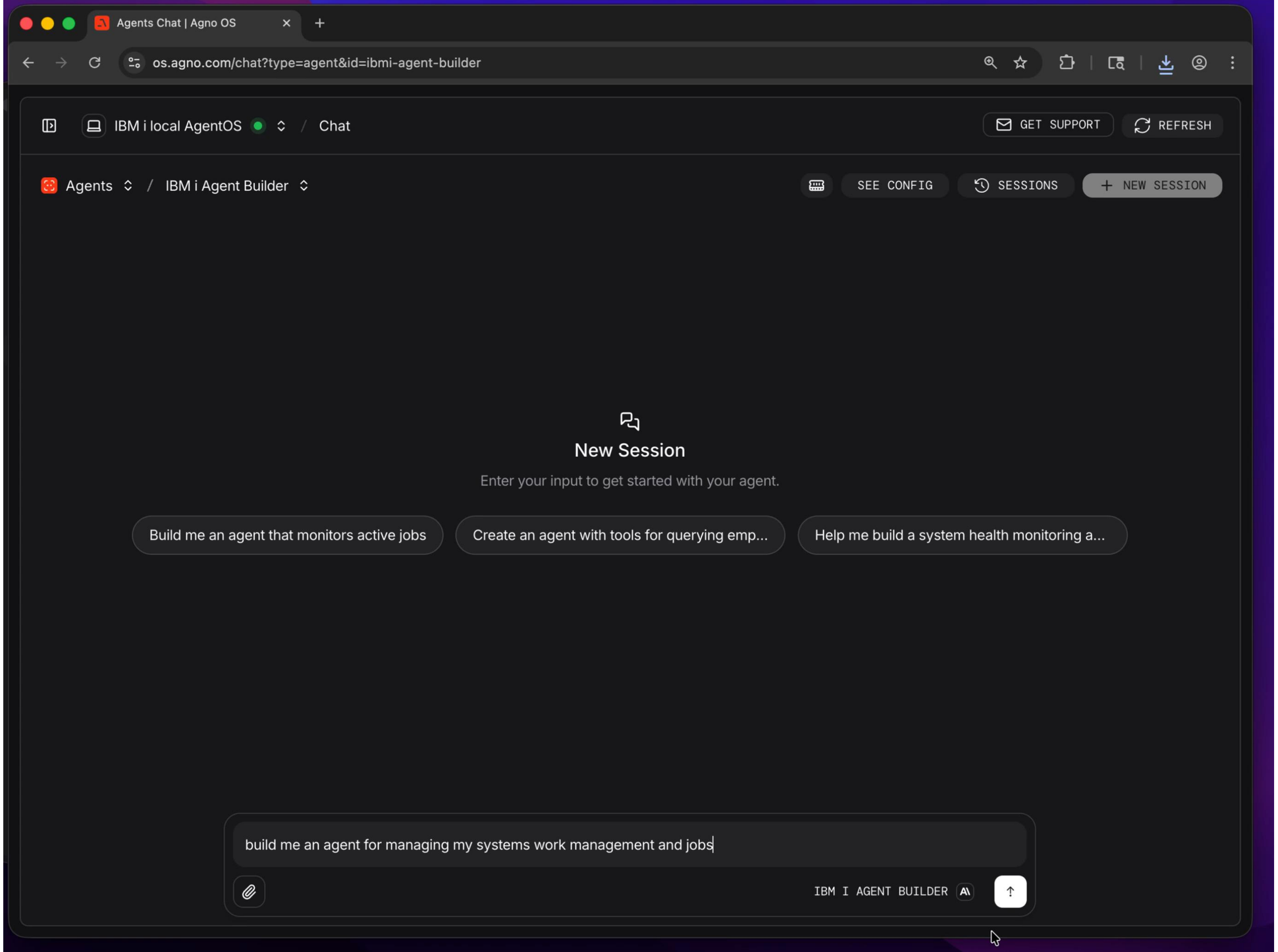
Gen AI
powered by watsonx.ai™

Live Demo: Claude

Sneak peek demo: Custom agent creation (minimal guidance)


Agents Chat | Agno OS
os.agno.com/chat?type=agent&id=ibmi-agent-builder
IBM i local AgentOS
Chat
GET SUPPORT
REFRESH
Agents
IBM i Agent Builder
SEE CONFIG
SESSIONS
NEW SESSION
New Session
Enter your input to get started with your agent.
Build me an agent that monitors active jobs
Create an agent with tools for querying emp...
Help me build a system health monitoring a...
create me an agent that monitors my department salaries using data from SAMPLE library, create 3-5 custom tools for me
IBM I AGENT BUILDER

Sneak peak demo:
Custom agent creation
(custom SQL-driven)


Agents Chat | Agno OS
os.agno.com/chat?type=agent&id=ibmi-agent-builder
IBM i local AgentOS
Chat
GET SUPPORT
REFRESH
Agents
IBM i Agent Builder
SEE CONFIG
SESSIONS
NEW SESSION
New Session
Enter your input to get started with your agent.
Build me an agent that monitors active jobs
Create an agent with tools for querying emp...
Help me build a system health monitoring a...
build me an agent for managing my systems work management and jobs
IBM I AGENT BUILDER

6. Securing AI agents

IBM i agent best practices

Leverage all the stuff you know

- Object level authorities
- SQL Query governor
- Workload capping groups

Maintain deny-by-default methodologies

- Agents run under minimal-access *USRPRF
- Audit using authority collection etc
- Run agents as “read only” when appropriate

Avoid usage of broad-spectrum MCP tools

- Run SQL
- Run CL command

Role-based access control

- Don’t expose all agents to all apps/users

7. Closing thoughts

The Honest Assessment

– The gap between "agent that can talk to a system" and "agent that can reliably operate within a system" is much larger than it looks

Why Building agents is hard:

- Agents hallucinate
- Memory is unsolved
- The packaging problem
- Production ≠ demo

Things to remember

- Start with a real problem, not a science project
- The most reliable governance is governance the agent can't opt out of
- Agents are as good as their tools
- You don't need to “boil the ocean”
- The platform question matters more than the model question

Practical next steps

Explore (this week)

- Run an SQL service you have never used before: `QSYS2.SERVICES_INFO`
- Investigate data that comes back
- This is the same interface an agent will use

Experiment (this month)

- Pick one repetitive task that your team does manually
- Write down the exact steps
- You've just designed your first agent

Build (this quarter)

- Connect the IBM i MCP server to Bob, Claude, Copilot, etc
- Build an agent from scratch (come to my session tm!!)

Follow along!

- GitHub: https://github.com/ajshedivy
- LinkedIn: https://www.linkedin.com/in/adam-shedivy-2619a1166/
- Email: adam.shedivy@ibm.com

- IBM i AgentOS demo: https://github.com/ajshedivy/ibmi-agentos
- MCP Server: https://github.com/IBM/ibmi-mcp-server