CICS 6.3 is AI Ready!

Kye Maloy – CICS AI Team Lead, IBM

What is an AI agent?

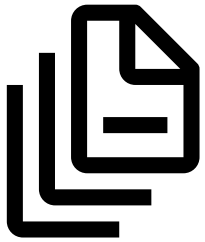
AI agents are autonomous programs powered by Large Language Models (LLMs) that can reason, plan, and take independent actions to achieve specific goals.

Key features include:

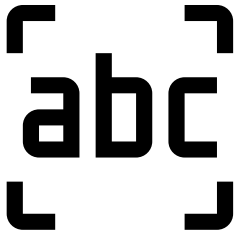
- Autonomy
- Tool use
- Goal-Orientated
- Memory
- Decision making

Building an LLM

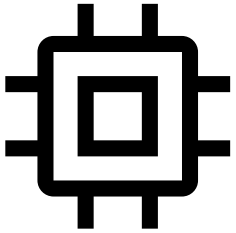
Training Data



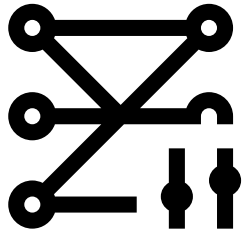
Tokenization



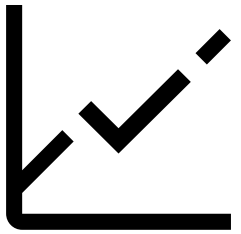
Pre-training



Fine Tuning



Evaluation



Time and cost

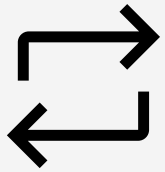
How an LLM tackles a question

1. Convert the query into tokens
2. Convert those tokens into vectors
3. Pass vectors through transformer and attention layers to refine understanding
4. Predict the next token
5. Feed new token plus original input back into model for autoregression
6. When complete, detokenize and display

“Who are you?”

[”Who”, “are”, “you”, “?”]

[1.256, 0.198, 2.201, 1.265]



[1.256, 0.198, 2.201, 1.265, 1.442, 2.012, 0.781....]

”Who are you? I am IBM Granite.”

LLM Limitations

1

If the answer is not in the original knowledge, the LLM cannot reference it. The data set is static until the model is updated.

2

Prediction means the LLM is going to generate the most probable token which can impact answers in less common domains. This can be controlled with parameters like temperature, but this is not foolproof.

3

No fact checking, and therefore hallucinations are generated as if they were fact. Bias' can also creep in depending on training data.

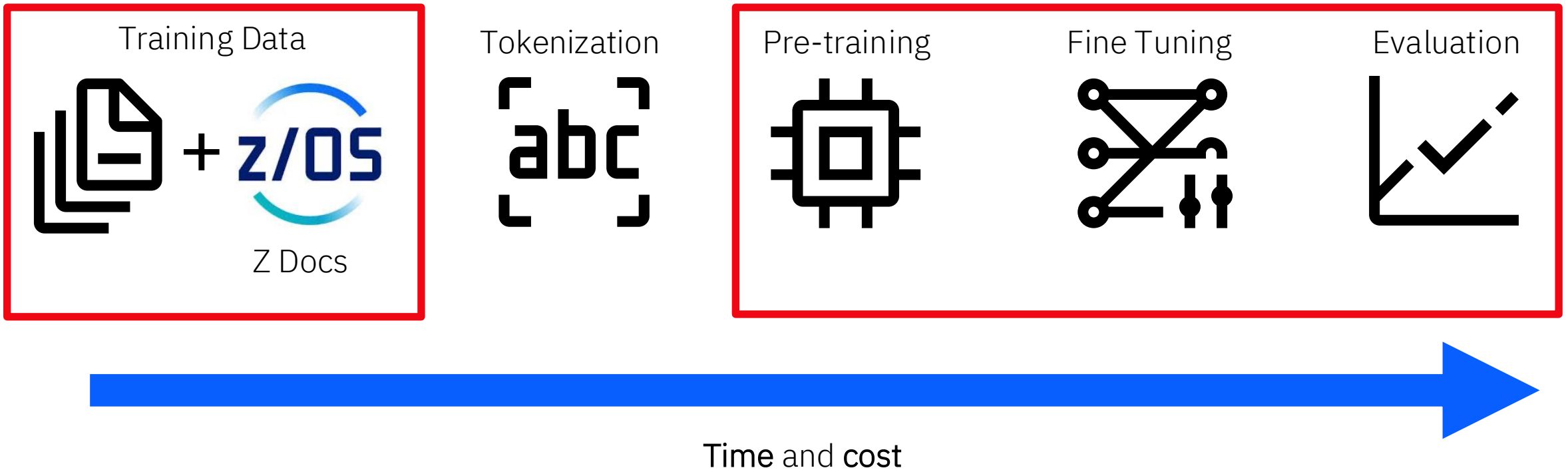
4

No internal understanding of why the model arrived at its answer. Black box model where the internals are hidden from the user.

Off-the-shelf LLMs do **not** understand the mainframe, it's nuances or its complexities!

OK… so why don’t we train an LLM to be Z tailored?

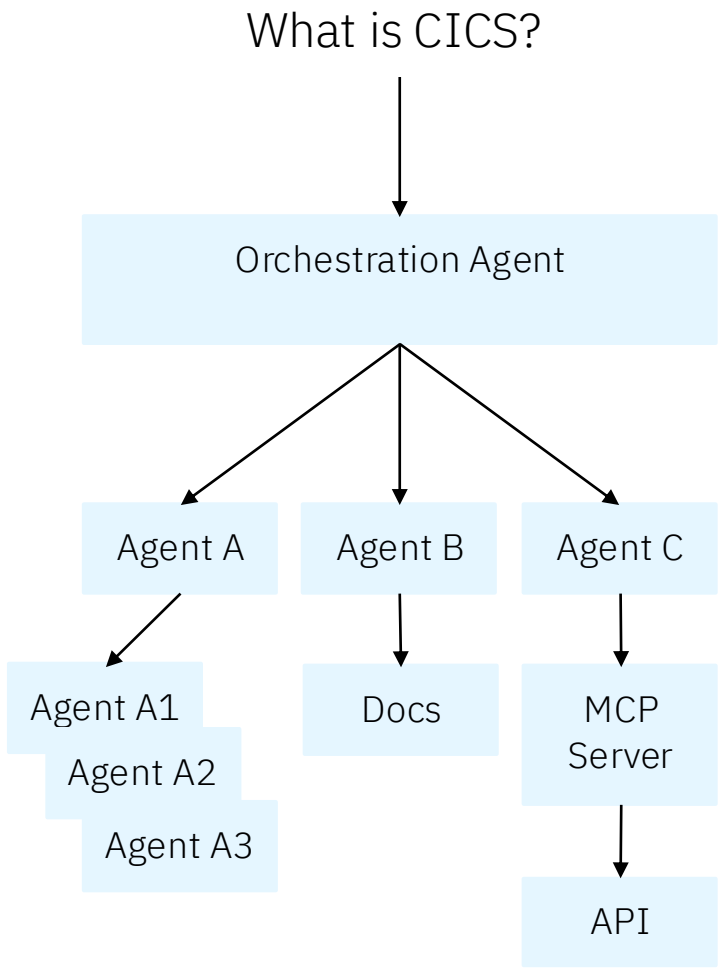
~~Building an LLM~~
Making a Z tailored LLM



So how can an AI Agent help solve this?

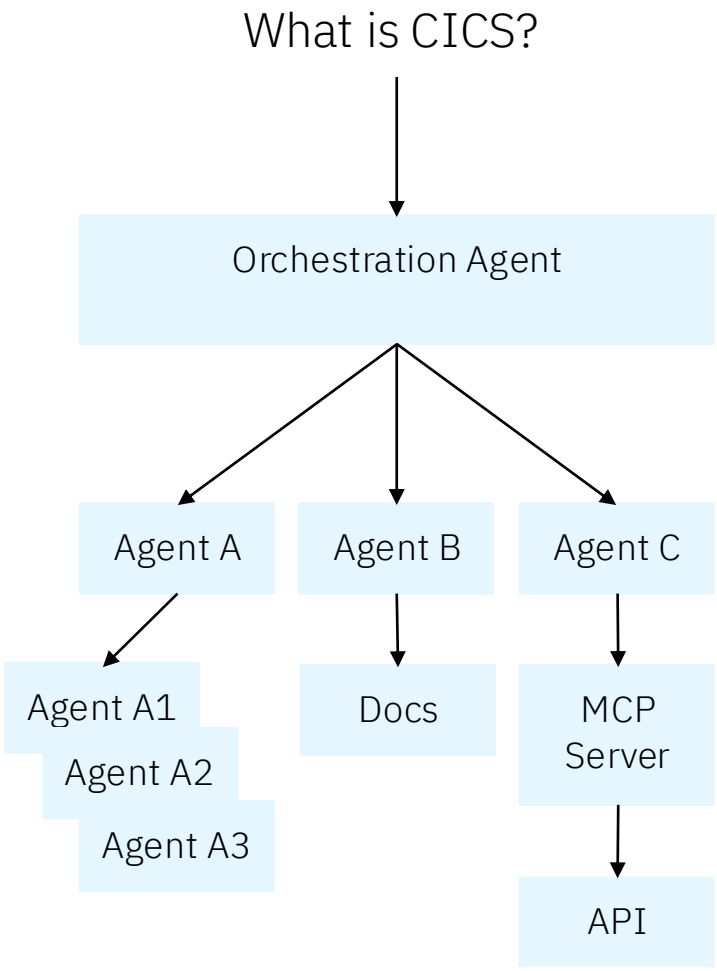
How an AI Agent tackles a question

1. Orchestrate a plan on how to get to the goal the user is asking for
2. Invoke tools, other agents, and LLMs to gather information, data and resources
3. Perceive this data, iterating upon results as they come in
4. Continue looping until a desired state is achieved
5. Return the answer to the user



How an AI Agent tackles a question

1. Orchestrate a plan on how to get to the goal the user is asking for
2. Invoke tools, other agents, and LLMs to gather information, data and resources
3. Perceive this data, iterating upon results as they come in
4. Continue looping until a desired state is achieved
5. Return the answer to the user



Prompt Example for AI Agent

Answer the following questions as best you can. You have access to the following tools:

{list_of_tools}

Use the following format:

Question: the input question you must answer
Thought: you should always think about what to do
Action: the action to take, should be one of [{tool_names}]
Action Input: the input to the action
Observation: the result of the action
... (this Thought/Action/Action Input/Observation can repeat N times)
Thought: I now know the final answer
Final Answer: the final answer to the original input question

Begin!

Question: {input}
Thought:{agent_scratchpad}

What is a Tool?

A tool encapsulates some sort of functionality an AI may find helpful in achieving a goal.

They can interact with an API to fetch, update or delete data.

These tools often take parameters to allow for the most flexibility possible.

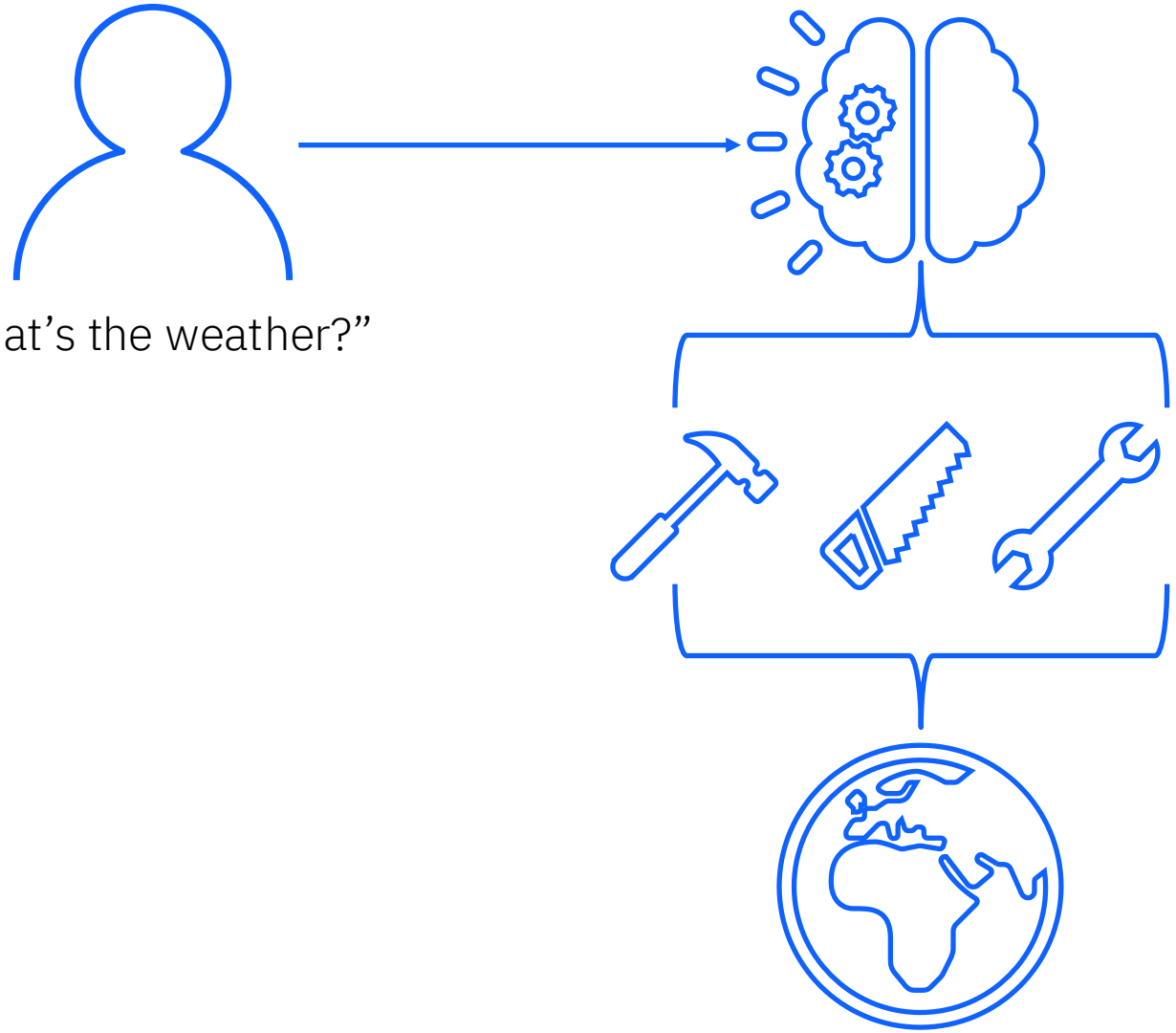
In a nutshell, tools wrap a function allowing an LLM to ”call” the function as it reasons

Example of a `get_weather()` tool

```
def get_weather(city: str) -> str:

    weather_database = {
      "London": {"temp": ”7°C", "condition": "Cloudy"},
      ”Orlando": {"temp": "21°C", "condition": "Sunny"},
    }

    data = weather_database.get(city.title())

    return f"The current weather in {city} is {data['temp']} and {data['condition']}."
```

Very simple encapsulation of some functionality that an LLM might want to access in an agentic workflow.

Binding Tools to an LLM



1.

LLM takes the question and creates a plan to execute to answer it.

2.

As the plan executes, the LLM decides when a tool usage is required, and the agentic framework then calls the functions.

3.

These tool calls give the LLM a wider scope of knowledge, and even access to real time data as it answers the question.

LLMs

- Foundational component of all AI workflows
- Knowledge cutoff means there's a limit to what can be known
- Very responsive/fast to return answers
- Cannot validate process LLM took to arrive at answer

AI Agents

- Plan and reason as they execute towards achieving a certain goal
- Uses tools, LLMs, and other agents as they progress through
- Can take longer as more process is going on under the covers
- Process can be traced as agent makes decisions

What does any of this have to do with CICS TS 6.3 being AI ready?

MCP Servers

An MCP (Model Context Protocol) server is a lightweight program that acts as a standardized bridge between AI applications and external data sources or tools (files, databases, APIs).

They typically provide 3 resource types: tools, resources and prompts.

Developed by Anthropic in Nov 2024, and donated to the Agentic AI Foundation (AAIF) in Dec 2025, MCP offers a standardisation of how tools and other MCP based resources are offered, removing any ambiguity.

Think of MCP as the "USB-C" for LLMs connecting to servers filled with useful tools and resources.

Key takeaway

Any MCP server can be used by any LLM application that supports the protocol, making the MCP server LLM application agnostic.

This opens up the CICS MCP server to any-and-all agents built with the protocol in mind.

How can we provide contextual data to AI Agents from a live CICS system?

Can we use tried and tested technologies we have been shipping for years?

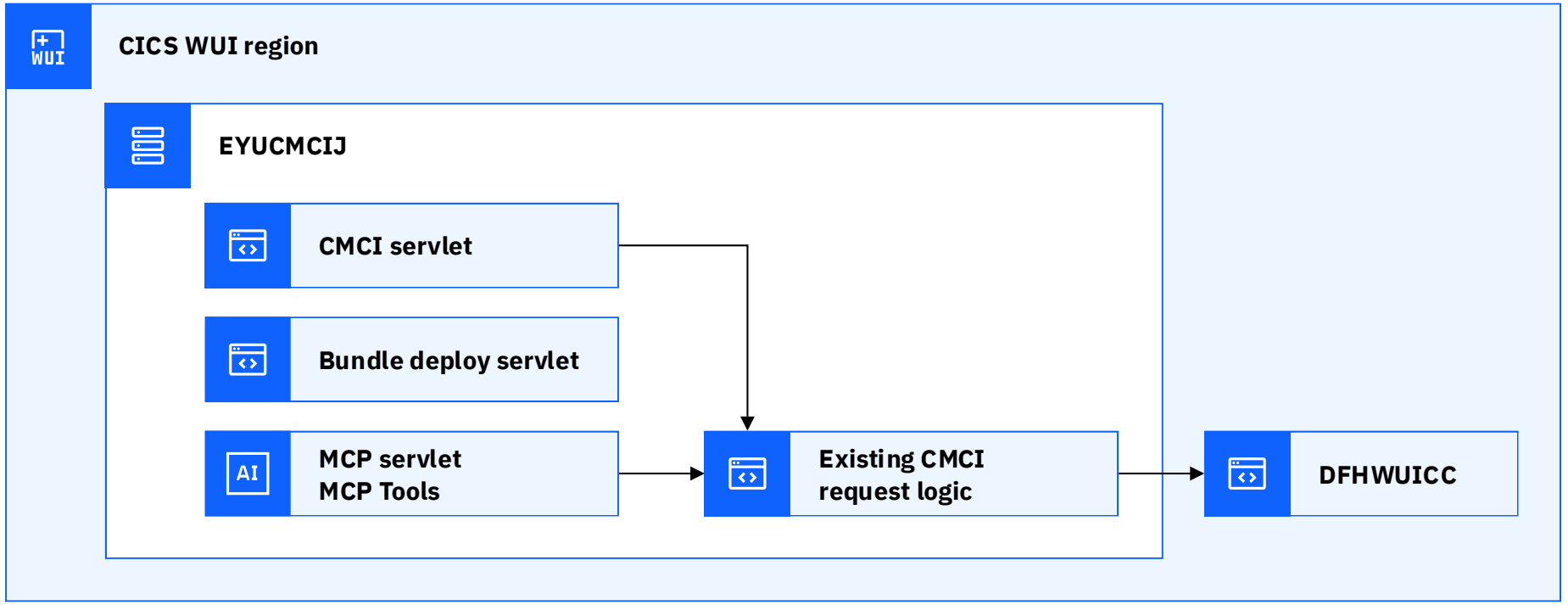
YES!

Introducing IBM CICS™ Auto-3270-reading and writing AI application enhancement suite for z/OS©

No, ***that's*** not it… MCP makes a lot more sense using CMCI as the underpinnings for communicating with CICS to retrieve live system data.

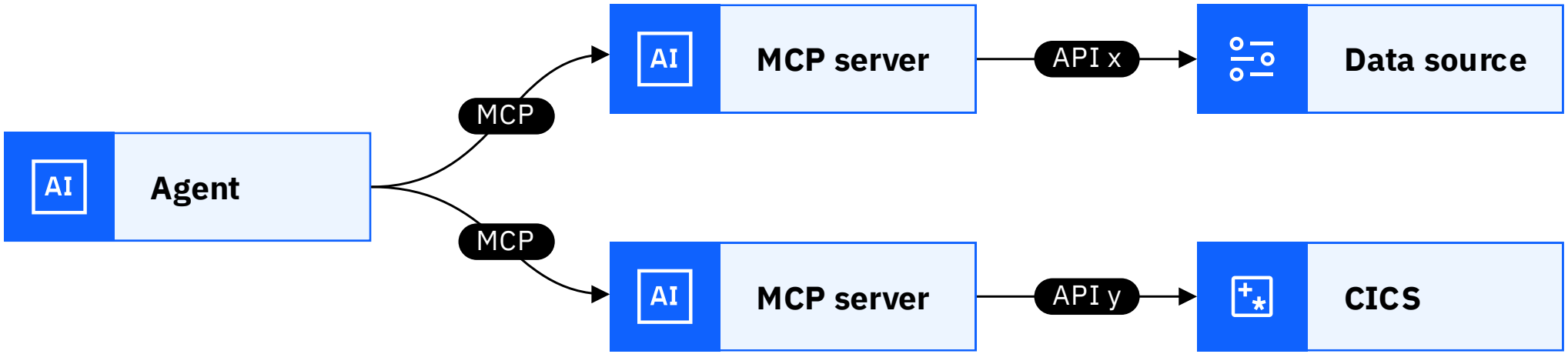
IBM CICS 6.3: MCP Server

Runs in a CMCI JVM in either a WUI or SMSS region. In a standalone region, the JVM server that runs the MCP server is called EYUCMCIJ. The MCP server is enabled by default from CICS TS 6.3 onwards. MCP work runs under the CICS-supplied transaction, CWMC.



CICS AI Agents with MCP access

Agents can connect to the CICS MCP server, just as they can any other MCP server and gather useful live data about CICS regions.



What data is MCP allowing access to?

The MCP server has access to provide an AI agent with the following resources and data, including (but not limited too):

- Local Transactions
- Dynamic Storage Areas
- System Dumps
- DB2 Entry
- Programs
- Local Files
- Transaction Dumps
- Unit of work links
- DB2 Connection
- MQ Connection

What data is MCP allowing access to?

The MCP server has access to provide an AI agent with the following resources and data, including (but not limited too):

Security with CICS 6.3 MCP Server

The MCP server uses the same security setup as the CMCI, including the existing CMCIUSER EJBROLE.

All MCP transactions run under CWMC and are distinct for auditability and security checks.

We support the same security mechanisms that are supported by the CMCI JVMServer.

Common Thoughts

Why do agents need to understand CICS Context

As we move toward agents that can control and update systems it is paramount that we can trust them with just read only capability.

Even then an agent that can understand the status of your system is far more valuable than just an agent that provides information

Why can't agents just use the CMCI?

The CMCI is a bespoke XML based protocol built on top of HTTP. For an agent to use it we would have to train every base LLM with the capability.

We don't want to expose every attribute of every resource back to the LLM, we are being opinionated about returned data to best suit common use cases.

Is this just another Interface like GraphQL?

Although it is another protocol, its not one that we expect users to write code against. This is purely for Agentic use.

We are not going to replace the CMCI with MCP at any point.

I probably won't use it

Probably not directly, but this future proofs CICS to interact with Agents, integrating with solution stacks from IBM while keeping data on-Z in a place you understand and can control.

”I've just received DFHAC2016 for transaction TXN1”

DFHAC2016 is an abend code in CICS (Customer Information Control System), which is used for online transaction processing. This specific code, DFHAC2016, indicates a program check condition. It means that the program you're running encountered an unexpected condition and couldn't proceed normally.

Here's a breakdown:

- **DFH**: CICS abend codes typically start with this prefix.
- **A**: This signifies an abnormal termination.
- **C**: It represents a program check or protection exception.
- **2016**: This is the specific condition code. In this case, it often indicates a failed storage access - perhaps trying to read from or write to memory that the program isn't allowed to access.

To resolve DFHAC2016:

1. **Check your logs and trace data**: Look for more detailed messages around the time of the abend. This can give clues about what
part of the code failed and why.

2. **Review the failing program**: Examine the area in your code where the abend occurred. It might be a buffer overflow, an attempt
to access uninitialized memory, or a null pointer dereference.

.....

The error DFHAC2016 indicates that transaction TXN1 cannot run because the initial program for transaction TXN1 is not available. Possible reasons include:

1. The program is missing.
2. The installed program definition is missing.
3. The program is disabled.
4. The program name in the installed transaction definition is invalid.
5. The installed transaction has been defined as remote and therefore has no program name, but the name of the remote system is the same as that of the local system.
6. The program requires a JVM to run but JVM initialization failed.
7. The autoinstall program abended while attempting to load the program.
8. The program requires a JVM server to run but the JVM server is not available.
9. To resolve this issue, you should determine the cause of the error using the list provided in the explanation. The response depends on the reason as follows:
    1. Load the program into the CICS program library.
    2. Create an installed program definition for the program.
    3. Enable the program.
    4. Use a valid program name in the installed transaction definition.
    5. Use a local version of this transaction.
    6. Use the correct remote version of this transaction.
    7. Logon to the correct system and retry the transaction.
    8. For JVM programs, check the CSMT log for further information as to why JVM initialization failed.
    9. Check the job output for further information as to why the autoinstall program abended.
    10. Please follow the appropriate steps based on the cause of the error.

The error DFHAC2016 indicates that transaction TXN1 cannot run because the initial program for transaction TXN1 is not available. Possible reasons include:

1. The program is missing.
2. The installed program definition is missing.
3. **The program is disabled.**
4. The program name in the installed transaction definition is invalid.
5. The installed transaction has been defined as remote and therefore has no program name, but the name of the remote system is the same as that of the local system.
6. The program requires a JVM to run but JVM initialization failed.
7. The autoinstall program abended while attempting to load the program.
8. The program requires a JVM server to run but the JVM server is not available.
9. To resolve this issue, you should determine the cause of the error using the list provided in the explanation. The response depends on the reason as follows:
    1. Load the program into the CICS program library.
    2. Create an installed program definition for the program.
    3. **Enable the program.**
    4. Use a valid program name in the installed transaction definition.
    5. Use a local version of this transaction.
    6. Use the correct remote version of this transaction.
    7. Logon to the correct system and retry the transaction.
    8. For JVM programs, check the CSMT log for further information as to why JVM initialization failed.
    9. Check the job output for further information as to why the autoinstall program abended.
    10. Please follow the appropriate steps based on the cause of the error.

The error DFHAC2016 indicates that transaction TXN1 cannot run because the initial program for transaction TXN1 is not available. Possible reasons include:

1. The program is missing.
2. The installed program definition is missing.
3. **The program is disabled.**
4. The program name in the installed transaction definition is invalid.
5. The installed transaction has been defined as remote and therefore has no program name, but the name of the remote system is the same as that of the local system.
6. The program requires a JVM to run but JVM initialization failed.
7. The autoinstall program abended while attempting to load the program.
8. The program requires a JVM server to run but the JVM server is not available.
9. To resolve this issue, you should determine the cause of the error using the list provided in the explanation. The response depends on the reason as follows:
    1. Load the program into the CICS program library.
    2. Create an installed program definition for the program.
    3. **Enable the program.**
    4. Use a valid program name in the installed transaction definition.
    5. Use a local version of this transaction.
    6. Use the correct remote version of this transaction.
    7. Logon to the correct system and retry the transaction.
    8. For JVM programs, check the CSMT log for further information as to why JVM initialization failed.
    9. Check the job output for further information as to why the autoinstall program abended.
    10. Please follow the appropriate steps based on the cause of the error.

The DFHAC206 code indicates that a transaction is not executable because its initial program is not available. This could be due to several reasons, including the program being missing, disabled or having an invalid program name in the transaction definition.

Live System Data

Looking at the live CICS data, I can see that the program PROGA1 associated with transaction TXN1 has a status of DISABLED, This suggests that the program is not available for execution due to its disabled status

Resolution

To resolve this issue, you must enable the program PROGA1, this can be done by updating the program's status to ENABLED

Extra Information

from the live system, I gathered the following information:
Transaction: TXN1
Program: PROGA1
Status: ENABLED
Routing: Static

Program: PROGA1
Status: DISABLED
JVMServer: None
Language: COBOL

Demo of accessing MCP server via MCP inspector and CICS AI Agent using tools

Where we are now with Agent Maturity

Closing Thoughts

We are early on the road of this cutting-edge journey.

The MCP server shows IBM and CICS commitment to providing cutting edge solutions and integrations to make your mainframe experience delightful, modern and to power advanced new capabilities.

Data is a key part of your organization, and we know a reason you choose Z is for its security, resiliency and stability.

By building AI solutions on Z, running in the systems and middleware's you know today, we build on these strong foundations and experiences.

Agentic AI is poised to change the way we all work, in all industries.

It is estimated that at least 15% of day-to-day work decisions will be made autonomously though Agentic AI by 2028, up from <1% in 2024.[1]

Agentic AI is not going away and has the capability of shifting the entire industry.

[1] https://www.gartner.com/en/newsroom/press-releases/2025-06-25-gartner-predicts-over-40-percent-of-agentic-ai-projects-will-be-canceled-by-end-of-2027


IBM

Experience more with IBM

Visit us at the IBM Booth #113

After a full day of technical sessions, take a break with us!

Connect with our experts, snap a photo with the z17 Plexi or the latest Telum II, and get an up-close look at our Spyre Accelerator.

Come back each day for fresh topics and demos at our expert stations.

Think 2026

Join 5000+ senior business and technology leaders who are seizing the AI revolution to unlock unprecedented growth and productivity at Think 2026.

Find out more information using the QR code below.

IBM Digital Asset Haven

IBM Digital Asset Haven is the operational backbone for financial institutions and regulated enterprises entering the digital asset economy.

Find out more information using the QR code below.

Your feedback is important!

SHARE
EDUCATE • NETWORK • INFLUENCE

Submit a session evaluation for each session you attend:

www.share.org/evaluation