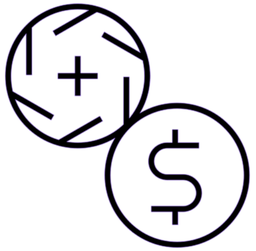
The All New CICS TS 6.3

Mark Cocker
mark_cocker@uk.ibm.com
Product Manager, CICS TS

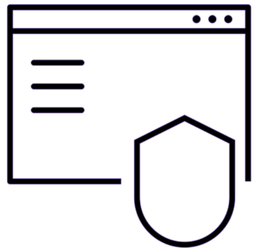
Stewart Francis
stewartfrancis@uk.ibm.com
CICS TS Development Lead and Developer Productivity

CICS Transaction Server for z/OS 6



Reduced cost of management and resiliency

CICS Admins can solve problems faster using OpenTelemetry observability, reduce costs by optimizing thread-safe access to data tables, reduce volumes of data written to SMF, and simplify and automate routine work with CICS configuration tools, CICS policies and Ansible

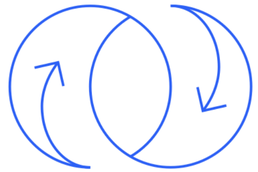


Improved security and compliance management

CICS and Security Admins can tighten security for valuable applications and data using CICS security recording and CICS Explorer tooling as part of a Zero Trust strategy, secure all connections with AT-TLS and TLS 1.3, and adopt best-practices with z/OS health checks

Enhanced developer productivity

Developers can use familiar tooling in Eclipse or VS Code to develop CICS applications. Java developers have access to the latest versions of Java, Jakarta, Spring Boot and Node.js to extend applications, with fast local access to CICS programs and data



Increased business agility

Architects can unlock access to CICS applications with API enablement, messaging event driven architecture, and AI in-transaction inferencing

CICS Transaction Server for z/OS 6.3

At a glance

OpenTelemetry

Manage and diagnose problems across hybrid cloud applications using OpenTelemetry dashboards and tools. CICS captures, propagates and emits trace spans for each task. Works across z/OS Connect, IBM MQ, CICS TG, Db2 and IMS for a complete end-to-end trace

Modernization

Develop CICS applications in VS Code with enhanced editors from IBM and Zowe. Developers can define and deploy CICS resources using YAML alongside the code, with auto completion and automation to ensure they meet local conventions

AI agent ready

Quickly answer questions about CICS regions and configuration using AI assistants in natural language. CICS includes an MCP server to provide accurate information to AI agents during app development and problem diagnosis

Core

Consolidate and simplify CICS region configuration using YAML for easier management and provisioning. New policies to issue messages to the system log and new policy rules for JVM servers and APPC connections

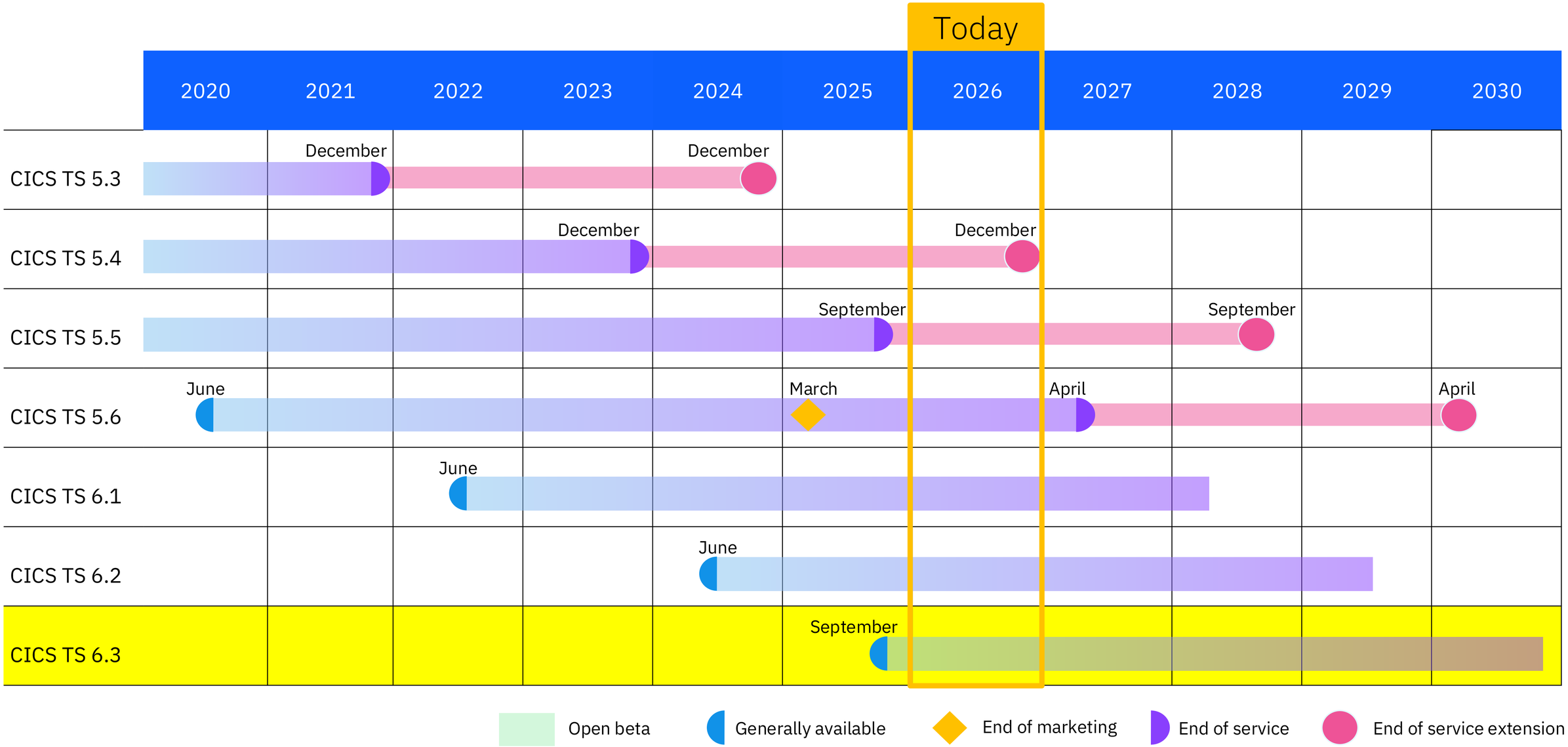
Security

Adopt Zero Trust principles and security best-practices with enhanced workflows in CICS Explorer, and and example security definition validation pipeline. AT-TLS can be used to secure IPIC and outbound HTTPS connections

Modern Languages

Enable developers to use the latest Java and Node.js support, including Java 21, Jakarta EE 10, and Spring Boot 3. CICS Java APIs have been further enhanced.

CICS TS 6



CICS TS announcements. IBM Lifecycle ”Extended” = at least 5 years in service + 3 years service extension

CICS TS 6.3 upgrading

Why

- *Reduce costs* and time by using new features to better manage and secure CICS, applications and access to data
- *Deliver more for your business* and developers with features to modernize CICS applications and optimized for latest IBM Z hardware and IBM middleware, providing opportunity for better performance and resilience
- *Includes all service fixes* from previous releases

Upgrade steps

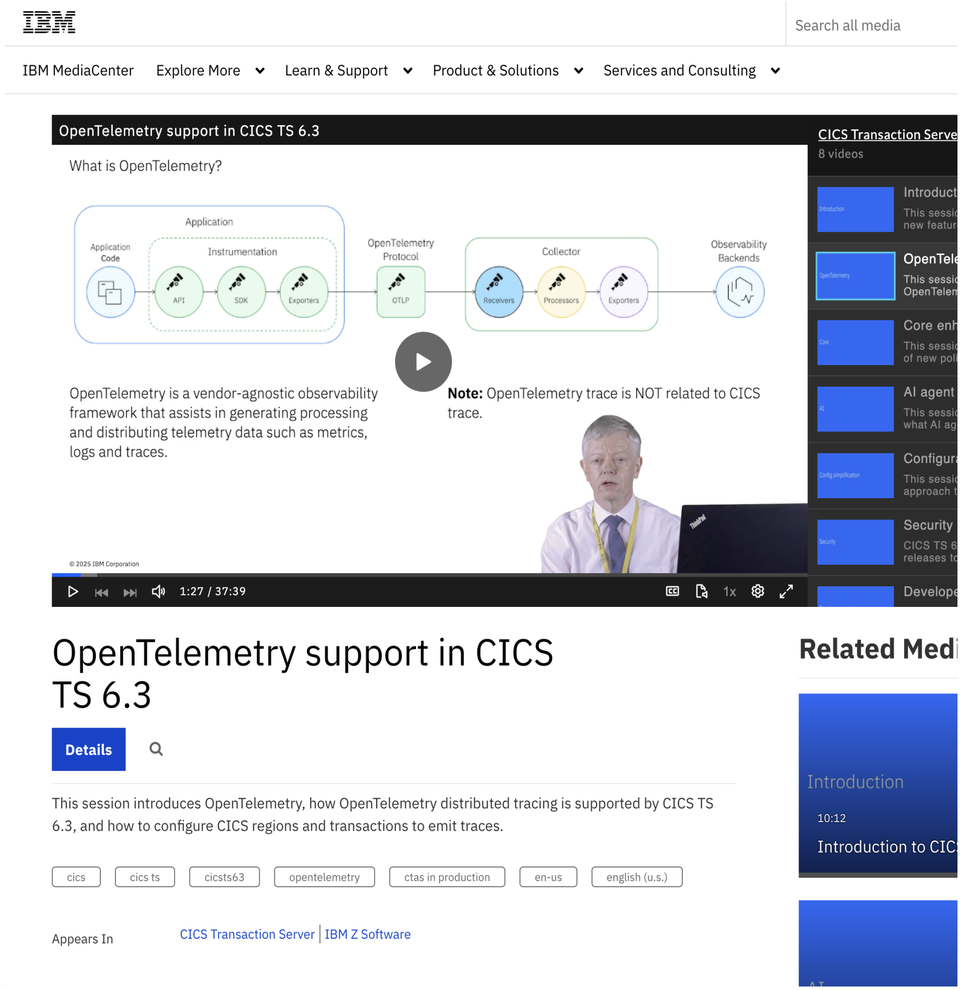
- *Upgrade directly* from any previous release
- *Simplified installation* with zOSMF (optional) and documentation
- *Hardware*: IBM z14 or higher
- *Software*: IBM z/OS 2.5 or higher
  - z/OS 3.1 or above required to support OpenTelemetry
  - Java 17 or higher
  - Full details in system requirements report
- Documentation for CICS TS version 6 has been combined and tables & annotations used to call out release specific info
- IBM Hursley Lab Services provide a range of upgrade workshops

CICS TS 6.3 upgrading

Education videos on IBM MediaCenter

[CICS TS 6.3 Education]

- Introduction
- OpenTelemetry
- Core enhancements
- AI agent support
- Configuration simplification
- Security enhancements
- Developer experience enhancements
- Modern languages enhancements



CICS TS 6.3 upgrading

Stabilized technologies
- unlikely to receive updates

- APPC password expiration management
- CICS Application Debugging Profile Manager
- CICS debugging tools sockets interface
- CICS Liberty features: cicsts:jcaLocalEci-1.0, cicsts:zosConnect-1.0, cicsts:zosConnect-2.0
- CICS system events
- CICS TS Application Handler Java interface
- CICSPlex SM Real-Time Analysis
- CICSPlex SM Web User Interface
- CICS Web Interface COMMAREA interface
- DFHWBCLI web client interface
- Enterprise Bundle Archive (EBA)
- ONC RPC
- PDF documentation
- Release sensitive XPI call RELSENSCALL
- SAML using the CICS STS
- Signing and encrypting SOAP messages for the WS-Security feature
- SOTUNING system initialization parameter
- USEROUTPUTCLASS function
- WS-Security infrastructure options
- WSDL 2.0 – use 1.1

Deprecated functions
- will be removed

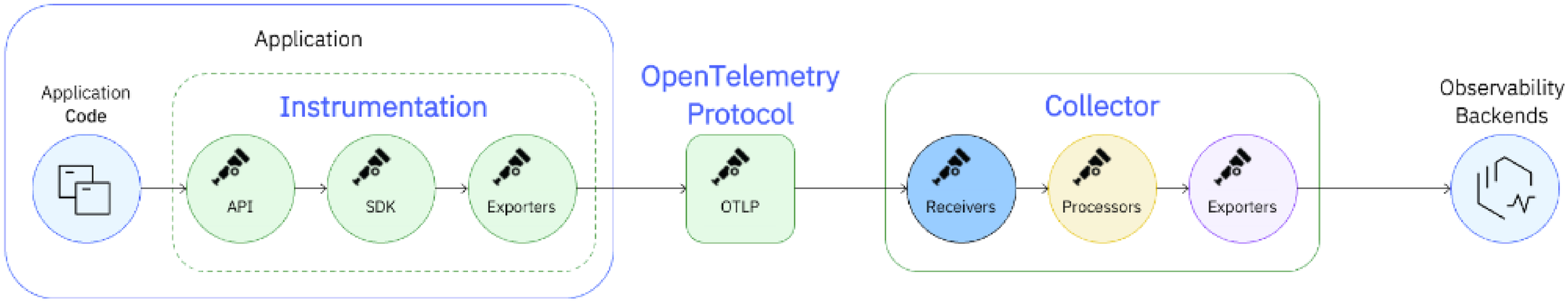
- CICS Liberty feature cicsts:zosConnect-1.0
- JVMSERVER-based configuration option for the web services data transformation service
- USEROUTPUTCLASS function
- CICS Service Flow Runtime
- Extended Recovery Facility

Discontinued functions
- Has been removed

- CICS Liberty feature cicsts:zosConnect-2.0
- Java 8 and Java 11
- JCA ECI adapter
- SAML using the CICS STS
- Signing and encrypting SOAP messages for the WS-Security feature
- Transport Layer Security (TLS) 1.0
- XSNEX global user exit

OpenTelemetry

What is OpenTelemetry



Open telemetry is a vendor-agnostic observability framework that assists in generating processing and distributing telemetry data such as **metrics**, **logs** and **traces**

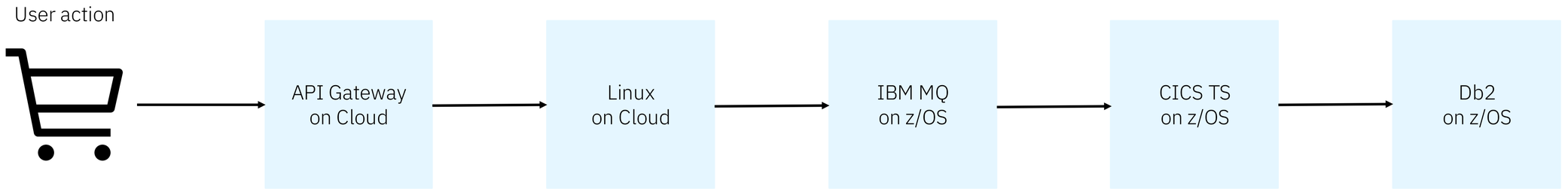
Evolution of OpenTelemetry signifies a move from proprietary vendor-specific observability solutions towards a standardized, open approach

N.B. OpenTelemetry trace is **NOT** related to CICS trace.

What is an OpenTelemetry trace?

The big picture of what happens when a request is made to an application

Understand the full path a request takes



OpenTelemetry represents a move away from vendor-specific observability solutions towards a standardized stack

OpenTelemetry encompasses more areas of observability than trace (logs, metrics). At this time, we're just talking about trace

Why OpenTelemetry trace?

To isolate the location of a problem in a distributed application, it is necessary to understand for an individual request journey it has taken...

... and where it finished?

With a fragmented observability architecture, this information might not be available, or needs to be pieced together manually

No-one can see the big picture

Diagnosing the cause of an outage takes longer, resulting in more downtime

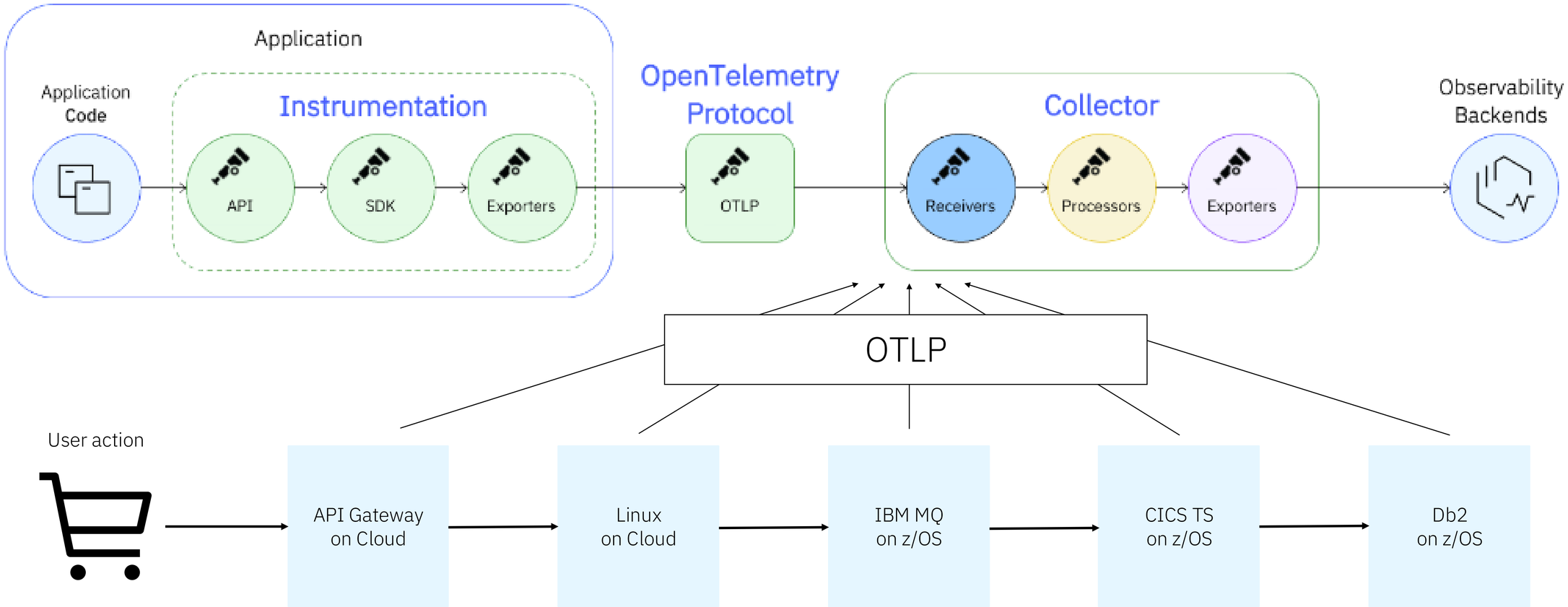
The promise of OpenTelemetry is that by observing the same standards, ubiquitous end-to-end observability of an application becomes possible

SREs can see the entire journey for a request, including where it went bang – no telemetry black holes!

Diagnosing and fixing outages is faster

Maybe you don’t even get paged!

Instrumentation

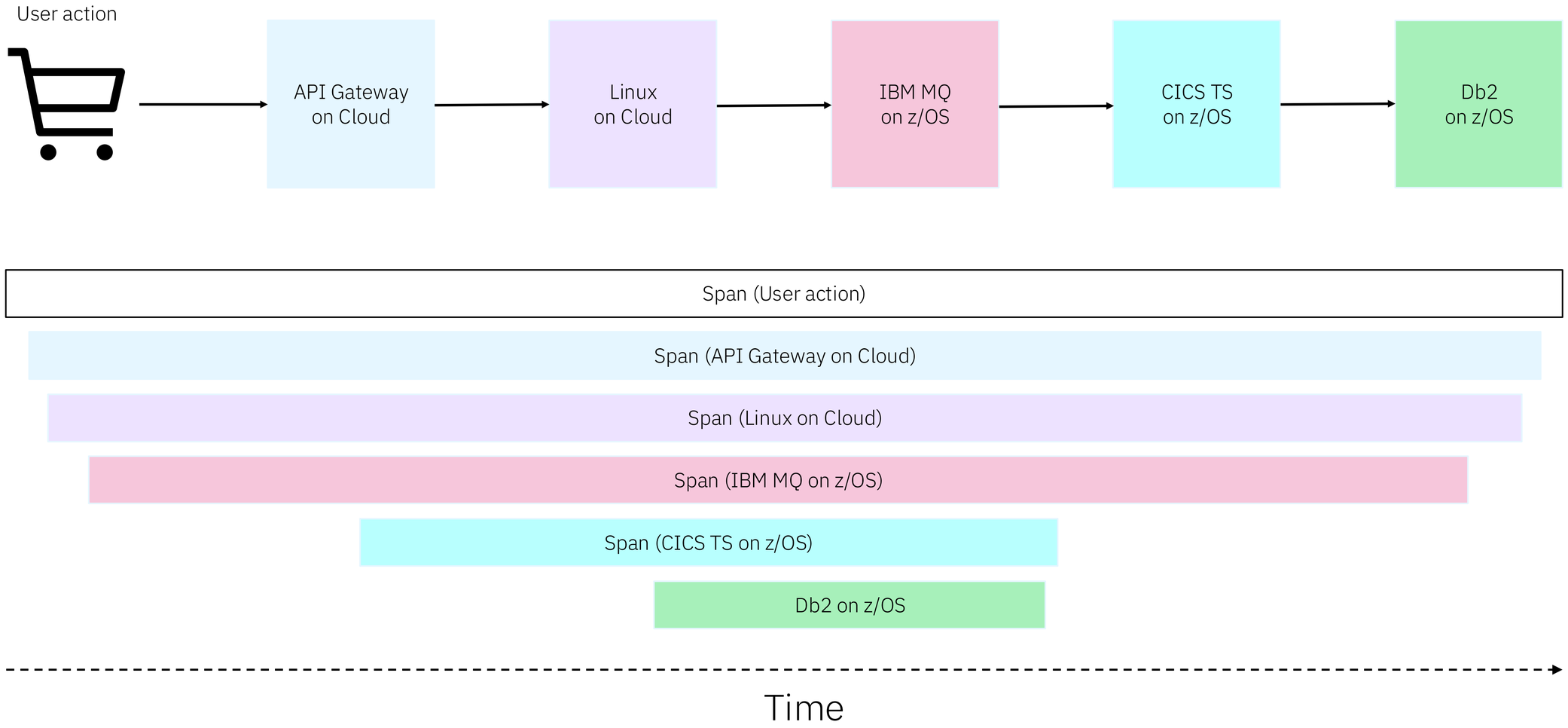


Runtimes are instrumented to emit signals corresponding to application request events

CICS TS 6.3 introduces a **zero-code** solution for emitting OpenTelemetry application trace

Trace data is aggregated and can be explored using an **Observability Backend**

Spans



For a request, you can see its entire journey through your distributed application, the hierarchy of invocation of services, how long it spent in each service, and additional meta data associated with each service (e.g. for CICS which transaction, etc)

The details…

Learn more about OpenTelemetry in CICS TS this week, including:

- How CICS TS OpenTelemetry instrumentation actually works
- Detailed information about software pre-reqs (e.g. z/OS 3.1)
- What information is collected in CICS TS spans
- How to configure OpenTelemetry in CICS TS
- See a demo scenario, with real data in an Observability Backend

OpenTelemetry Tracing With CICS TS 6.3
Speaker: Mark Cocker
Thursday 26th Feb
10:30am

AI

Large Language Models

In recent years LLMs and the agents that they power have exploded in popularity, and utility

LLMs are incredibly complex statistical models of the English language, which can generate tokens in a sequence based on some initial context

LLMs power many of the AI applications you'll be familiar with, or using today, like ChatGPT and Gemini

How Agents work

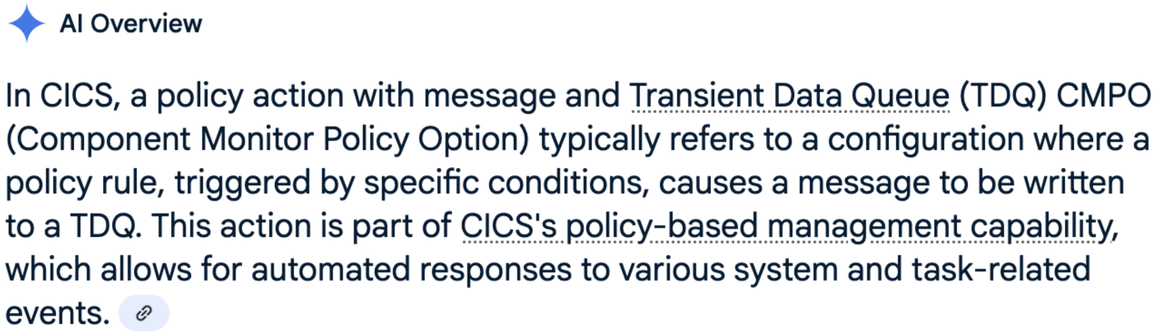
LLMs are very powerful, but to get the most out of them, you need to carefully construct your prompt, rather than relying on what the model fundamentally “knows”

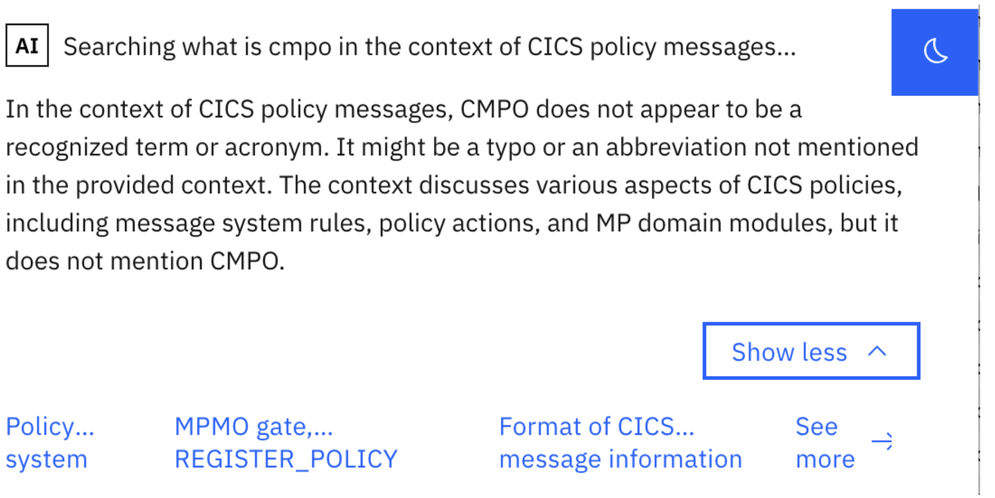
Agents are the programs that are doing this “prompt engineering”

One interaction with an agent may prompt an LLM many times!

Agents are typically constrained to a specific task, which helps to improve accuracy

[AI Agents Statement of Direction]





Agent maturity

AI tools like the IBM Docs search are advisory agents
What if when we built the prompt, we could provide additional context about your systems?

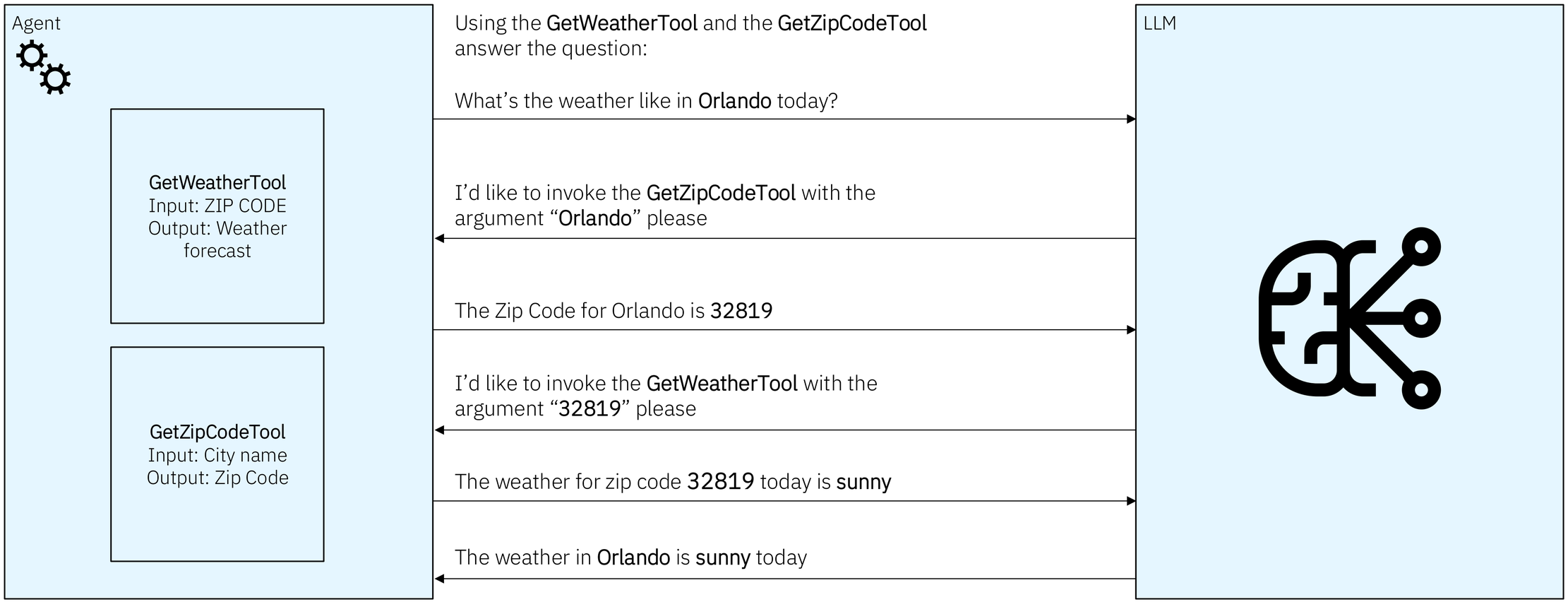
“Is the program **UPDCUST** threadsafe **in my environment**?”

The LLM doesn’t know anything about about “my environment”
We could provide that information in the prompt… but the user could have typed anything! How could we know that information was relevant ahead of time?

Tools

LLMs can decide that the information is relevant, and ask for it!
But they can't just ask for anything...
Agent developers can provide a "menu" of **tools** that the model can use to get more information

"What's the weather like in Orlando today?"



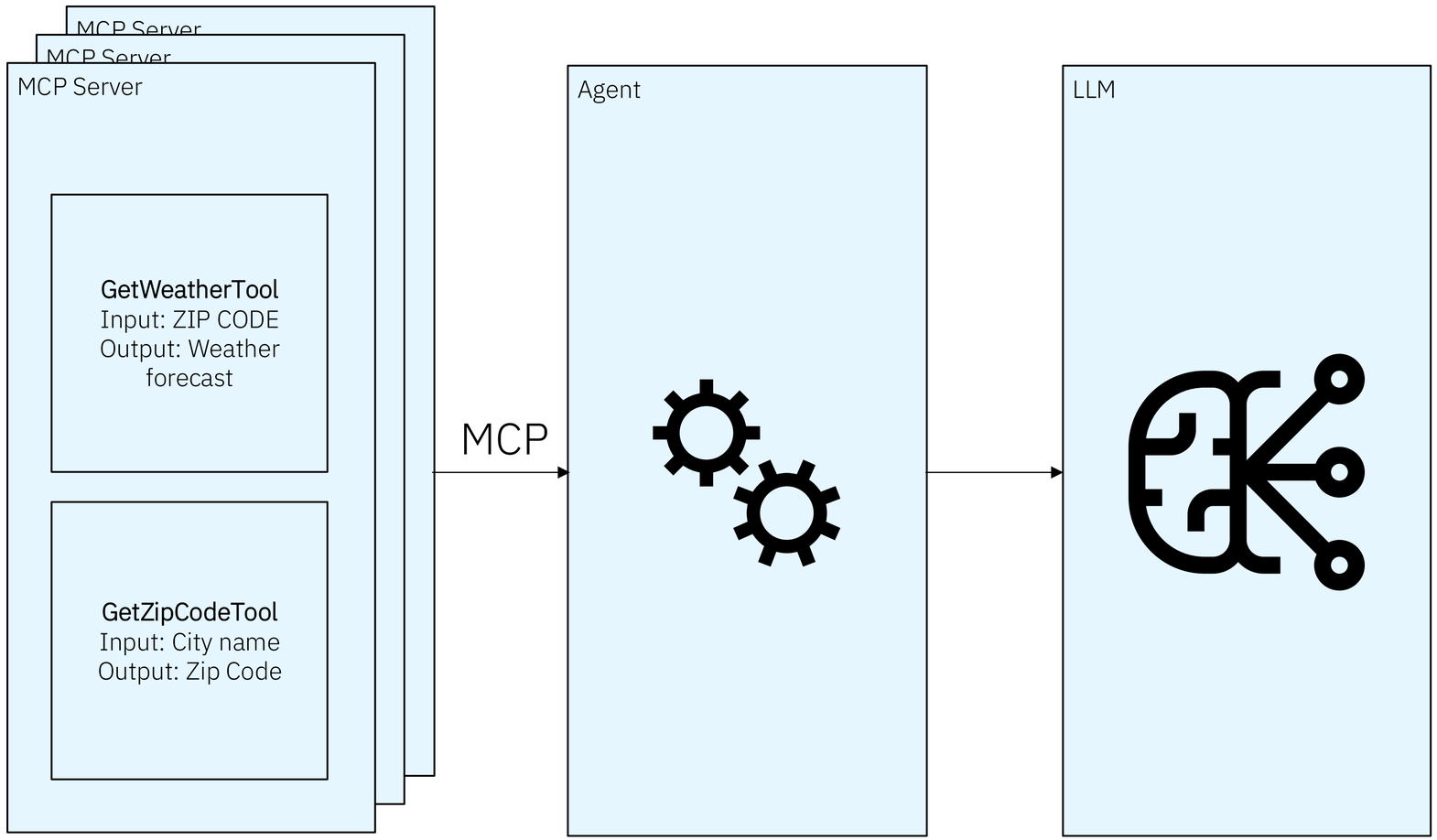
Model Context Protocol

MCP is used to dynamically wire tools into agents

This makes it easier to build an agent, and make it context-aware, by wiring in pre-existing off-the-shelf tools

Platforms like IBM Watsonx Orchestrate let you do exactly this

CICS TS 6.3 includes an embedded MCP server, to help us build context-aware agents for CICS TS

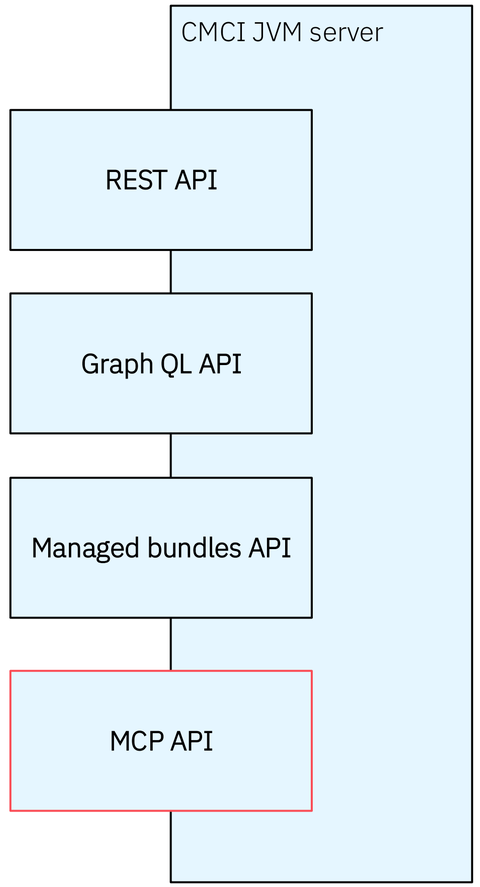


CICS MCP Server

CICS TS 6.3 has an embedded MCP server which provides context to agents about:

- System topology
- System configuration
- Transactions
- Programs
- Files
- more!

The MCP server is a new feature of CMCI JVM server, so does not require any additional infrastructure



CICS Transaction Sever Agents for Z

- Custom LLM-based agents, built with purpose for common CICS use cases
- Connects to MCP server as part of CICS 6.3 to provide live data about the CICS system and its configuration
- Built with SME knowledge from the CICS team to provide LLMs with some grounding information about the complexities of CICS
- Support for Granite, Llama and Spyre Accelerator cards on the z17 to keep all processing on-prem and on-box

Topology Agent

- Ask questions about CICS topologies, setup, inter-region connections, JVMServers, architecture, programming concepts and more!

- Uses a special index of data, built by the CICS team and formed on interviews with SMEs on key topics and concepts

Problem Determination Agent

- Uses live CICS data via MCP (in CICS 6.3) to diagnose causes of DFHAC class of message codes (transactions)

- Uses that live data to recommend a fix for the issue, all in one message along with other key data about the transaction and its related program

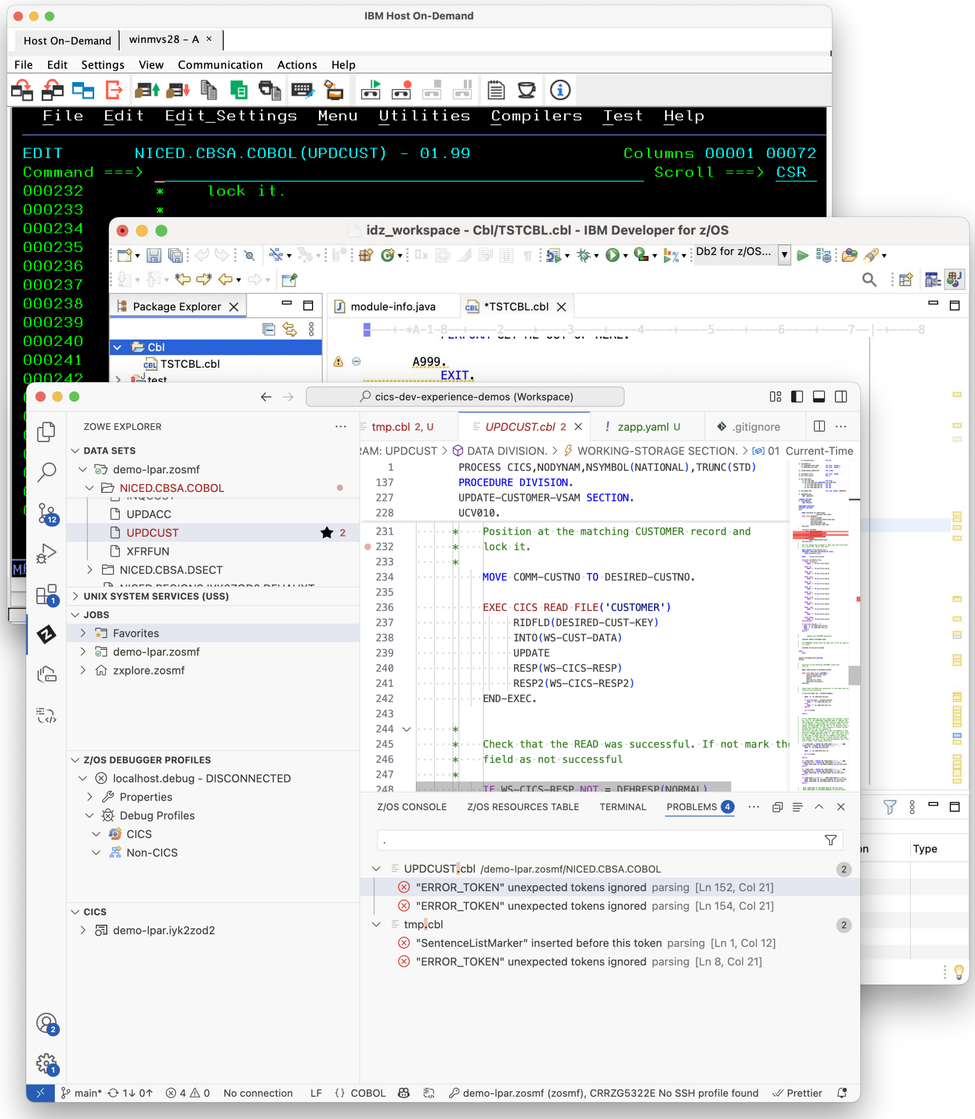
The details...

To find out more about CICS TS's MCP server, Agents, and to hear more from the CICS TS team on AI, consider these sessions:

LNL: AI Where It Matters Most: Agentic AI on IBM Z Speaker: Kye Maloy Monday 23rd Feb 12pm	CICS 6.3 is AI Ready! Speaker: Kye Maloy Monday 23rd Feb 3:45	An Introduction to Leveraging AI to Increase Productivity in CICS Speaker: Kye Maloy Wednesday 25th Feb 9:15

Developer Experience

CICS TS Application Development tooling choices



3270 (ISPF, CEMT)

- Traditional choice
- Keyboard, keyboard, keyboard!

Eclipse

- Mature, stable, feature-rich

Visual Studio Code (VS Code)

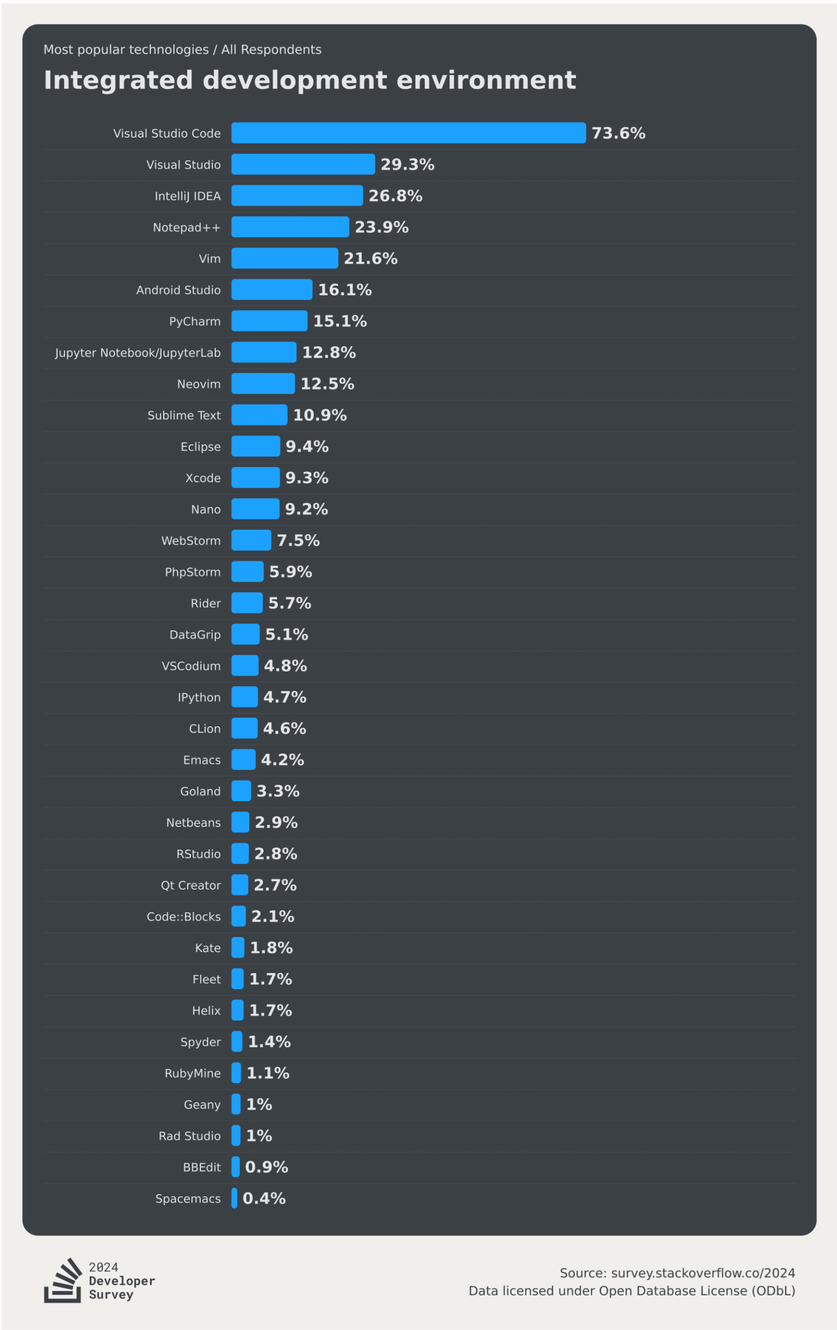
- The new(ish) kid on the block
- Lightweight
- Modular / extension-oriented

VS Code

In the wider industry, VS Code is far and away the most popular IDE

- Light-weight
- Flexible
- Extensible
- Fast
- Great UX
- Unique feature set

The developers you are hiring tomorrow, will already know how to use VS Code. They will want to be able to use VS Code to develop z/OS applications too.



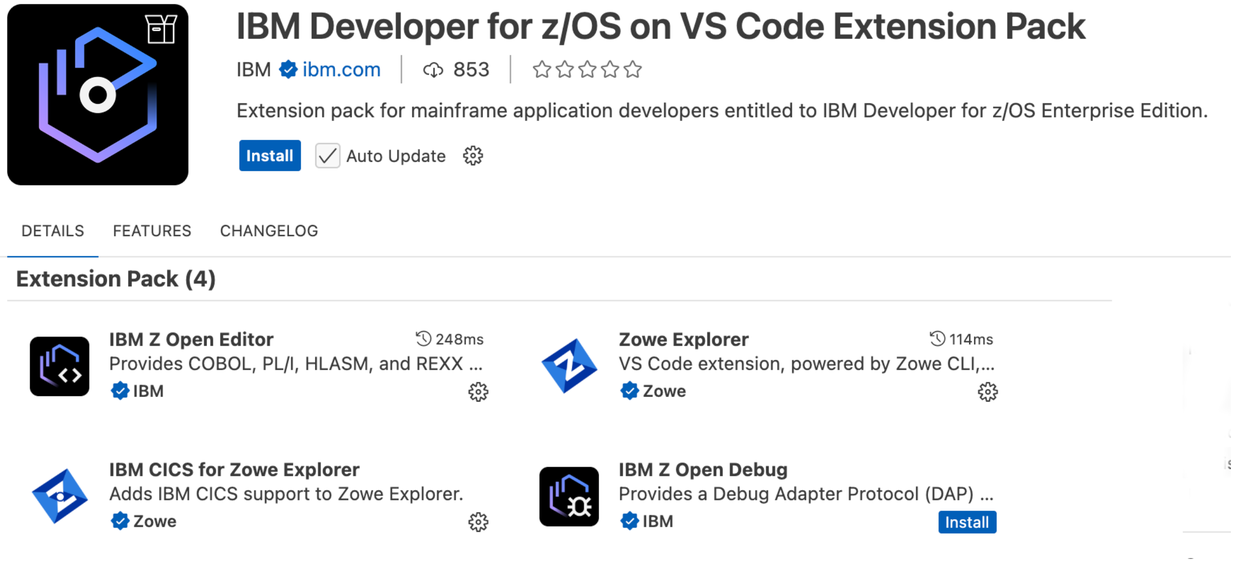
What's out there, and what have we been doing?

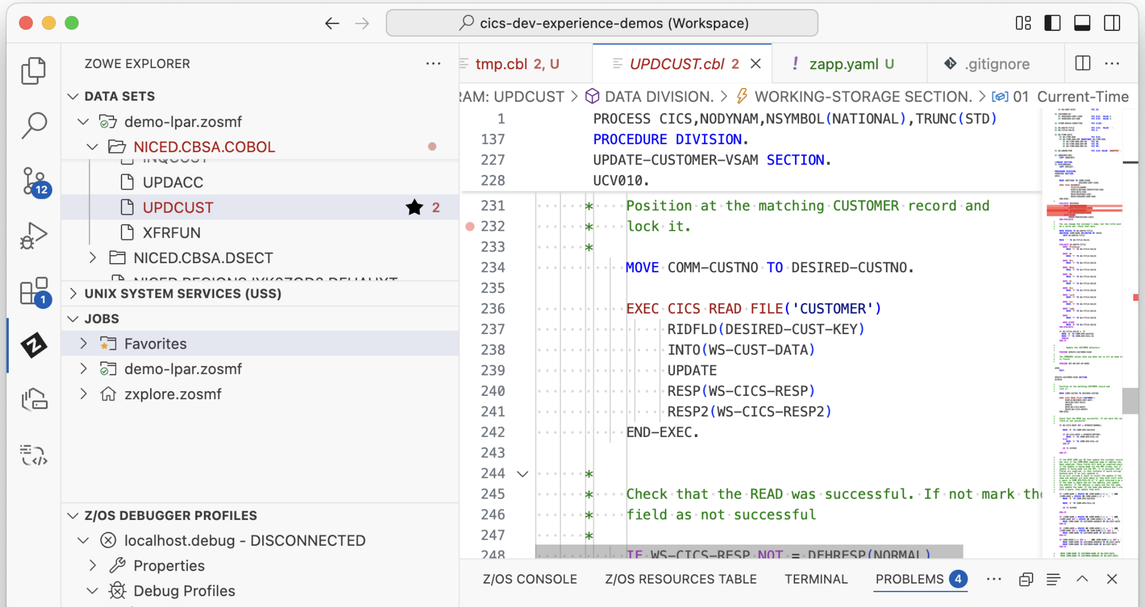
There's already a great set of extensions from IBM and Zowe for developing z/OS applications

We've been updating these extensions to make big improvements for CICS TS application developers

Particularly in the IBM CICS for Zowe Explorer extension

Lots is free to use. If you're already an IDzEE or ADfZ customer, you're entitled to premium features and support for IBM VS Code extensions too!





VS Code

Local file manipulation
Git tools

VS Code

Data sets, z/OS Unix, Jobs, z/OS Console
Profiles (connection details)

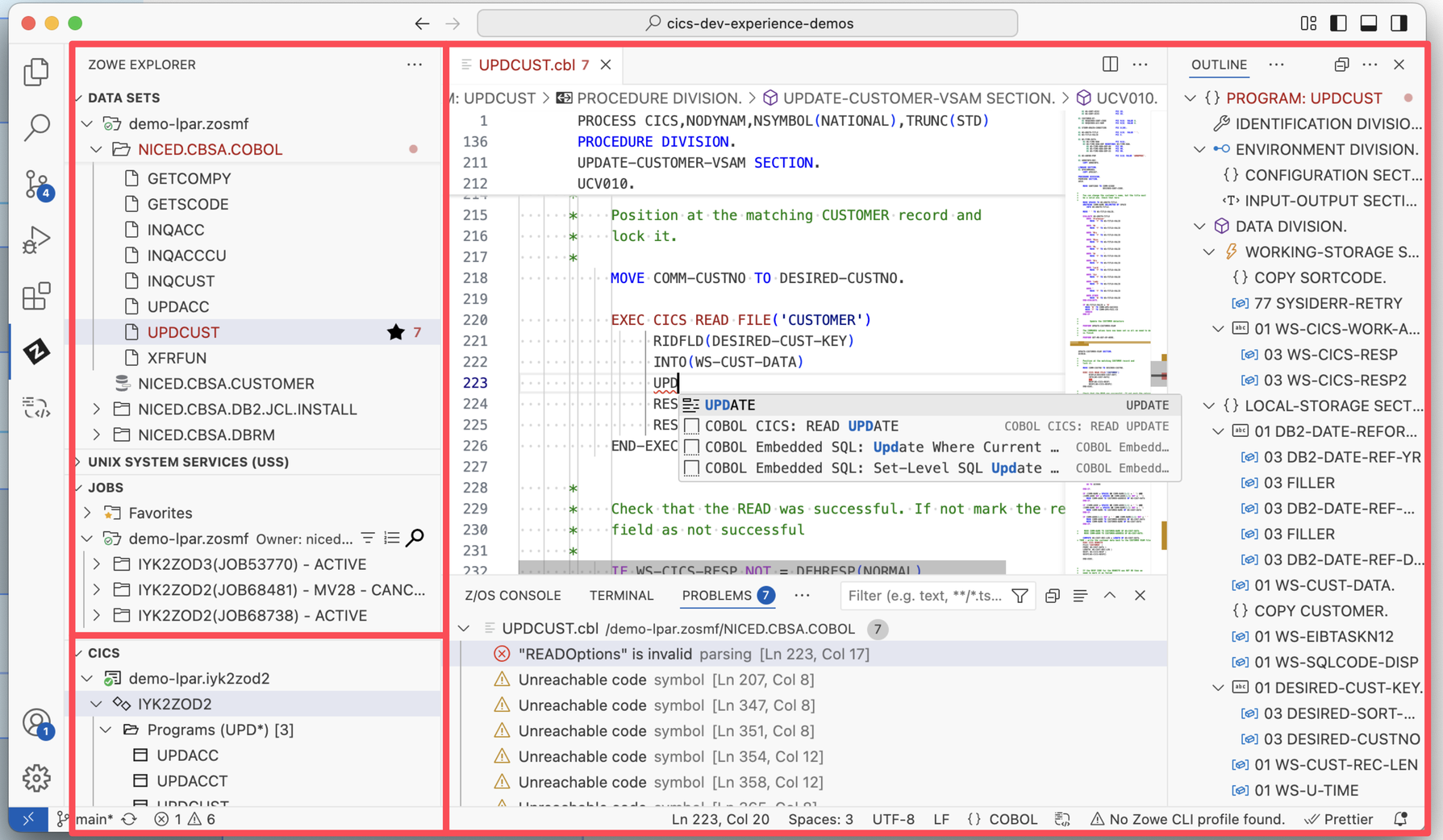
Zowe Explorer

Managing CICS resources in a CICS region

IBM CICS for Zowe Explorer

COBOL, PL/I, Assembler
syntax highlighting, code complete

IBM Z Open Editor

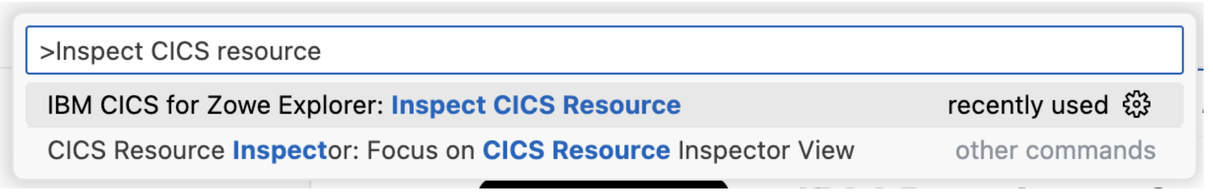


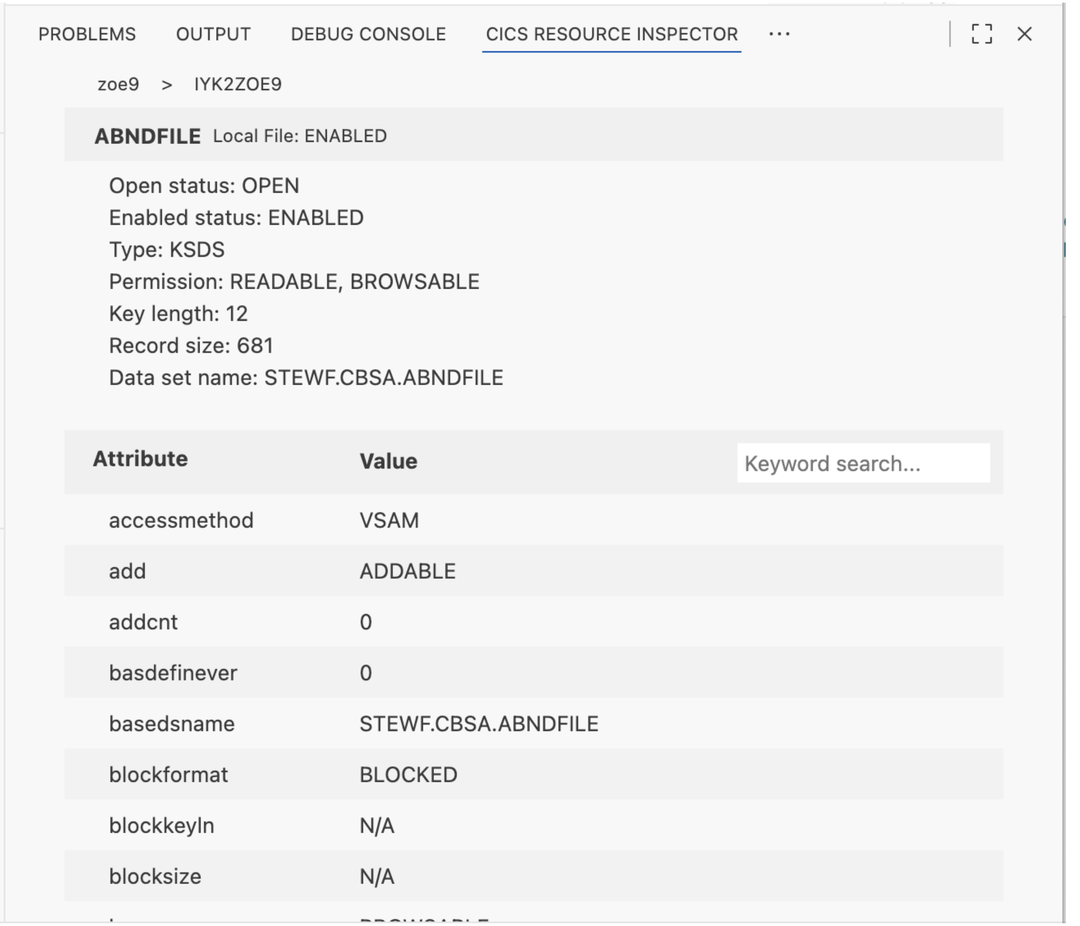
Our philosophy

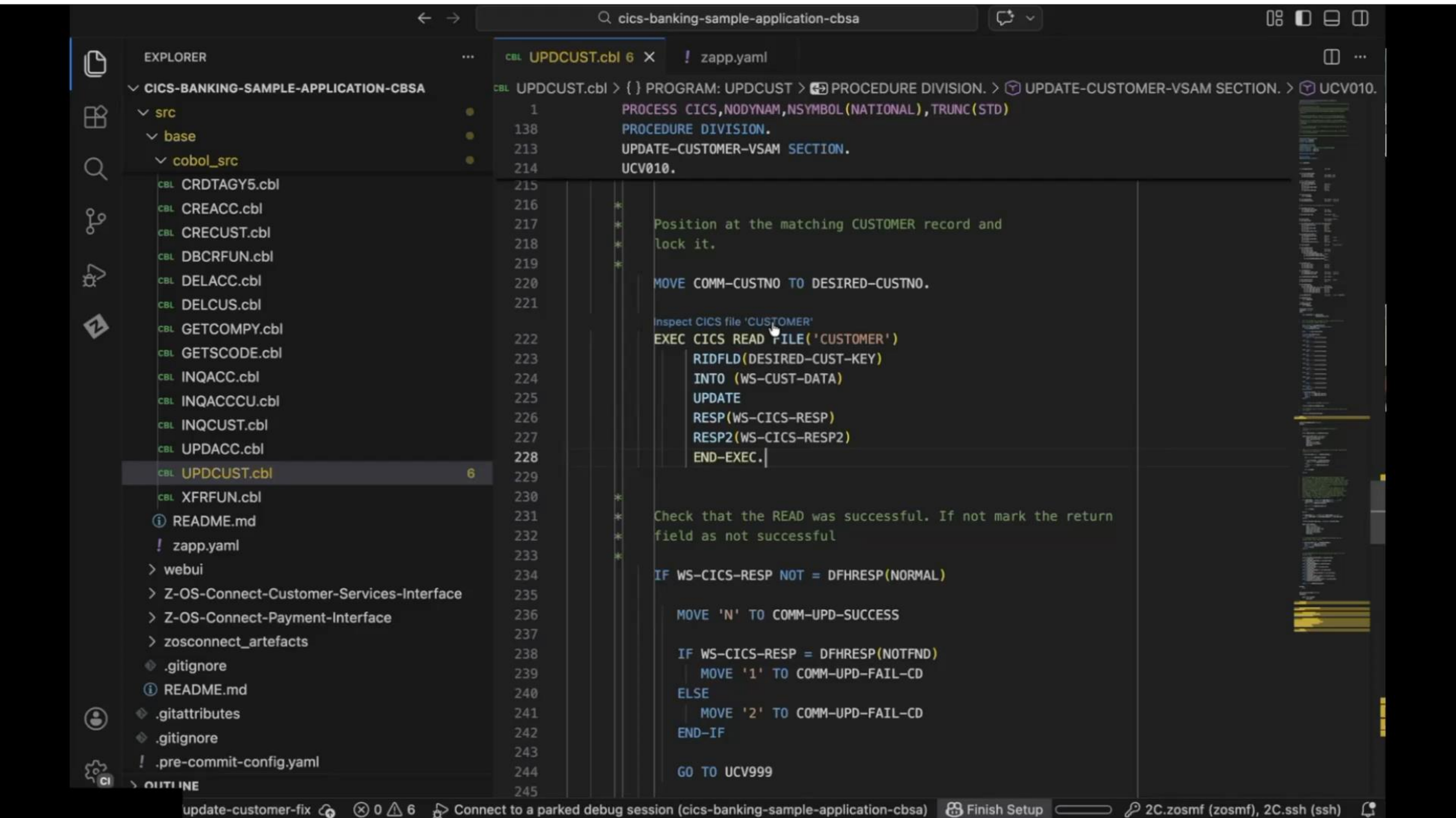
VS Code is for Application Developers – don't expect a CICS Explorer-like experience

Keep developers in the code – don't make them navigate to a different part of the UI when they need to answer CICS questions

Keyboard-friendly – VS code design philosophy is away from click-heavy wizards and towards lighter-weight keyboard-driven interactions. Try and use this to capture what keeps people coming back to 3270






cics-banking-sample-application-cbsa
EXPLORER
CICS-BANKING-SAMPLE-APPLICATION-CBSA
src
base
cobol_src
CRDTAGY5.cbl
CREACC.cbl
CRECUST.cbl
DBCRFUN.cbl
DELACC.cbl
DELCUS.cbl
GETCOMPY.cbl
GETSCODE.cbl
INQACC.cbl
INQACCCU.cbl
INQCUST.cbl
UPDACC.cbl
UPDCUST.cbl 6
XFRFUN.cbl
README.md
zapp.yaml
webui
Z-OS-Connect-Customer-Services-Interface
Z-OS-Connect-Payment-Interface
zosconnect_artefacts
.gitignore
README.md
.gitattributes
.gitignore
.pre-commit-config.yaml
OUTLINE
UPDCUST.cbl 6
zapp.yaml
UPDCUST.cbl > { } PROGRAM: UPDCUST > PROCEDURE DIVISION. > UPDATE-CUSTOMER-VSAM SECTION. > UCV010.
1 PROCESS CICS,NODYNAM,NSYMBOL(NATIONAL),TRUNC(STD)
138 PROCEDURE DIVISION.
213 UPDATE-CUSTOMER-VSAM SECTION.
214 UCV010.
215
216 *
217 * Position at the matching CUSTOMER record and
218 * lock it.
219 *
220 MOVE COMM-CUSTNO TO DESIRED-CUSTNO.
221
Inspect CICS file 'CUSTOMER'
222 EXEC CICS READ FILE('CUSTOMER')
223 RIDFLD(DESIRED-CUST-KEY)
224 INTO (WS-CUST-DATA)
225 UPDATE
226 RESP(WS-CICS-RESP)
227 RESP2(WS-CICS-RESP2)
228 END-EXEC.
229
230 *
231 * Check that the READ was successful. If not mark the return
232 * field as not successful
233 *
234 IF WS-CICS-RESP NOT = DFHRESP(NORMAL)
235
236 MOVE 'N' TO COMM-UPD-SUCCESS
237
238 IF WS-CICS-RESP = DFHRESP(NOTFND)
239 MOVE '1' TO COMM-UPD-FAIL-CD
240 ELSE
241 MOVE '2' TO COMM-UPD-FAIL-CD
242 END-IF
243
244 GO TO UCV999
245
update-customer-fix
0 6
Connect to a parked debug session (cics-banking-sample-application-cbsa)
Finish Setup
2C.zosmf (zosmf), 2C.ssh (ssh)

Configuration Simplification

Challenges facing the next generation of CICS system programmers

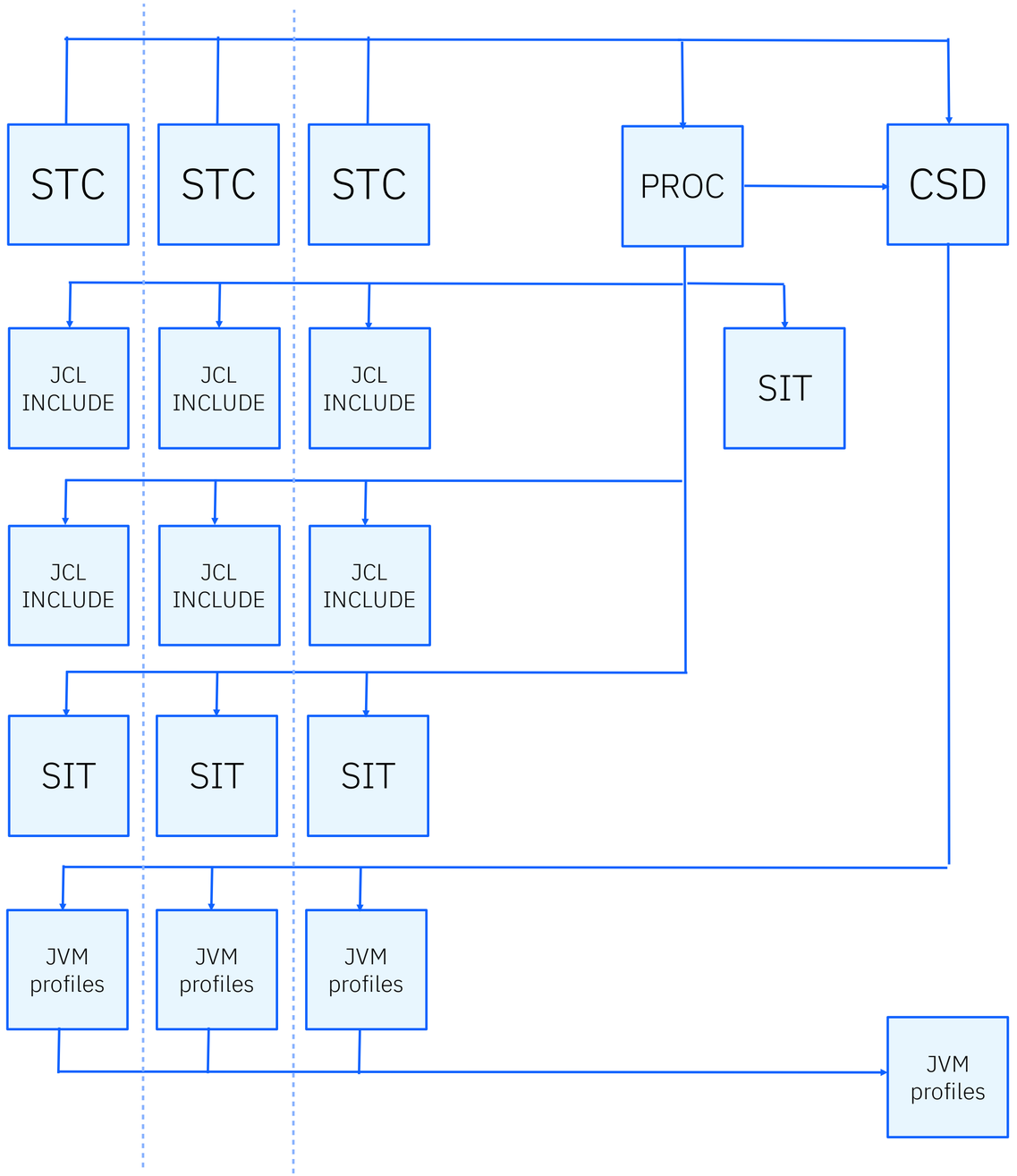
- Takes a long time for early tenure system programmers to become comfortable with the platform
- Not something most universities teach

AS-IS SCENARIO

Stan teaches Julian

CICS system programmer Stan is teaching his junior colleague Julian about how CICS regions are built, configured and managed.

He talks Julian through the complicated web of JCL PROCs, shared and region-specific CSD groups, SIT parameters and zFS configuration files that his organization have created.

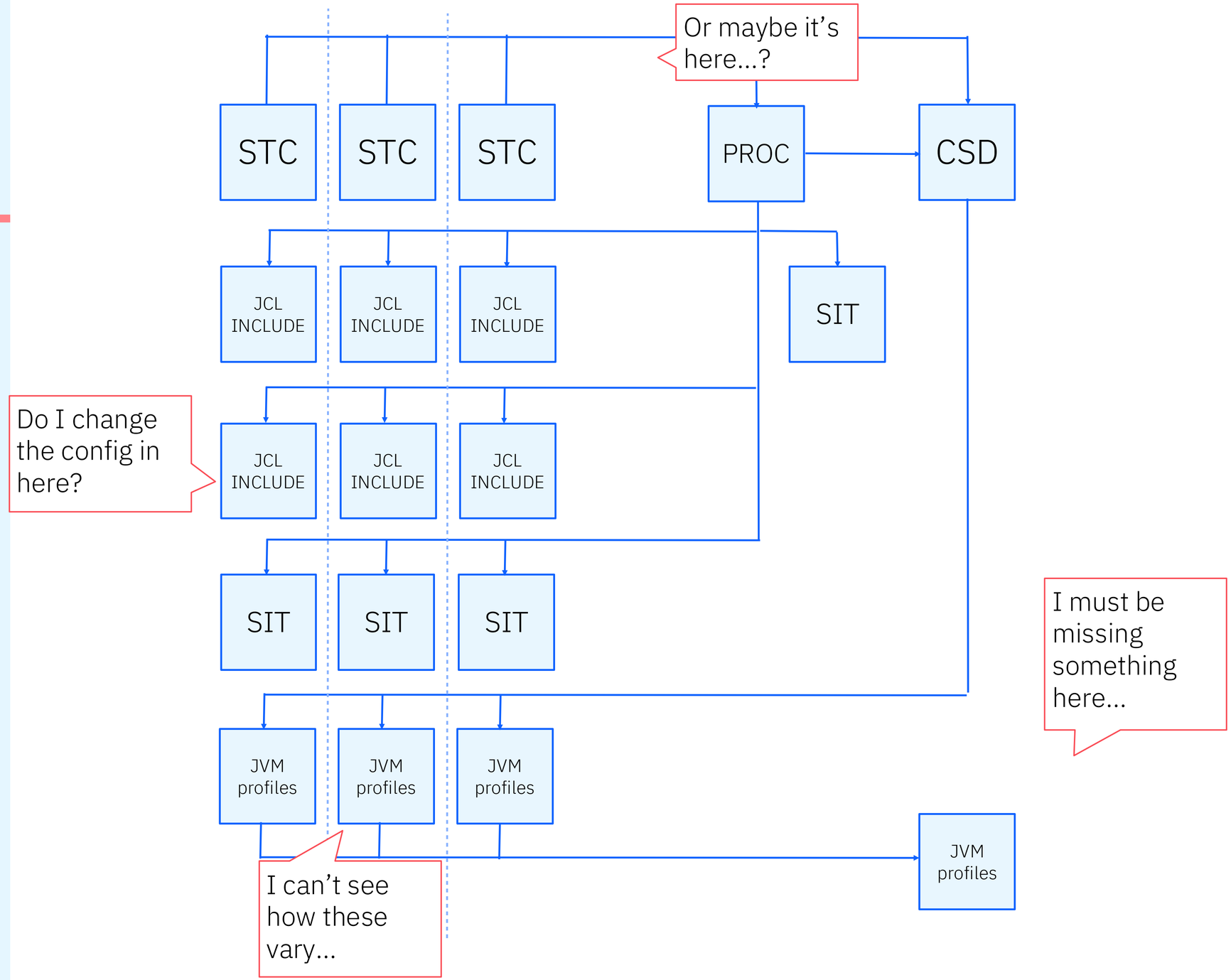


AS-IS SCENARIO

Complicated system

Julian finds it hard to understand the way the complex interrelated systems are put together, and as he's new to z/OS, all the concepts are new to him too.

He finds that he can't look at the system and understand or reason about what will happen at runtime — there is too much complexity locked up in the configuration.



What can we do about it?

“Simplify the configuration of CICS using a declarative configuration language”

1

2

CICS configuration as YAML

Configure CICS with YAML

- SIT parameters
- CSD content
- Data set allocation
- JVM profiles
- Programmatic extensions

Variable support

Validation and content assist in editor with a schema

Easy to store in a source code management repository – all the benefits of configuration as code

```yaml
# yaml-language-server: $schema=./cics_region_0.1.1.schema.json
cics_region:
  applid: "{{ vars.cics_region_applid }}"
  sysid: "{{ vars.cics_region_sysid }}"
  region_hlq: "{{ vars.cics_region_hlq }}"

  installation:
    dir: "{{ vars.uss_home }}"
    svc: 217
    srbsvc: 218
    data_sets:
      cics:
        hlq: "{{ vars.cics_hlq }}"
        sdfhlic: "{{ vars.cics_lic }}"
      cpsm:
        hlq: "{{ vars.cpsm_hlq }}"

  csd:
    content:
      - script: cbsa.csdup.txt
        type: csdup_script
      - script: debug.csdup.txt
        type: csdup_script

  extensions:
    cics_cmci:
      provider: JVMSERVER
      port: "{{ vars.cics_region_cmci_port }}"
      authentication: AUTOMATIC
      ssl: "NO"
```

CICS configuration as YAML

YAML configuration is translated into configuration files and data sets that CICS TS already understands – no runtime changes to support this

USS tool zconfig performs the translation

Works with all in-service releases of CICS

Configuration is declaratively applied to the system, just change the configuration document and re-run, for any config change

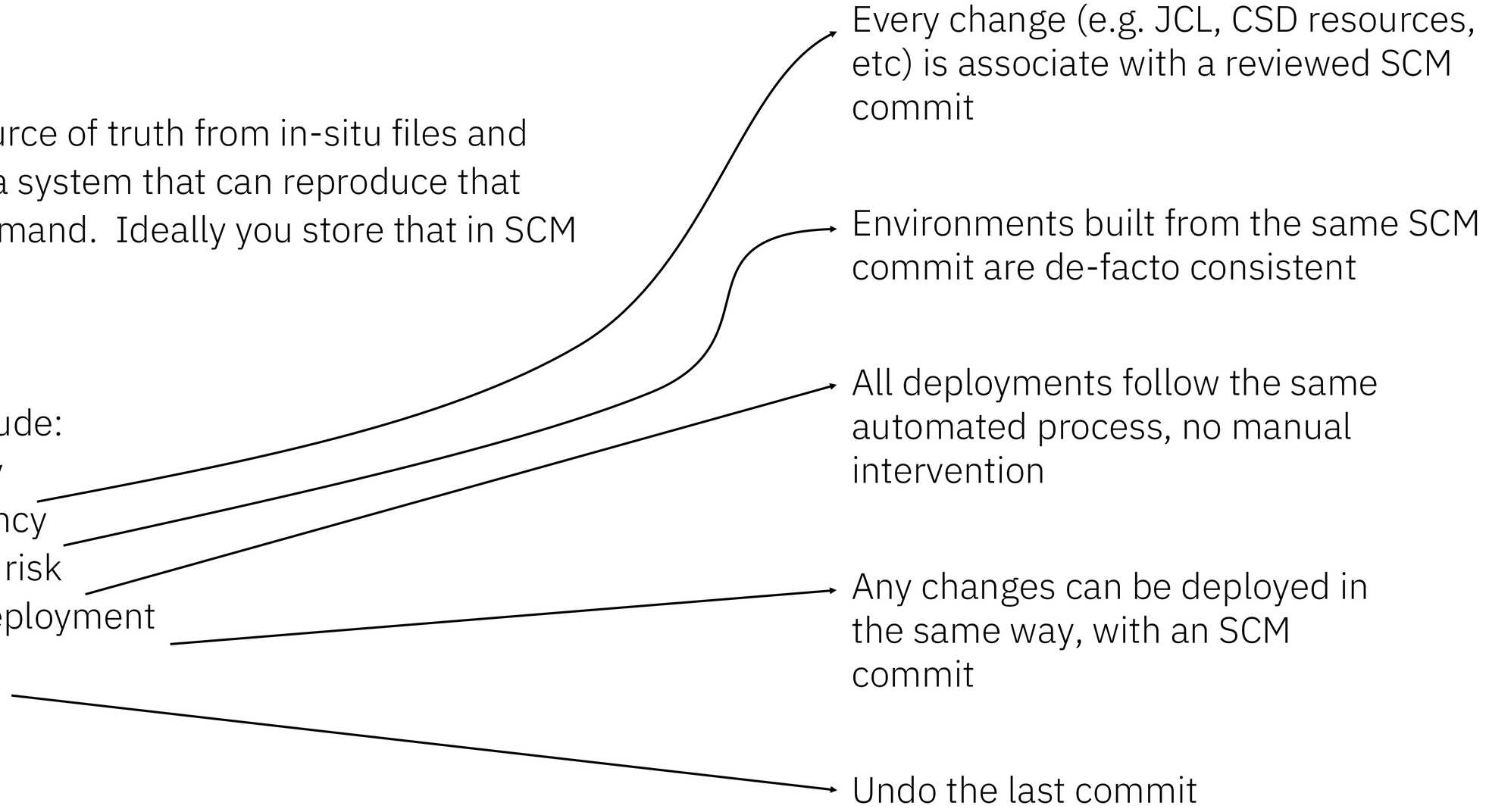
```yaml
# yaml-language-server: $schema=./cics_region_0.1.1.schema.json
cics_region:
  applid: "{{ vars.cics_region_applid }}"
  sysid: "{{ vars.cics_region_sysid }}"
  region_hlq: "{{ vars.cics_region_hlq }}"

  installation:
    dir: "{{ vars.uss_home }}"
    svc: 217
    srbsvc: 218
    data_sets:
      cics:
        hlq: "{{ vars.cics_hlq }}"
        sdfhlic: "{{ vars.cics_lic }}"
      cpsm:
        hlq: "{{ vars.cpsm_hlq }}"

  csd:
    content:
      - script: cbsa.csdup.txt
        type: csdup_script
      - script: debug.csdup.txt
        type: csdup_script

  extensions:
    cics_cmci:
      provider: JVMSERVER
      port: "{{ vars.cics_region_cmci_port }}"
      authentication: AUTOMATIC
      ssl: "NO"
```

Why configuration as code?

Move the source of truth from in-situ files and data sets to a system that can reproduce that config on-demand. Ideally you store that in SCM (e.g. git)

Benefits include:

- Audibility
- Consistency
- Reduced risk
- Faster deployment
- Back-out



The details

Hear more about this topic this week:

- Which elements of CICS config are supported
- How they're configured in YAML
- Editing experience
- Demos of using this tool to configure some real CICS regions

Sign up to our closed beta:
https://ibm.biz/zconfig-beta

Simplifying CICS Configuration - Harness the Power of Code
Speaker: Stew Francis
Wednesday 25th Feb 3:45

Modern Languages

Support for Java 21

- Backports to 5.5, 5.6, 6.1, 6.2
- Minimum Java version 17
- Default codepage UTF-8 (already true for Liberty apps)
- JCICS String API usage unaffected
- -Dfile.encoding=COMPAT
- CICS Explorer for Aqua 3.4 supports Java 21 for development

- Jakarta EE 10
- Microprofile 6.x
- Spring Boot 3

New with CICS TS 6.3, also backported to 6.1, 6.2

New JCICS API for WRITE OPERATOR, VERIFY PHASE

JCICSX compatibility for Jakarta EE 9 / 10

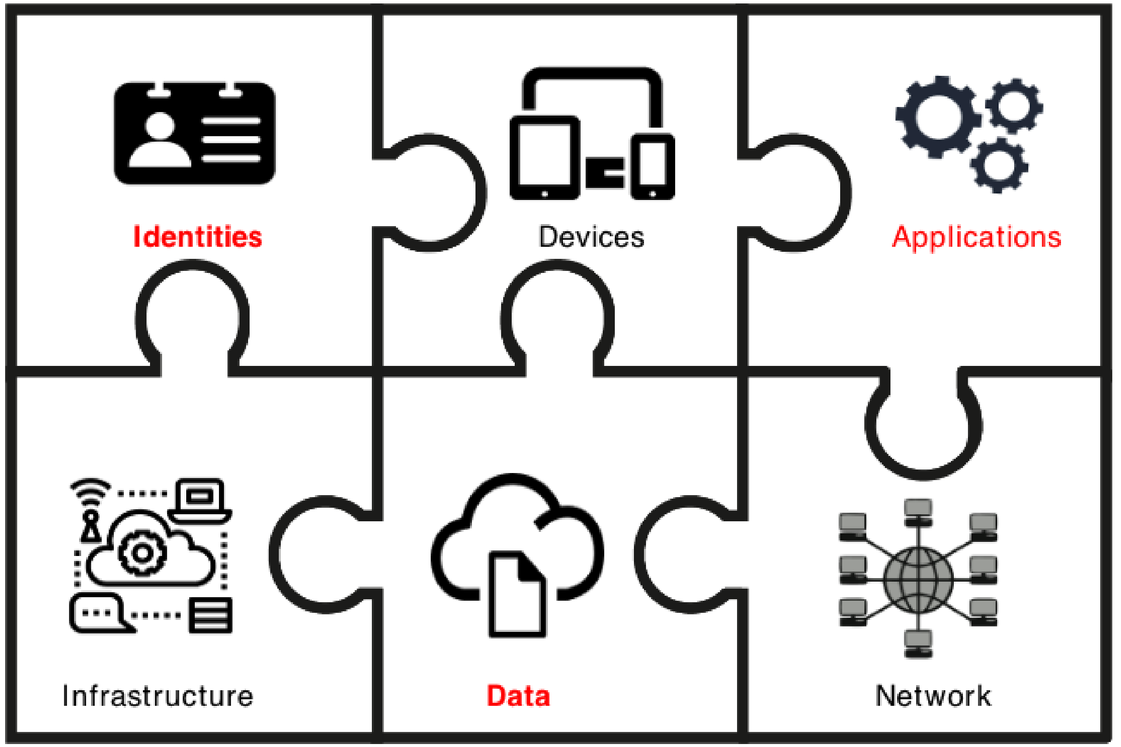
Zero Trust

Focus on protecting resources not perimeters

Enable the **right user**, to have the **right access**, to the **right data**, for the **right reasons**.

Never trust, always verify

Assume the bad actor is already present and continuously monitor.



Security

Security discovery journey started in CICS TS 6.2, as part of initiative to help implement

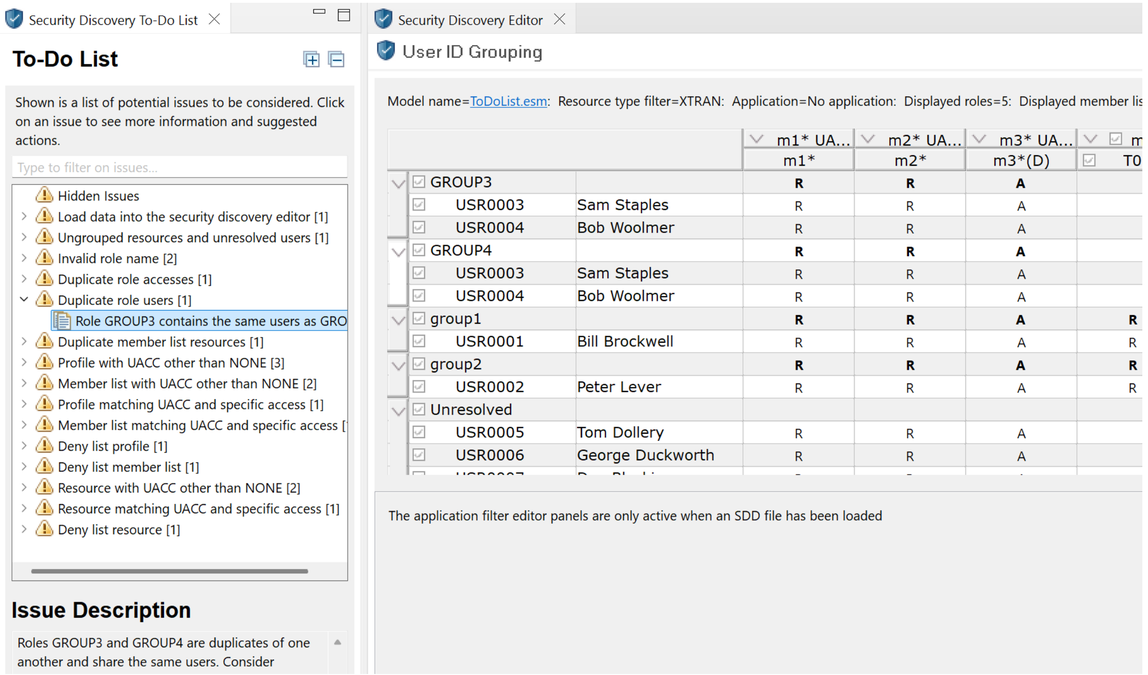
- Export security definitions from your security database to YAML
- Import YAML definitions into CICS Explorer for analysis
- Collect instrumented security logs from your running application
- Compare the security definitions with the data, and find out what access people *really* need

New in 6.3:

- To-Do List populated with suggested actions from automated analysis
- Report of changes made in editor

Also of note:

- SECURETLS SIT parameter
- AT-TLS more widely supported



[CICS TS 6 - Journey to Zero Trust & DORA Compliance]

Speaker: Lewis James

Tuesday 24th Feb 9:15

[Security Discovery in CICS]

Speaker: Lewis James

Thursday 26th Feb 9:15

Foundation

Things we didn’t already cover:

Policies configured with an action to issue a message will also send a message to the system log

New policy rules for APPC (LU6.2) connection and JVM server status

Other Sessions on the CICS track

Beyond Monitoring: Unleashing Automated Operations for CICS Diagnostics
Speakers: Teodoro Novo, Flavio Yano
Tuesday 24th Feb 8:00am

Profiling a CICS Transaction Using Trace
Speaker: Ezriel Gross
Tuesday 24th Feb 10:30am

CICS Debugging Essentials
Speaker: Ehimare Uiyoshioria
Tuesday 24th Feb 2:30pm

CICS Nuts, Bolts and Gotchas
Speaker: Ehimare Uiyoshioria
Wednesday 25th Feb 1:15pm

CICS TS 6 - Performance Update
Speaker: Lewis James
Wednesday 25th Feb 2:30pm

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