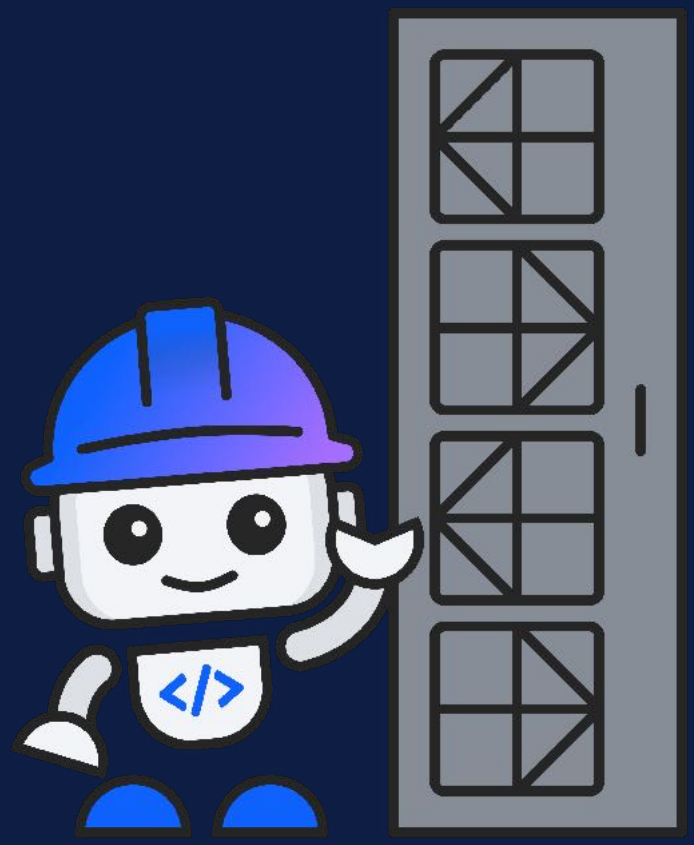
IBM Bob

IBM Bob Premium Package for Z

Common Polska Conference, 20.10.2026
Maciej Zrobek
IBM Z SW Technical Sales
email: maciej.zrobek@pl.ibm.com



A special message for PK

AI was heavily used

and occasionally asked to

“MAKE IT LOOK BETTER”

during the making of this presentation.

No AI was harmed.

Its patience, however,

was thoroughly tested.

IBM watsonx Code Assistant for Z
->
IBM Bob Premium Package for Z

AI-powered development for IBM Z
Where do we come from?
Where do we go?

Maciej Zrobek
maciej.zrobek@pl.ibm.com
IBM Z Software Technical Sales

Common Polska Conference
Gdynia, 18.05.2026



Now we can tell the rest of the story ...

THE CHALLENGE

Mainframe runs the business — and modernization can't wait

Billions

of transactions every day

Decades

of proven reliability

Hybrid

No rewrite

Slow development

Lengthy release cycles slow business responsiveness.

Risk amplification

Changes cascade across batch, online and data.

Knowledge gaps

Business logic buried in code, thin documentation.

Skill shortage

z/OS experience retiring faster than it is replaced.

Don’t want to get boring -

you know all of this too well

IBM

THE SHIFT

Agentic AI — with decades of IBM Z expertise built in

AI has assisted with code for years, but in fragmented, isolated tasks. Agentic workflows now operate with far greater context — coordinated, automated and governed modernization.

IBM Bob Premium Package for Z

Extends IBM Bob with deep mainframe expertise
Enhances watsonx Code Assistant for Z
Plan, execute, validate and govern across the SDLC — in the IDE

Other AI tools

Syntax-level help, no architectural understanding, blind to middleware and data relationships.

IBM Bob Premium Package for Z

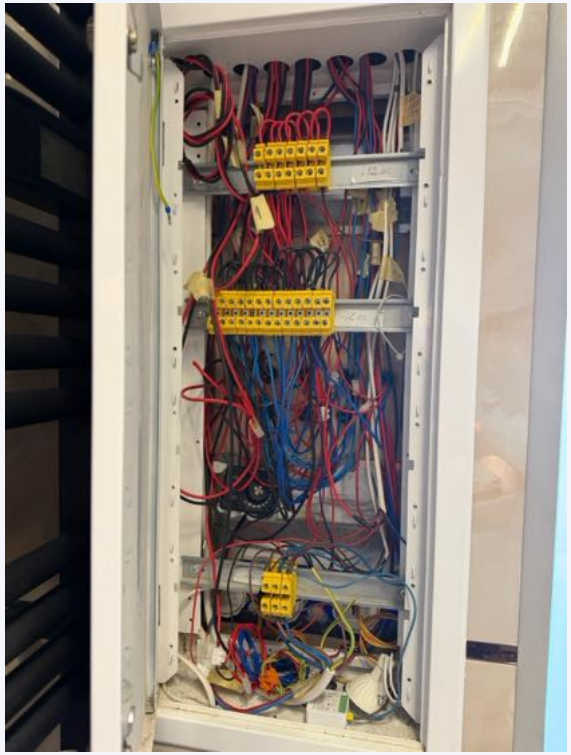
Knows how applications, data and transactions interact, and how complex Z workflows are orchestrated.

True modernization requires architectural understanding — not autocomplete.

MAINFRAME COMPLEXITY

Mainframe applications require purpose-built tooling

IBM Z applications are complex, mission-critical, and often involve millions of lines of code, requiring platform-awareness and a deep understanding of Z technologies



Large, highly intertwined applications

Mainframe environments are massive and tightly interconnected, making change impact difficult to assess without clear boundaries

Vague, cryptic variables

Mainframe applications contain vast numbers of variables with embedded semantics that cannot be fully documented or interpreted in isolation

Hidden business logic and intent

Critical business rules are embedded across decades of complex code, making application intent and behavior difficult to understand without contextual understanding

© 2026 IBM Corporation

HOW IT WORKS

Intelligent mainframe developer experience

Your IBM Z–savvy development partner comes with specialized mainframe modes that understand intent, anticipate impact, and accelerate workflows with collaborative automation

Z Architect mode

Plan and analyze mainframe applications to identify how to enhance or modernize applications before implementation

Examples prompts:

“What is the impact of changing this business rule in my application?”

“I want to change the data type of Policy number in the system from numeric to alphanumeric”

Z Code mode

Explain, generate, and modify mainframe application code; refactor monolithic code to reduce complexity; transform code to modern languages

Examples prompts:

“Create a new COBOL program that does X using my coding standards”

“Add an electric-vehicle indicator column to the Motor Policy Table and update all dependent programs to support the new field.”

Benefits:

Boost Productivity:
Innovate faster by offloading routine tasks to AI

Accelerate Delivery:
Achieve faster time to value with goal-driven automation

Improve Quality:
Identify issues early with continuous guidance

© 2026 IBM Corporation

KEY USE CASES

From enterprise understanding to delivered change

Understand applications

Metadata across hundreds of programs gives architecture-aware, end-to-end visibility.

Pre-change impact assessment

Dependency and architecture analysis reveals how a change ripples through the system.

Documentation at scale

Business-aware explanations and generated documentation, with rules extracted from legacy code.

Accelerate application testing

Automated tests generated from application metadata, mapped to functional entry points to catch defects before production.

Execute complex coding tasks

Generate, modify and transform code to enterprise standards via an agentic chat interface.

Incremental refactoring

Modernize selected parts of the application and validate each step — no full rewrite.

Optimize, debug, improve quality

Prioritized insights and automated issue resolution speed higher-quality releases.

Data dictionary context

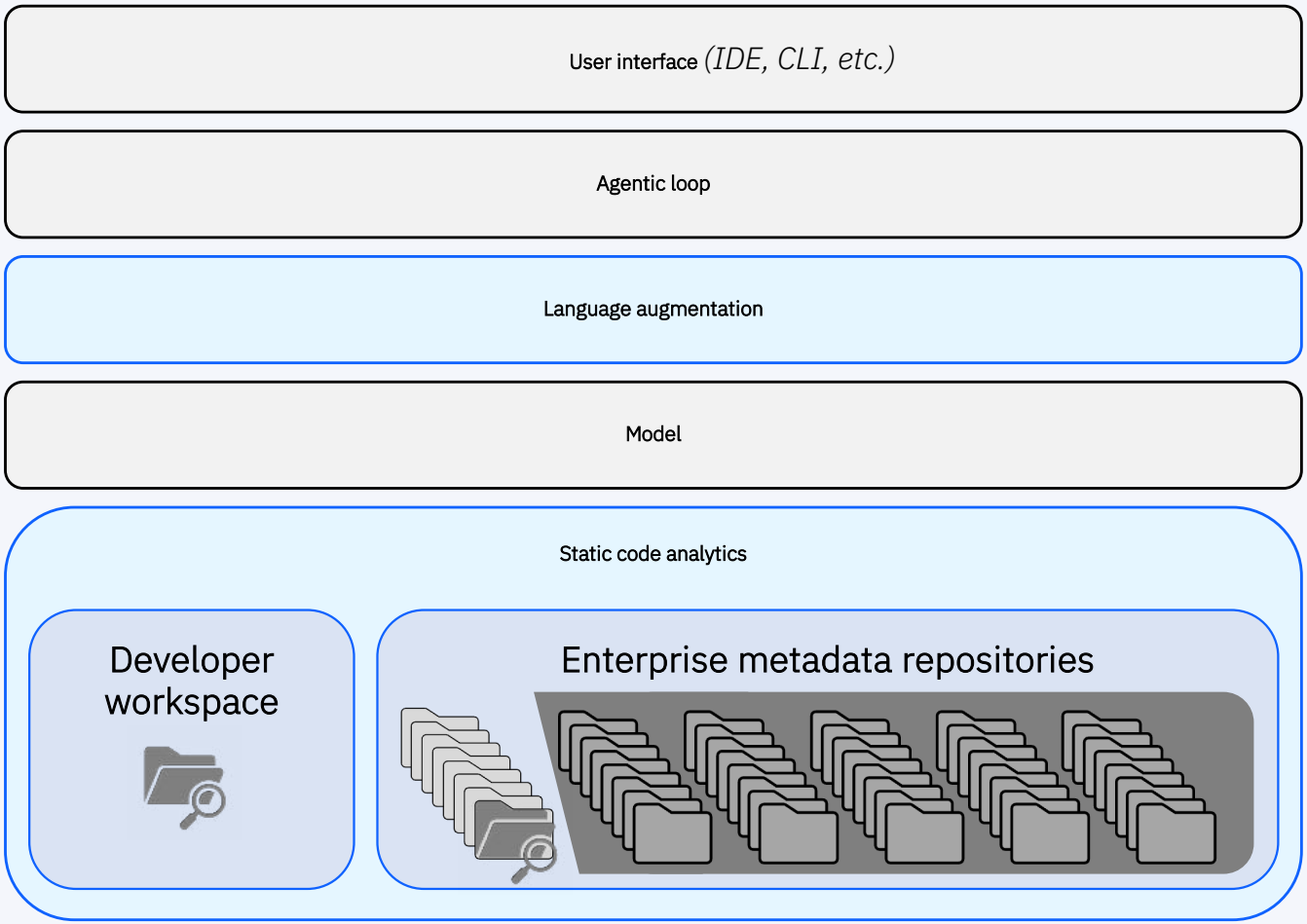
Enterprise data dictionaries let Bob read variables and abbreviations with the right context and explain true business purpose.

IBM

FRAMEWORK

IBM Bob with Premium Package for Z

Framework overview



Interaction layer (user, event, context)

SDLC Support (specify, plan, code, test, fix, deploy)

IBM Z Alignment (IBM COBOL, middleware platform)

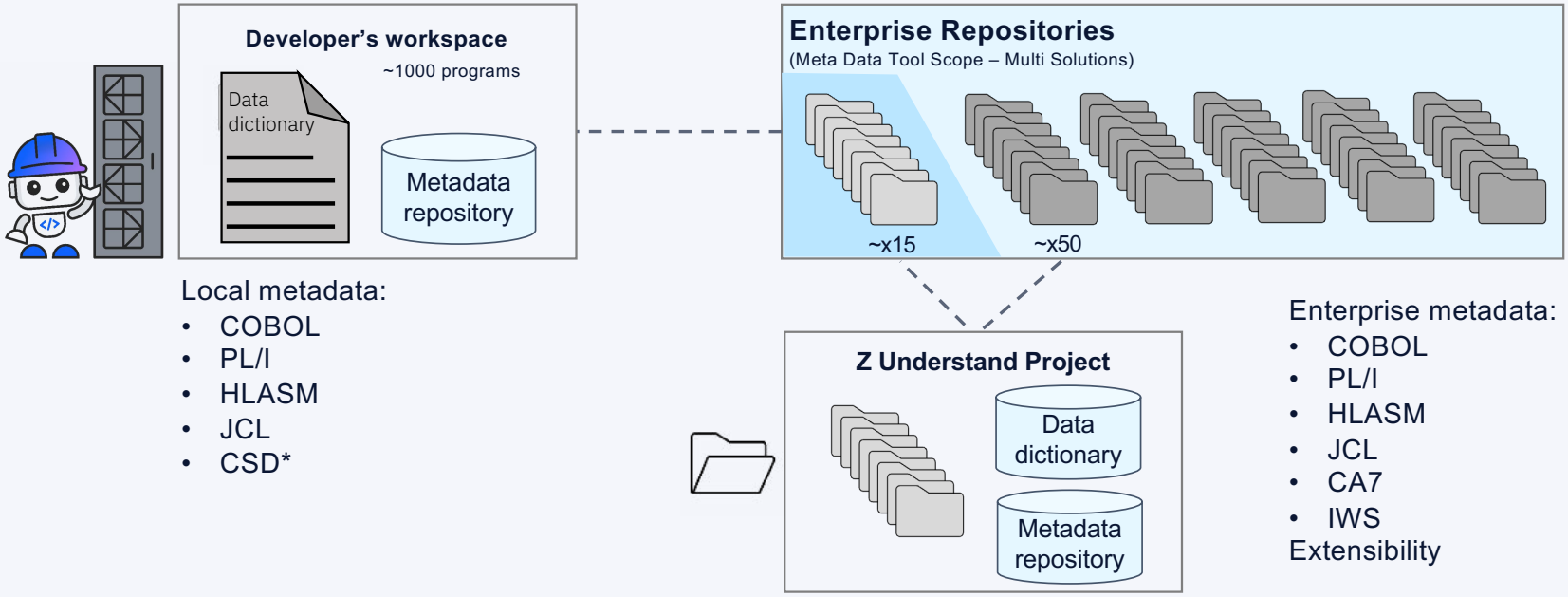
Selected Models (Frontier, Open Weight, IBM)

Enterprise Scale Understanding (data dictionary, business rules, control flow, code artifacts)

Blue boxes indicate Premium Package for Z additions

METADATA SCOPE

Scope of Z Understand project and developer's workspace



Local metadata:
- COBOL
- PL/I
- HLASM
- JCL
- CSD*

Enterprise metadata:
- COBOL
- PL/I
- HLASM
- JCL
- CA7
- IWS

Extensibility

** Roadmap item*

Recommendation is to keep local workspace under 1K artifacts

Terminology:
- The Enterprise solution landscape is comprised of multiple solutions.
- Each solution combines multiple applications.

Example:

• Enterprise	= Insurance Company IT	(50 Solutions)
• Solution	= Car Insurance Domain	(15 Repos)
• Application	= Claims Processing App	(1k Artifacts)

© 2026 IBM Corporation

ACCURACY

Why metadata matters?

Benchmarking across 30 impact analysis questions on Z applications demonstrates a significant gap in accuracy.
AI achieved 30% accuracy, while Bob Premium Package for Z has the real data

SQL logic is aligned with ground truth, enabling **accurate**, **validated insights** through **NL2SQL** (Natural Language to SQL)

Example prompt: *Which SQL tables are accessed but never updated anywhere?*
Correct answer: None

AI & agentic loop
Lists incorrect tables

GLOBAL_PARMS
INTERCONNECT_XREF
SLOG
CN_FILES_TRACKING
CUSTOMER_FILENAME
OUTBOUND_FILENAME
MQ_TO_CN_TRACKING
CN_TO_MQ_TRACKING
MQ_APPS_TRACKIN

Bob Premium Package for Z
Accurately detects **no tables**

Based on the analysis of the codebase, ALL 57
SQL tables have both read and write operations.
There are NO tables that are only read and never updated.

Key Findings:
Total tables analyzed: 57
Tables with only writes (no reads): 0
Tables with only reads (no writes): 0
Tables with both reads and writes

Example prompt*: List paragraphs in NET750*
Correct answer: 503 Paragraphs

AI & agentic loop
Finds 211 paragraphs

211 paragraphs

Bob Premium Package for Z
Finds **503** paragraphs

Found **503 paragraphs in NET750**

(ProgramID: 782, COBOL program). The paragraphs span from line 2175 to line 15099 and include….

Other example prompts:

"Show the top 5 most frequently accessed SQL tables."
"Identify programs that are not called by any other programs."
"Detect recursive program calls across the application."
"List all CICS transactions."
"Identify variables that are written but never read in Program A."
"Find all programs that read and write the same SQL table."

© 2026 IBM Corporation

EFFICIENCY

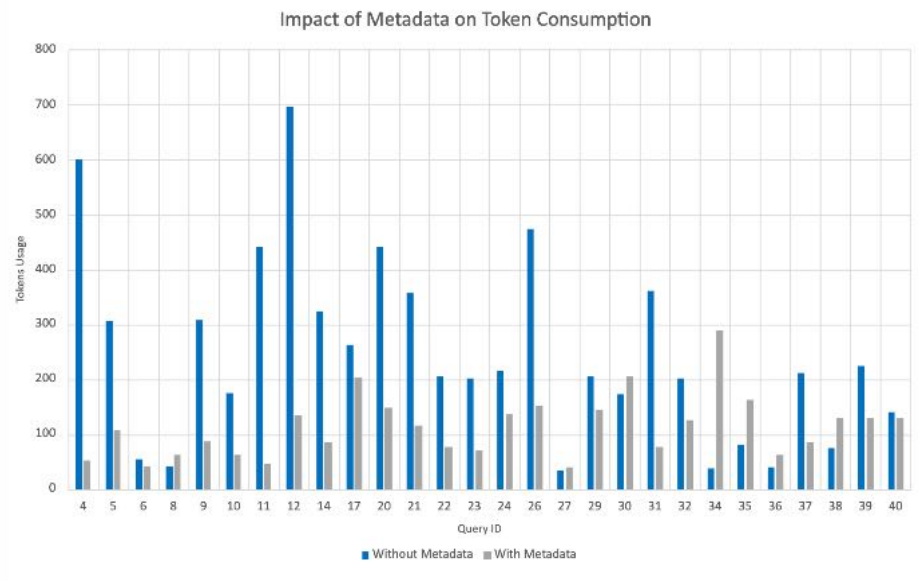
#zdevx-sdlc-3q-enablement

Bobcoin reduction with zUnderstand metadata

Since IBM Z applications typically involve large, complex codebases, the Bobcoin consumption may be larger than when working with non-IBM Z applications.

Metadata provides critical business context that helps the model understand application-specific terminology, structures, and relationships, enabling more efficient and accurate analysis while reducing unnecessary token consumption.

Benchmarking across 40 impact analysis questions on the Z application demonstrates a **significant reduction in token usage.** On average, **metadata decreased token consumption by 54%**, from 246K tokens to 113K tokens.



Example prompt: *list paragraphs in NET750.*

Without metadata	*With metadata*
600k Tokens Used	**52K Token Used**

Example prompt: *Which SQL tables are accessed but never updated anywhere?*

Without metadata	*With metadata*
696k Tokens Used	**134K Token Used**

Exposing static code analytics metadata through tool interfaces ensures response accuracy while optimizing token consumption across most use cases.

© 2026 IBM Corporation

Note: This is an IBM test and customer results may vary.

DOCUMENTATION

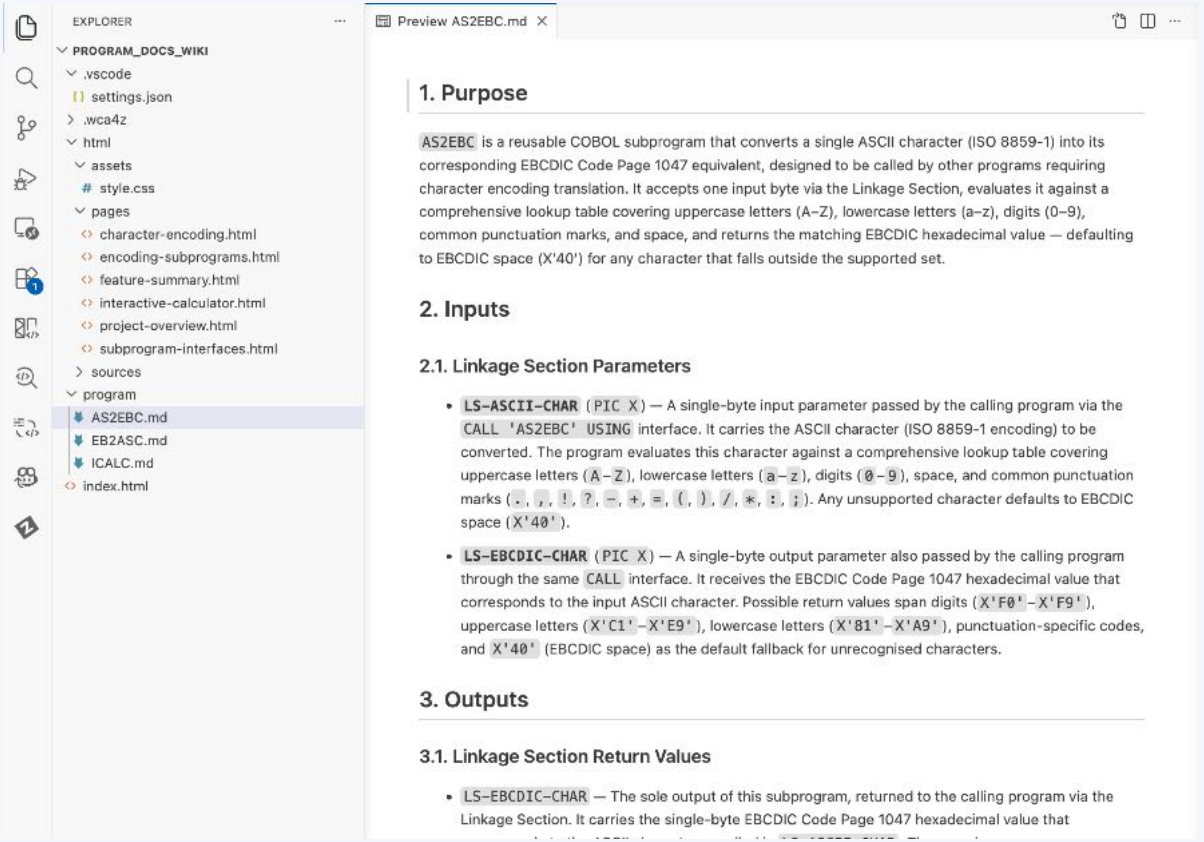
Documentation Generation

Static analysis, semantic enrichment, and controlled prompting produces **structured**, **business-aware documentation** across the entire application.

It delivers consistent, modernization-ready outputs at the program level (micro-view) and application level (macro-view).

Key Benefits:

- Scalable, deterministic, consistent outputs
- Metadata-driven visibility across programs and dependencies
- Integrated business rule extraction
- Continuously updated, enterprise-ready documentation



© 2026 IBM Corporation

DOCUMENTATION

Documentation & Application Wiki

Scaling code explanations to program & application documentation including an Application Wiki

Generate in the Bob IDE & Z Understand Container

Available for **COBOL, PL/I, and HLASM***

Option 1 - Bob IDE
uses local scanner metadata for a few to a few hundred programs - triggered via prompt and workflows in the Bob IDE

Option 2 - zUnderstand Container delivers full application documentation via the container scanners – triggered via CLI command

— **structure stays consistent** across both & include **Bulk Business Rule Identification**

Application Wiki

Turns an entire application into one interactive **wiki** automatically — no manual documentation effort

Self-service understanding: navigable by area with drill-down, so teams find answers without tracking down an expert

Business analysts grasp logic and business rules without reading code — bridging the business–IT gap

Accelerates **onboarding and modernization** — new developers get productive in days, not months

*no custom macro support

© 2026 IBM Corporation

On-demand skills for COBOL, PL/I, HLASM, JCL, REXX, CICS, Db2, IMS and MQ work

CODE GENERATION

`cobol-code-generation`

Mandatory COBOL syntax rules and post-generation validation for any new or changed COBOL source in Z Code mode.

CODE GENERATION

`hlasm-code-generation`

The same guardrails for HLASM/Assembler: syntax guidelines and validation for generated or modified Assembler source.

PROGRAM UNDERSTANDING

`data-dictionary-management`

Builds and maintains COBOL/PL/I data dictionaries: scans programs, extracts variables, generates descriptions, supports user review.

ARCHITECTURE & ANALYSIS

`impact-analysis`

Structured impact reports for COBOL, PL/I, JCL, Assembler and REXX changes: dependencies, risks and change-propagation diagrams.

ARCHITECTURE & ANALYSIS

`impact-analysis-field-expansion`

Two-way, transitive trace of a COBOL PIC expansion across CICS/batch, Db2, VSAM, BMS, MQ, JCL, copybooks and REDEFINES.

ARCHITECTURE & PLANNING

`implementation-planning`

Spec-driven plans for mainframe changes: system context, dependency mapping, risks, execution sequence and test strategy.

IBM Z KNOWLEDGE

`z-knowledge-analysis`

Pulls vetted IBM Z documentation and examples for COBOL, CICS, Db2, IMS, MQ and HLASM questions outside the Z modes.

AT A GLANCE

7 skills · 4 categories

Activated on demand when a request matches.
Analysis skills use the local scan DB first, with the Z Understand service as an option.
Outputs: validated code, data dictionaries, impact reports and plans.

HOW IT WORKS

How IBM Bob uses MCP

Bob is the MCP client. MCP servers offer extra tools and data. One standard connection means Bob can gain new skills without being changed itself.

IBM Bob
(MCP client)
Understands your request and decides which tool to use

↔
MCP
standard messages
(JSON-RPC)

Local MCP servers (STDIO)
Run on your own machine – fast, simple, no network exposure

Remote MCP servers (Streamable HTTP)
Hosted centrally – shared by many users, updated in one place

1. Discover

Bob connects to each server and learns which tools it offers, what they do, and what inputs they need.

2. Ask and call

When a tool helps your task, Bob asks your approval (unless you chose “Always allow”) and calls it.

3. Use the result

The server sends back structured results, which Bob uses to answer or continue the task.

Get MCP servers from the Bob Marketplace, connect your own, or ask Bob to build one for you.

IBM

RSE API*) MCP Server & Extension Toolkit

Any AI assistant. Full z/OS access. Extensible in minutes.

Docs: ibm.com/docs/en/explorer-for-zos/3.4.0

RSE API MCP Server

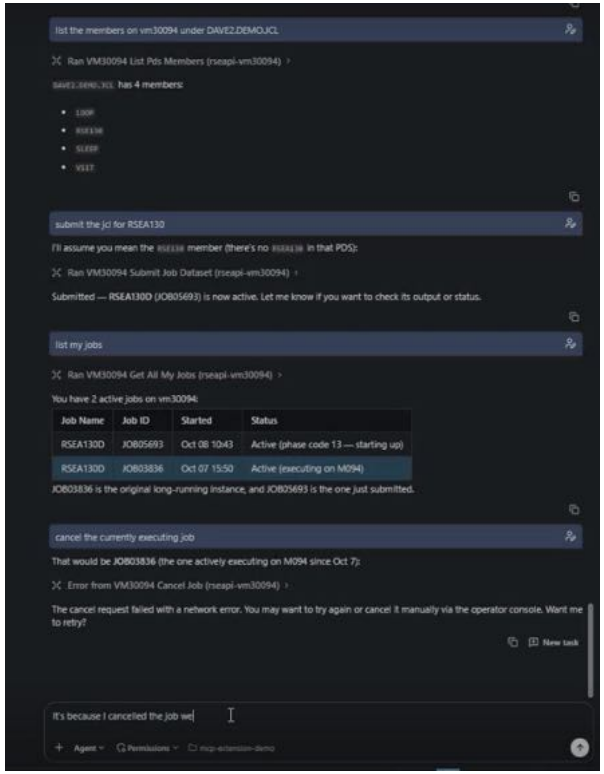
- Exposes RSE API operations as MCP tools
- Enables AI agents (IBM Bob, Claude, Copilot, etc.) to interact with z/OS resources
- Tools: list/read/write data sets & z/OS UNIX files, submit jobs, run shell commands, manage properties
- Configured via standard MCP JSON in Claude Desktop, Bob, VS Code, or any MCP host

RSE API MCP Extension Toolkit

- Customers and partners can extend the RSE API MCP server with their own custom tools
- Tools are built using RSE API's adapter APIs – the same layer that powers REST and MCP, with no overhead

Why it matters

- Brings generative AI workflows directly to z/OS without custom scripting
- Secured by existing SAF/RACF credentials – no new credential is required



© Copyright IBM Corporation 2026

*) RSE API is a part of IBM Developer for Z EE and IBM Application Delivery Foundation for Z

AGENTS.md: Project Guidance for AI Agents

Licence to code

Mission briefing

- It tells the agent what the workspace contains
- Loaded automatically into the agent's context at the start of every session
- Declares the platform, languages and folder layout
- Sets conventions: fixed-format COBOL, JCL dataset names etc.
- Points to the data dictionary, Db2 rules and coding standards
- Maps each program to its documentation

For your AI's eyes only.

The agent's dossier

```
# AGENTS.md

This file provides guidance to agents when working with code in this repository.

## Workspace Type

**Type:** IBM Enterprise COBOL for z/OS with Db2
**Detected Languages:** IBM Enterprise COBOL for z/OS, JCL

## Directory Structure

| Directory | Purpose |
|-----------|---------|
| `cobol/` | COBOL source programs (`.cbl`) and copybooks (`.cpy`) |
| `cpy/` | Shared COBOL copybooks |
| `jcl/` | JCL job definitions (compile, run, DDL) |
| `doc/` | Technical documentation for programs |

## Data Dictionary

**DATA_DICTIONARY:** `bobz/DD.json`

Before reading any `.cbl` or `.pli` file, invoke the `data-dictionary-management` skill to check whether `bobz/DD.json`
exists and contains entries scoped to that program. In multi-root workspaces each workspace folder has its own `bobz/DD.
json`.

## Coding Standards

Coding standards are enforced via the workspace's coding standard skill. Do not apply ad-hoc formatting rules — defer to
the skill when available.

## JCL Conventions

- JCL lines must not exceed **70 characters** (columns 1–70 only; col 72–80 is sequence/comment)
- COBOL compile pipeline: DB2 Precompiler (`DSNHPC`) → COBOL compiler (`IGYCRCTL`) → Link-edit (`IEWL`) → DB2 Bind
(`IKJEFT01`)
- COBOL compile options for Db2-enabled programs: `APOST,RENT,OPT,MAP,LIST,XREF,NODLL`
- Enterprise COBOL compiler PGM: `IGYCRCTL`; library HLQ: `IGY.V6R4M0.SIGYCOMP`
- Link-edit PGM: `IEWL`; options: `RENT,REFR,REUS,LIST,XREF`; include `DB2V13.SDSNLOAD(DSNELI)` in SYSLIN
- Link-edit SYSLIB: `DB2V13.SDSNLOAD` and `CEE.SCEELKED`

## COBOL Coding Conventions

- Copybooks: COPY members in `cpy/` or `cobol/`; include via `COPY <member>.`
- SQL embedded in COBOL: wrap host variable structures in a copybook and `COPY` it into WORKING-STORAGE; include `EXEC SQL
INCLUDE SQLCA END-EXEC` in the copybook
- Input date parameters from SYSIN: read as `PIC X(10)` in `RRRR-MM-DD` format, then move to host variables before use in
```

© Copyright IBM Corporation 2026

OUTCOMES

Productivity, delivery and quality gains

Boost productivity

Innovate faster by offloading routine tasks to AI.

Accelerate delivery

Achieve faster time to value with goal-driven automation.

Improve quality

Identify issues early with continuous guidance.

Onboard new developers faster

IBM Bob accelerates onboarding of new development team members through its code discovery and automated documentation generation capabilities.

Demo

“It worked five minutes ago.”



WAIT… GOOD NEWS!

I recorded the demo!

Nothing can go wrong now!

…right?

Demo outline

Customer email validation

PURPOSE

Make sure every customer email address is validated.

01 **Trace the data flow**

See how customer data flows when adding a new customer record.

02 **Check existing validation**

Find out whether the customer’s email address is verified in any way.

03 **Build the missing validation**

If no validation exists, analyse, design and code a solution that validates the email address.

Demo
po polsku!

Bob rozumie
język polski.



Demo videos

Customer email validation – recordings

ON YOUTUBE

Watch the full demo in five parts.

Part 1	https://youtu.be/ylZmwwuwWRc
Part 2	https://youtu.be/TOKnyXMa-gI
Part 3	https://youtu.be/V0D1mWCFOhY
Part 4	https://youtu.be/h3heVA3LMpg
Part 5	https://youtu.be/a_oWhHNShK4

IBM Bob

Thank you

