Modernizing the Mainframe with the Modern Muscle of z/Cobol

VM Workshop 2026
June 10th – June 12th, 2026
John Rankin
CSI International

Overview

- The Challenge
- The Current State of Legacy COBOL
- IBM Builds Hardware to meet the Challenge
- z/Cobol Architecture
- Modernizing COBOL
- Modernization Tools and Methods
- Modernization Legacy Extensions

The Challenge

- Legacy data and legacy code is here to stay
- The Hardware managing the data is extremely modern and sophisticated
- The Software managing the data is reliable but missing modern tools
- How do we meet the challenges of the modern world without compromising reliability
- The Educational Systems has failed the Mainframe community
  - ✓ Universities don't teach important topics
  - ✓ Students aren't excited by legacy systems
- Executives are disconnected from what is really going on in the Mainframe
- Real Modernization must be Accomplished through Legacy Enhancements

The Modern World

- Artificial Intelligence and Machine Learning
- Containerization and Orchestration (Docker, Kubernetes)
- Communications through SMS and RCS, Handheld Devices
- Big Data and Analytics
- Blockchain and Distributed Ledgers
- Natural Language Processing and Large Language Models
- DevOps/CI/CD Pipelines
- Robotic Process Automation
- Zero Trust Security Architecture

Current VSE Market

- z/VSE or VSEn
  - ✓ COBOL VSE/ESA
    - Requires LE for all math calls
    - Produces code that performs poorly on a z Series Mainframe
    - Code output is designed for ESA
    - Last release 1998
    - No longer for sale with IBM
    - 21st Century selling, but unchanged as of yet
  - ✓ DOS/VS COBOL
    - Still in use in VSE
    - Produces code that performs extremely poorly on a z series Mainframe
    - Last release 1.3.1 (1983)
    - No longer being sold, but prevalent in the field

Current VM Market

- z/VM
  - ✓ COBOL for OS/390 & VM
    - Completely abandoned
    - Produces code that performs poorly on a z Series Mainframe
    - Code output is designed for 370
    - Last Release September 2000
    - Still in the IBM sales manual

Current Linux for z Market

- There is currently no solid offering for Cobol in this market
- GNUCobol provides the only possible Compiler
  - ✓ This Compiler converts Cobol code to C
- IBM offers nothing
- The environment presents significant challenges due to Linux design

Current OS Market

- z/OS
  - ✓ Enterprise COBOL Version 6.5
    - Current staff producing new release with each new Mainframe
    - Uses Vectors sparingly, to increase performance
    - Requires LE for all Math and Manipulation
    - Poorly designed JSON support, only for web interfacing
    - Modern facilities require exit from Cobol and connection to JAVA
    - JAVA performs poorly and is expensive, requiring a specialized ZIIP
  - ✓ Automatic Binary Optimizer for z/OS (ABO)
    - Allows customers to bring object code to a current release without compilation
    - Severely limits major changes to Compiler output and structure
  - ✓ Cobol Report Writer Pre-compiler
    - Replaces the need for the Report Section found in DOS/VS Cobol
    - Has not been updated since 2002

Current World Market

- HP Cobol II XL
  - ✓ Hewlett Packard
  - ✓ For HP 3000 and Unix Platforms
- NetCobol v11.0
  - ✓ Fujitsu
  - ✓ Designed for Windows and Linux for x86
- GNUCobol
  - ✓ Free software Foundation
  - ✓ Built to Compile Cobol to C on many platforms
- Visual Cobol
  - ✓ Rocket Software
  - ✓ Designed for x86, windows, macOS, and Linux
- AIX and IBM i
  - ✓ Enterprise Cobol from IBM

COBOL Standards

- CODASYL Report on COBOL, April 1960
- CODASYL Extended COBOL Specifications, November 1962
- CODASYL COBOL, July 1969, Adopted by ANSI and ISO
- ANSI X3.23-1974, American National Standard COBOL, May 1974
- ANSI X3.23-1985, American National Standard COBOL, Sept. 1985
- ISO/IEC 1989, International Standard COBOL, June 2014
- ISO/IEC 1989, International Standard COBOL, January 2023
- Over 65 years of stable design

IBM Meets the Challenge

- IBM is Fundamentally a Hardware Company
- Reliable and Powerful
- Absolutely meeting the Challenge of the New World
- The IBM z17 meets both the needs of the Legacy Requirements and Support for AI
- The new Mainframe Fully Supports
  - ✓ Hybrid Cloud
  - ✓ Linux for z through ILF
  - ✓ zCX for Linux Containers through Ziip
  - ✓ Large Language Models
  - ✓ Prepared for Quantum Computing

IBM Mainframe meets Physics

- As IBM enters the twenty first century
- The machines can't get much smaller to include more circuits
- IBM meets the challenge with creative and innovative design
  - ✓ Parallel activity
  - ✓ Extreme Caching
  - ✓ Path prediction
  - ✓ Moving more and more of the operation onto the chip
- Highly memory oriented programs suffer
- IBM once again sees COBOL as an important Partner

Twenty Years of IBM Mainframe Development

- z900/z800 – December 2000 (Architecture 5)
- z990/z890 – June 2003 (Architecture 6)
- z9ec/z9bc – September 2005 (Architecture 7)
- z10ec/z10bc – February 2008 (Architecture 8)
- z196/z114 – August 2010 (Architecture 9)
- zEC12/zBC12 – September 2012 (Architecture 10)
- z13/z13s – March 2015 (Architecture 11)
- z14/z14R1 – August 2017 (Architecture 12)
- z15-T01/z15-T02 – September 2019 (Architecture 13)
- z16-A01/z16-A02 – May 2022 (Architecture 14)
- z17-ME1/z17-??? – April 2025 (Architecture 15)

z/Cobol™ for z Series

- z/Cobol™ is designed for all modern IBM Mainframes
  - ✓ 100% Compatible for IBM Cobol VSE/ESA 1.1, 21csw 1.2
  - ✓ Meets the National Institute of Standards and Technology
    - 100% Certified against ANSI 1985 Cobol Standard
    - 340,000 Lines of Cobol, Testing Code Suite with 500+ programs tests
  - ✓ Maximum Performance on all IBM z Series mainframe platforms
- Completely New
- Generates High Performance Code
- Built for z/VSE, VSEn, z/VM, z/OS, Linux for Z, and zCX
- Designed for Modernization Enhancements

Compiler Operation

- Cobol Source code can be structured to match:
  - ✓ ANSI 1968, ANSI 1974, ANSI 1985, or ISO/IEC 1989:2023(E)
  - ✓ Enhancements specifically designed for z Series
    - TCP/IP for ANSI Communication Description Entries
    - Addition Operating System based file methods, and all IBM extensions
- Generated Object code:
  - ✓ Linked with provided library of non LE routines.
  - ✓ No LE environment required.
  - ✓ Object code can be executed anywhere in 24bit, 31bit, and 64bit space
- Supports fully functional export of assembler code as output
- Object code that runs on z/VSE, VSEn, z/VM zCMS, z/OS, Linux on z, and zCX

Code Generated

- 100% 64 bit.
  - ✓ Dynamically adjusts to the calling environment, and uses IBM z/OS save area structures
- Reentrant, no SIIS issues
- Optimized for non memory to memory operation
- Takes full advantage of Long Displacements and Relative Branching
  - ✓ Each storage section can be addressed 1 Megabyte at a time
  - ✓ Branching works with full word relative movements
- All supporting routines
  - ✓ Provided in linkable 64bit object decks
  - ✓ Designed to work with 24/31/64 objects where necessary
- Runtime error recovery, including vector displays

Intellectual Property
z/Cobol™ for z Series

- U.S. Patents:
  - ✓ 10,901,739 Issued: January 26th, 2021
  - ✓ 11,429,390 Issued: August 20th, 2022
- COBOL Standards
  - ✓ CODASYL COBOL – Issued: July, 1968
  - ✓ X3.23-1974 ANSI COBOL, Issued: May 10th, 1974
  - ✓ X3.23-1985 ANSI COBOL, Issued: September 10th, 1985
  - ✓ ISO/IEC 1989/Amendment 1, Intrinsic function module
- Guiding the International Committee for Information Technology Standards (INCITS)
  - ✓ Building the new COBOL Standards for the future

Embracing the Power of the Hardware

- Levels of Support
  - ✓ Basic z/Series support, Architecture level 5
    - 64 instructions, relative branching, and long displacements
  - ✓ Immediate Values, Architecture level 7
    - Reduces the use of literal pools, and moves literals into instructions
  - ✓ Decimal Floating Point, Architecture level 9
    - Moves binary coded decimal away from memory-to-memory operations
  - ✓ Vector Facility, Architecture 12
    - Uses all 32 Vector Registers
    - Completely eliminates binary coded decimal memory instructions
    - All math operations occur on the chip, storing only when necessary

Vector Code Generated

- As math operations occur when z/Cobol™ elements are loaded into Vectors
- All 32 Vectors are continually used
- Vectors are saved prior to call operations
  - ✓ Allows z/Cobol™ code to use vectors while in CICS
  - ✓ Interfaces with standard callable routines
- Perform Verb operations
  - ✓ Utilizes Vectors for comparisons, and increments
  - ✓ Operates with as much data loaded into Vectors
- High performance operation
- Maintaining maximum instructions and data on chip and not memory

Modernizing COBOL

- Translating and Converting COBOL has Proved to be a Failure
- COBOL Modernization Solves the Legacy Support and Reliability Problem
- The legacy code must be allowed to connect and merge with the modern world
- These Features are:
  - ✓ Artificial Intelligence and Machine Learning
  - ✓ Containerization and Orchestration (Docker, Kubernetes)
  - ✓ Communications through SMS and RCS, Handheld Devices
  - ✓ Big Data and Analytics
  - ✓ Blockchain and Distributed Ledgers
  - ✓ Natural Language Processing and Large Language Models
  - ✓ DevOps/CI/CD Pipelines
  - ✓ Robotic Process Automation
  - ✓ Zero Trust Security Architecture

Modernization

- z/Cobol is a completely compatible base with IBM COBOL
- Structure:
  - ✓ Object code is 100% 64-bit operation
  - ✓ Working Storage section can be: 24, 31, or 64 bit in the same program.
  - ✓ Each working storage section can be 2G in size
- Modern Outreach:
  - ✓ Complete JSON control:
    - Building JSON and parsing Json string is built into the language.
    - Utilized for web interfacing and communication with applications
  - ✓ Entire Network Section and Control structure
    - Clean, simple, build in Network Communication
    - RESTful API, JSON-RPC, MCP, TCP/IP, Secure Communication
- Completely extendable language base

z/Cobol embraces the Modern High-tech World

- z/Cobol Extends beyond standard Cobol
- JSON Implementation
  - ✓ Places the definition of JSON objects in the data area
  - ✓ Adds extremely flexible verbs to the PROCEDURE DIVISION
    - GENERATE JSON, and PARSE JSON Verbs
- New NETWORK SECTION and NETWORK-CONTROL
  - ✓ Within the design form of COBOL
  - ✓ Defines Modern Networking connections
  - ✓ Identifies Methods for managing Protocols
  - ✓ Within the PROCEDURE DIVISION adds
    - SEND and RECEIVE Verbs
- Clean, Usable, Simple, and fits within COBOL

RESTful API

- RESTful API
  - ✓ Combining the JSON support with NETWORK methods
  - ✓ Allows for connections to any standard provider
  - ✓ Examples of real-world use:
    - OpenAI, to make programmatic calls to Chat-GPT
    - MongoDB, mySQL in Linux platforms
    - Twilio SMS API platform
    - Cybersource Credit Card Platform
    - Stripe.com, manage collecting credit card funding
    - Quickbooks complete integration with offplatform accounting
    - Coinbase Interfacing for complete trading management
    - Google Ads can automate advertisement bidding
    - Facebook.com can be controlled with campainges
    - Amazon allows for sales integration

Java Script Object Notation JSON

- JSON-RPC
  - ✓ Connects and drives crypto blockchain management
  - ✓ Updates crypto currencies for payments and queries
  - ✓ Numerous server systems
- TCP/IP
  - ✓ Secure communication
  - ✓ Complete access to all modern cryptographic methods
- Blockchain
  - ✓ Read and Write to the Blockchain, as if it was a file
- JSON support allow web services with TCP/IP for z/VSE
  - ✓ Web servers to fully support calls
  - ✓ Call outs to web interfacing
  - ✓ Compatible with z/OS CICS web services

Modernization Tools and Methods

- Once COBOL is enhanced there is a wide range of modern tools
  - ✓ Tools that are within the Mainframe Complex
  - ✓ Tools that are external to the Mainframe Complex
- COBOL Programs can Connect, Augment, and Integrate
  - ✓ All Modern Services
  - ✓ Extensive Modern Hardware

IBM z17 Spyre AI Accelerator

- Available October 2025 for the z17
- Each card contains 32 AI Accelerator Cores and 128G
- Each I/O support drawers holds 48 cards
- Massively Powerful AI LLM engine, specifically designed for z17
- Spyre AI loaded with Pytorch and TensorFlow for LLM
- When combines with z/Cobol and TCP/IP
  - ✓ Full access to the Spyre array for AI
  - ✓ Cross connection between Linux on z and z/OS for sharing
  - ✓ Legacy can contribute data and training to the LLM model
  - ✓ Full access to the LLM model is available from any z/Cobol program

AI Services

- Compiler advice and guidance
  - ✓ Source code is communicated to LLM of choice, (ChatGPT, Claude, etc.)
  - ✓ LLM analyzes COBOL code, and makes recommendations
  - ✓ No modification of original source, only analysis
- RESTful API, and JSON
  - ✓ OpenAI ChaptGPT API
  - ✓ Anthropic Claude API
  - ✓ Google Gemini API
  - ✓ Mistral API
  - ✓ xAI Grok API

Cloud Platforms

- RESTful API, and JSON
  - ✓ Amazon Web Services
  - ✓ Microsoft Azure
  - ✓ Google Cloud Platform
  - ✓ IBM Hybrid Cloud

Databases

- RESTful API, and JSON
  - ✓ PostgreSQL
  - ✓ MySQL
  - ✓ MongoDB
  - ✓ CouchDB
  - ✓ Elasticsearch

CRM Systems

- RESTful API, and JSON
  - ✓ Salesforce
  - ✓ Hubspot
  - ✓ Microsoft Dynamics 365
  - ✓ Zoho CRM

ERP Systems

- RESTful API, and JSON
  - ✓ SAP
  - ✓ Oracle ERP Cloud
  - ✓ NetSuite
  - ✓ Workday

Financial Services

- RESTful API, and JSON
  - ✓ Banking APIs
  - ✓ Credit scoring services
  - ✓ Fraud detection systems
  - ✓ Payment processors
  - ✓ ACH Gateways
  - ✓ Treasury Management Systems
- Examples
  - ✓ Stripe
  - ✓ PayPal Developer Platform
  - ✓ Plaid

Cryptocurrency/Web3

- JSON-RPC
  - ✓ Coinbase Developer Platform
  - ✓ Alchemy
  - ✓ Infura
  - ✓ Ethereum node APIs
  - ✓ Wallet services
  - ✓ Smart contract interfaces

Communications

- RESTful API, and JSON
  - ✓ SMS
  - ✓ Email
  - ✓ Voice
  - ✓ Video
- Examples
  - ✓ Twilio
  - ✓ SendGrid
  - ✓ Zoom Developers

Identity and Security

- RESTful API, and JSON
  - ✓ Single Sign-On
  - ✓ Multi-Factor Authentication
  - ✓ Identity Verification
  - ✓ Fraud Detection
- Examples
  - ✓ Okta
  - ✓ Auth0
  - ✓ Microsoft Entra ID

Business Intelligence

- RESTful API, and JSON
  - ✓ Dashboards
  - ✓ Reporting
  - ✓ Analytics
- Examples
  - ✓ Tableau
  - ✓ Microsoft Power BI
  - ✓ Looker

Government and Public Data

- RESTful API, and JSON
  - ✓ Weather Services
  - ✓ Postal Services
  - ✓ Tax Services
  - ✓ Census Data
  - ✓ Regulatory Reporting

Social Media

- RESTful API, and JSON
  - ✓ LinkedIn Develoeprs
  - ✓ X Developer Platform
  - ✓ Facebook for Developers

Enterprise DevOps

- RESTful API, JSON
  - ✓ GitHub API
  - ✓ GitLab API
  - ✓ Jira Software
  - ✓ ServiceNow

Modernization Legacy Extensions

- z/Cobol can be used to Build Modern Servers
  - ✓ Interacting in Modern Infrastructures
  - ✓ Providing a connection to Legacy Data
  - ✓ Extending a value-added network for profit sharing
- z/Cobol can move workloads off CP processors and onto Linux Workloads
  - ✓ z/Cobol can produce object code that can be containerized
    - Executing within Linux on z
    - Executing within zCX, with direct access to Legacy data within z/OS

Ethereum Oracle

- A Legacy Oracle can profit from providing access to information
  - ✓ Shipment status and location
  - ✓ Customs clearance confirmation
  - ✓ Product authenticity verification
  - ✓ KYC/AML verification status
  - ✓ Credit scores
  - ✓ Academic or professional credentials
  - ✓ Government ID verification results
  - ✓ Asset prices (crypto, stocks, ETFs, commodities)
  - ✓ Exchange rates (forex pairs) Interest rates and yield curves
  - ✓ Market capitalization and trading volume
  - ✓ Options and derivatives pricing

Model Context Protocol (MCP)

- Designed by Anthropic for connection to LLM
- Adds a proprietary LLM to an Anthropic Context Window for query
- The Claude training period is fixed and only includes data from training
  - ✓ Natural Language Queries do not include data outside of the training window
  - ✓ Proprietary LLM Neuro Networks can be built and connected to Claude
    - MCP is the protocol for this connection
    - This allows a powerful and seamless connection to Claude
  - ✓ Company Specific data can be connected
- Proprietary LLMs can be
  - ✓ Data concerning company specific information
  - ✓ Source code libraries
  - ✓ Human resources

Ethereum/Web3

- By combining:
  - ✓ Blockchain
  - ✓ Smart Contracts
  - ✓ WEB3 Enabled Wallets
- A program can be built to communicate, store, and manipulate data
- Complete secure, with modern presentation, and decentralized data
- WEB3 wallets
  - ✓ Applications like TRUST, and METAMASK
  - ✓ Connect to a blockchain network, like Ethereum
  - ✓ Allow html presentations
  - ✓ Provide a virtual machine for smart contract execution
  - ✓ These can also be extensions for web browsers, like OPERA

Docker

- Containerization Platform
  - ✓ Docker packages applications and their dependencies into lightweight, portable units called containers, ensuring consistent behavior across any environment
- OS-Level Virtualization
  - ✓ Unlike traditional VMs, Docker containers share the host operating system kernel, making them far lighter and faster to start than full virtual machines
- Docker Images
  - ✓ Containers are built from images, which are read-only templates defined by a Dockerfile; images are layered, so only changed layers need to be rebuilt or transferred
- Docker Hub
  - ✓ A public registry where pre-built images can be shared, downloaded, and versioned, covering everything from Ubuntu to Node.js to entire application stacks
- Isolation
  - ✓ Each container runs in its own isolated process space, filesystem, and network, preventing conflicts between applications sharing the same host

Thank You