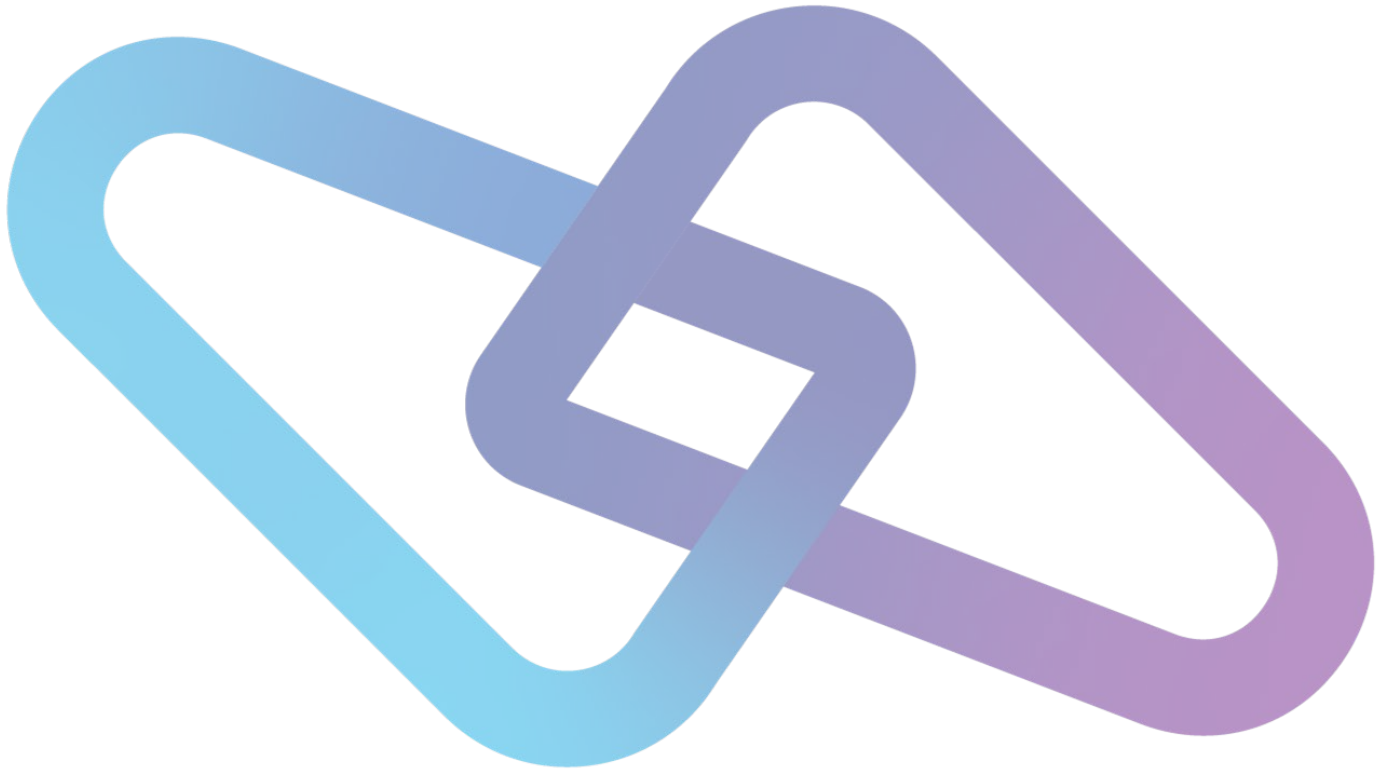
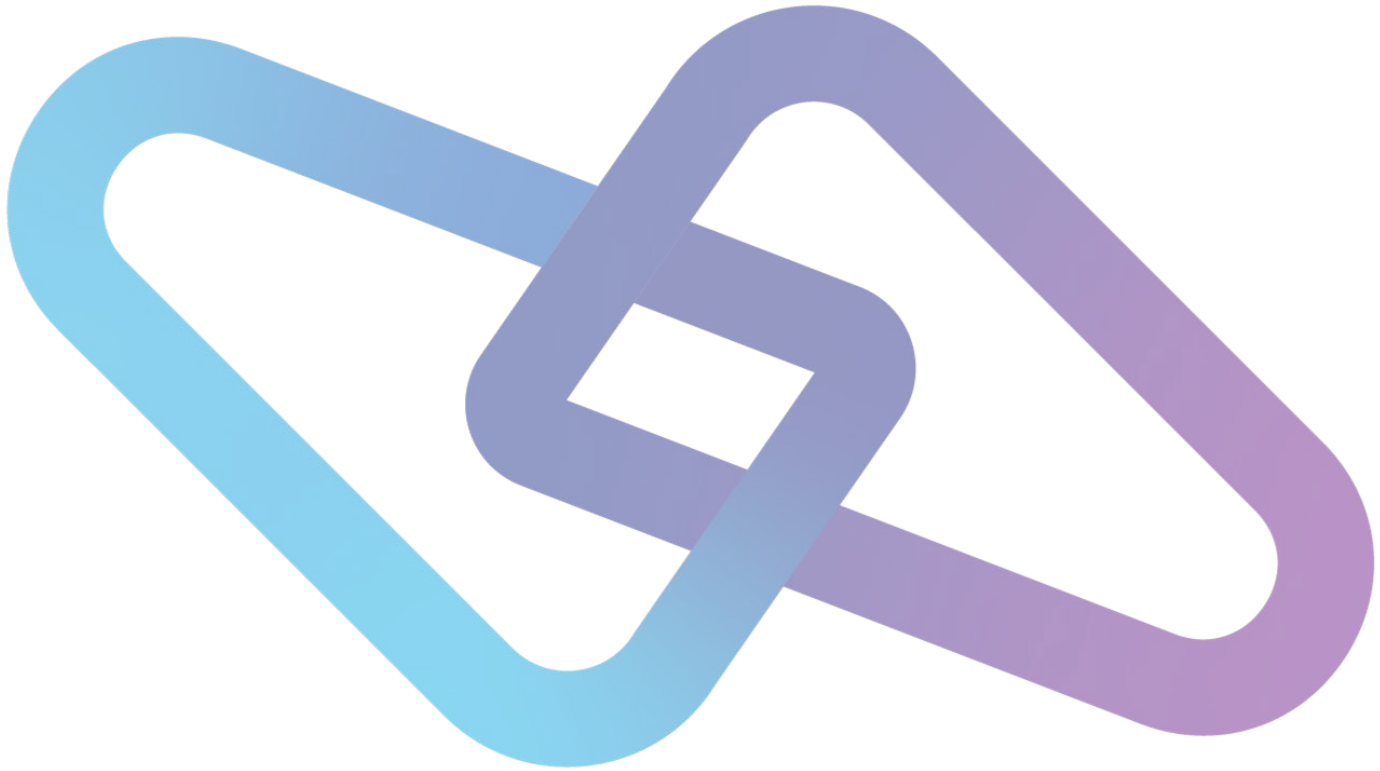


VSEn Connectors and VSCode

21CS VSE^{n} Team
vse@21cs.com

Todays Topics

- VSEn Connectors
  - Health Checker
  - VTape Server
  - Script Server
  - Navigator
- VSCode
  - Goals, Scope and Challenges
  - Modernizing Development



VSEn Connectors

VSEn Connectors

Health Checker

System diagnosis and rule-based performance analysis. Evaluates live configuration data against customizable rules.

Connects to a live VSEn system
Pulls config data, runs user-defined rules
Exports structured XML reports

VTape Server

Virtual tape emulation. Treat any file on any platform like a physical tape on VSEn.

Maps files as virtual tape drives
No changes to existing tape-based workflows
Stores tape data in portable AWS binary format

Script Server

Accessing VSEn from any platform, in any language, without Java.

Any language can talk to VSEn
C/C++, COBOL, Python, VBA — all supported
Three-tier: Client → Script Server → VSEn

Navigator

The broadest connector tool. Full graphical access to Librarian, ICCF, VSAM, POWER, and system management.

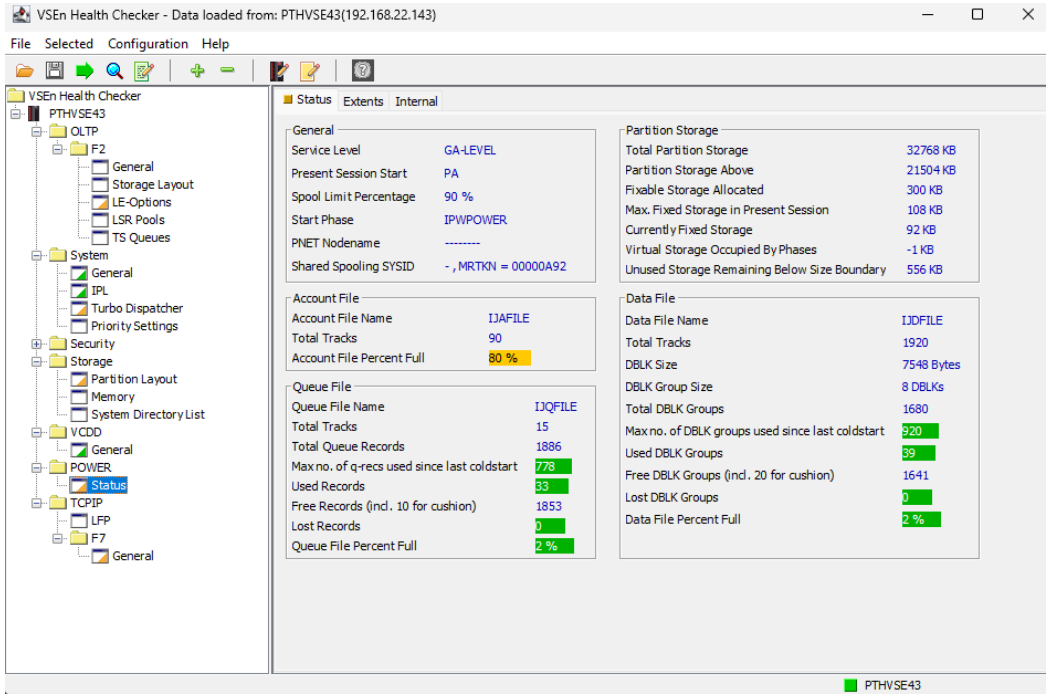
GUI access to Librarian, ICCF, VSAM, POWER
Browse, upload & manage source without a terminal
Leverages the open Connector Client API

Connectors: VSEn Health Checker

Diagnostic and monitoring tool

Connects to a live VSEn system, pulls configuration and performance data, and evaluates it against user defined rules. Flagging attributes in various status.

The rules database is customizable XML — user defined "*healthy*" aspects of the environment.



TD_OVERALL_UTIL

Evaluation

Actual value of TDOVERALL	-1
Matching calculation	($TDOVERALL <= 90);
Partition	-
Analysis Result	OK

Explanation

Overall CPU load should not exceed 90%.

How to display this parameter on VSEn

```
query td,internal
AR 0015  CPU   STATUS    SPIN_TIME    NP_TIME TOTAL_TIME NP/TOT  DISP_ENTR
AR 0015   00   ACTIVE            0      12554      17731  0.708    3065846
AR 0015   01   INACTIVE
AR 0015                 ------------------------------------------------
AR 0015 TOTAL                   0      12554      17731  0.708    3065846
AR 0015
AR 0015               NP/TOT: 0.708      SPIN/(SPIN+TOT): 0.000
AR 0015  OVERALL UTILIZATION:   0%        NP UTILIZATION:   0%
AR 0015
AR 0015  ELAPSED TIME SINCE LAST RESET:     72413040
AR 0015  NUMBER OF SVC'S SINCE LAST RESET:   1713307
AR 0015 1I40I  READY
```

How to change this parameter

N/A (output only)

TD_NP_UTIL

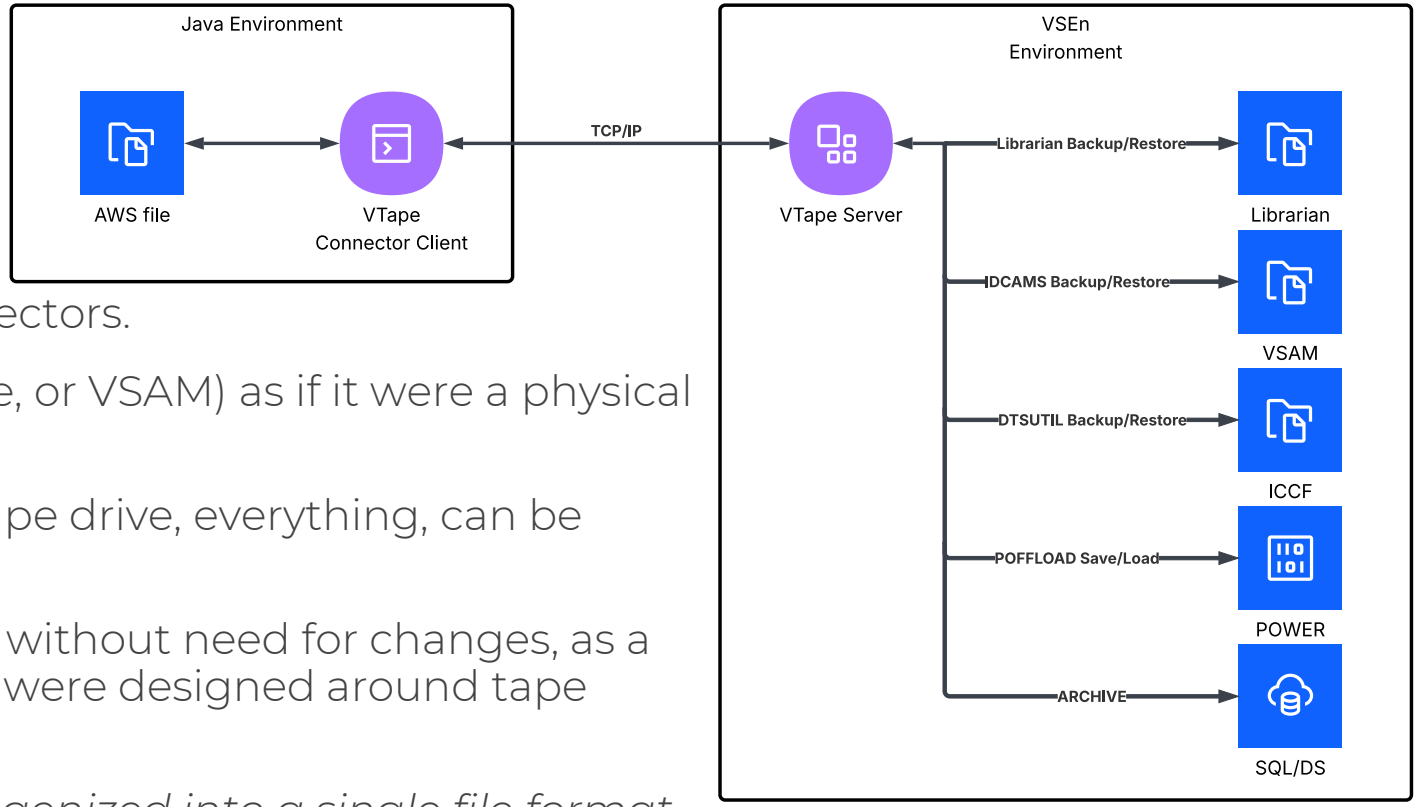
Data retrieval via console commands and submitting jobs; entirely API-driven, no special hooks.

Non-continuous monitoring; creates a structured snapshot that can run either regularly, or after any significant change.

Output can be exported to XML and viewed in a browser; structured and portable.

Connectors: VSEn VTape Server

Virtual tape interface component of VSEn



Possibly the most useful of the VSEn Connectors.

A middleware that treats files (local, remote, or VSAM) as if it were a physical tape.

Everything that would require a physical tape drive, everything, can be proxied as a “tape”.

Allowing for existing processes to still work without need for changes, as a lot of VSEn backup systems and processes were designed around tape storage mediums.

**All write services that use tapes are homogenized into a single file format, AWS binary files. Highly portable.*

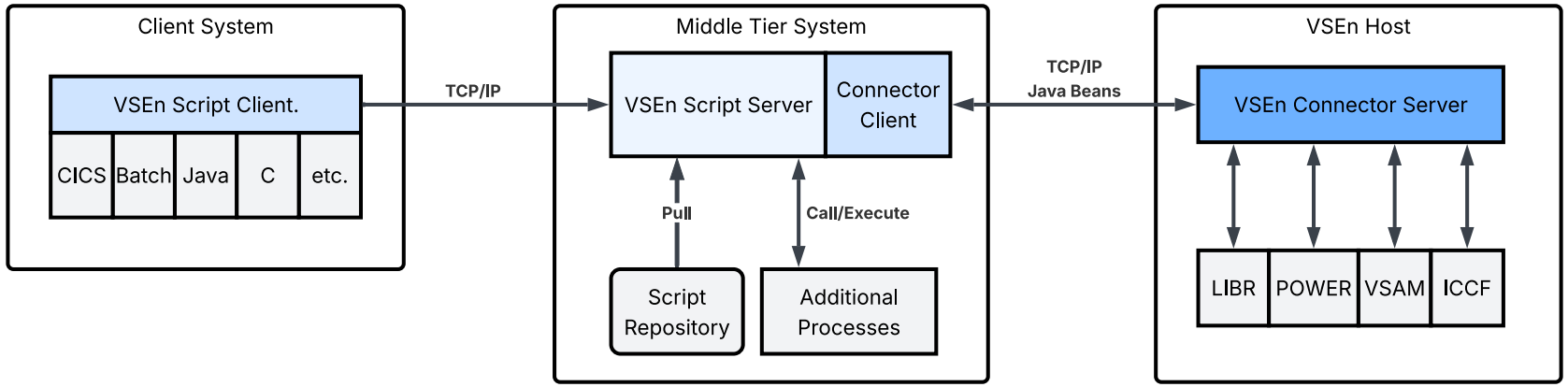
Connectors: VSEn Script Server

Middle-tier translation server that lets any platform — Java or not — talk to VSEn.

Operates at a three-tier architecture Client → Script Server → VSEn.

The client can run in any language. C/C++, COBOL, CGI and even Visual Basic for Excel marcos.

As the script server is just a wrapper for the connector Java-beans you can use any language, even those not designed for it, like Python or C#. If it's a script file, it can talk to VSEn.



Example: A spreadsheet pulling live VSEn data through a VBA macro. Unusual, but entirely supported. Demonstrates how accessible the connectors are intended to be.

**The main advantage of this connector. Java Beans already covers Java clients. The Script Connector covers everything else — Windows, Linux, legacy systems, office tools.*

Connectors: VSEn Navigator

General purpose interface. Providing a clean access view outside of the terminal.

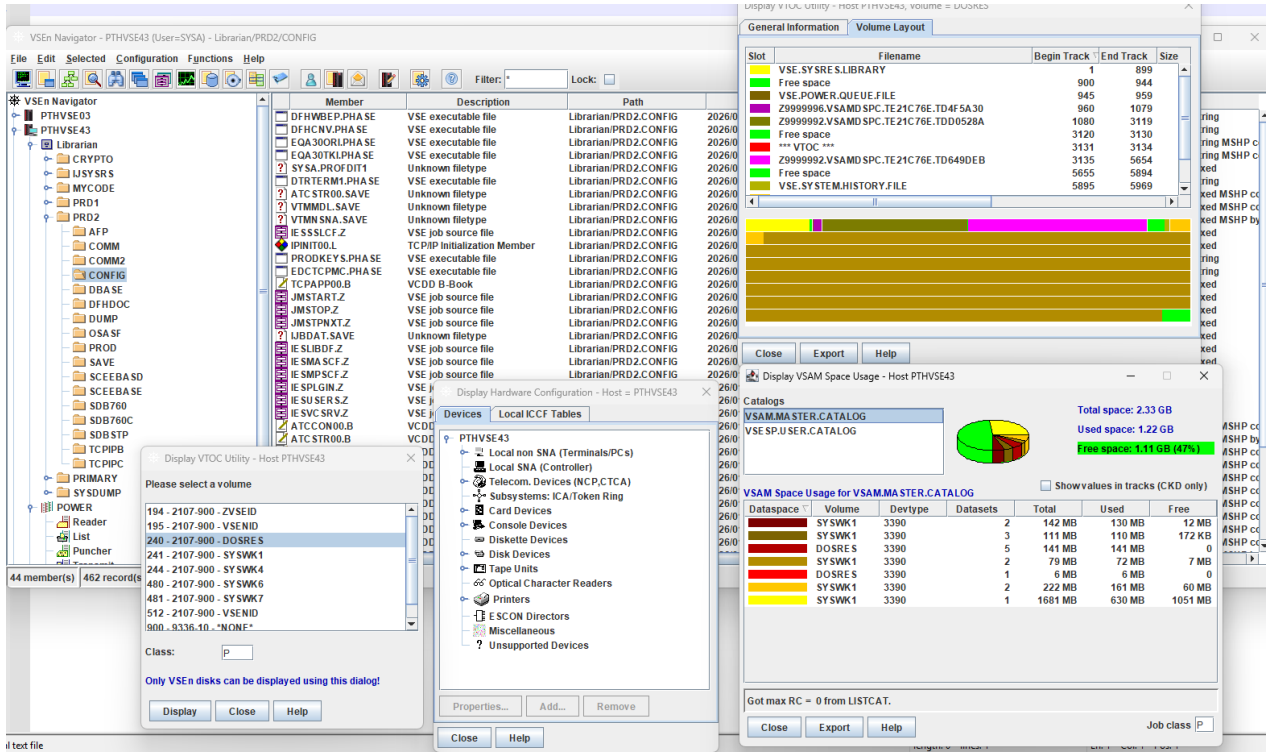
Librarian, ICCF, VSAM, and POWER interaction.

Rough IDE with text editors.

Browse, download, upload, and manage source members without a terminal.

Provides convenient displays for various systems,

- VSAM catalog/Cluster information
- POWER queue entries
- System information…
  - VTOC allocations
  - Partition/Task status
  - Terminal control
  - Hardware information
  - *and a whole lot more…*

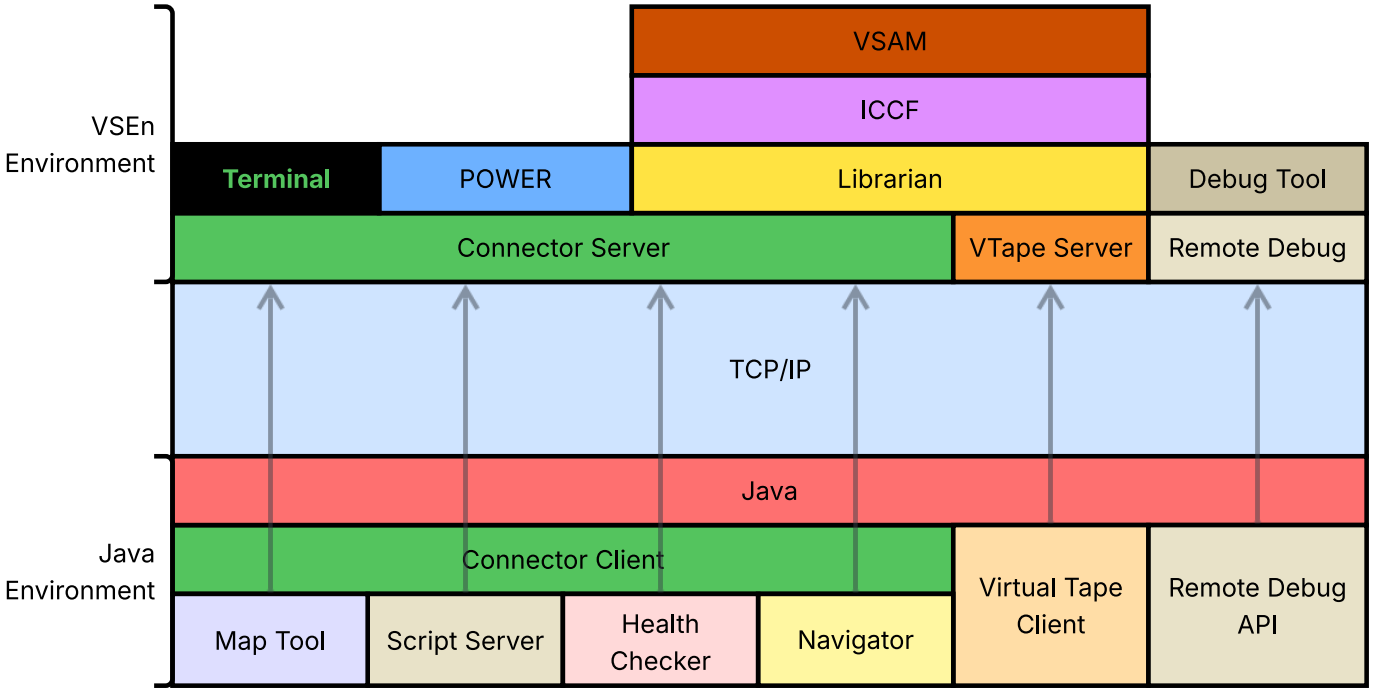


Utilizing the VSEn Connector Client API to interface with VSEn.

**A freeware Java API library that any customer can develop their own systems from.*

Connectors: The Common Thread

Many tools. One underlying Factor. Java powers them all.

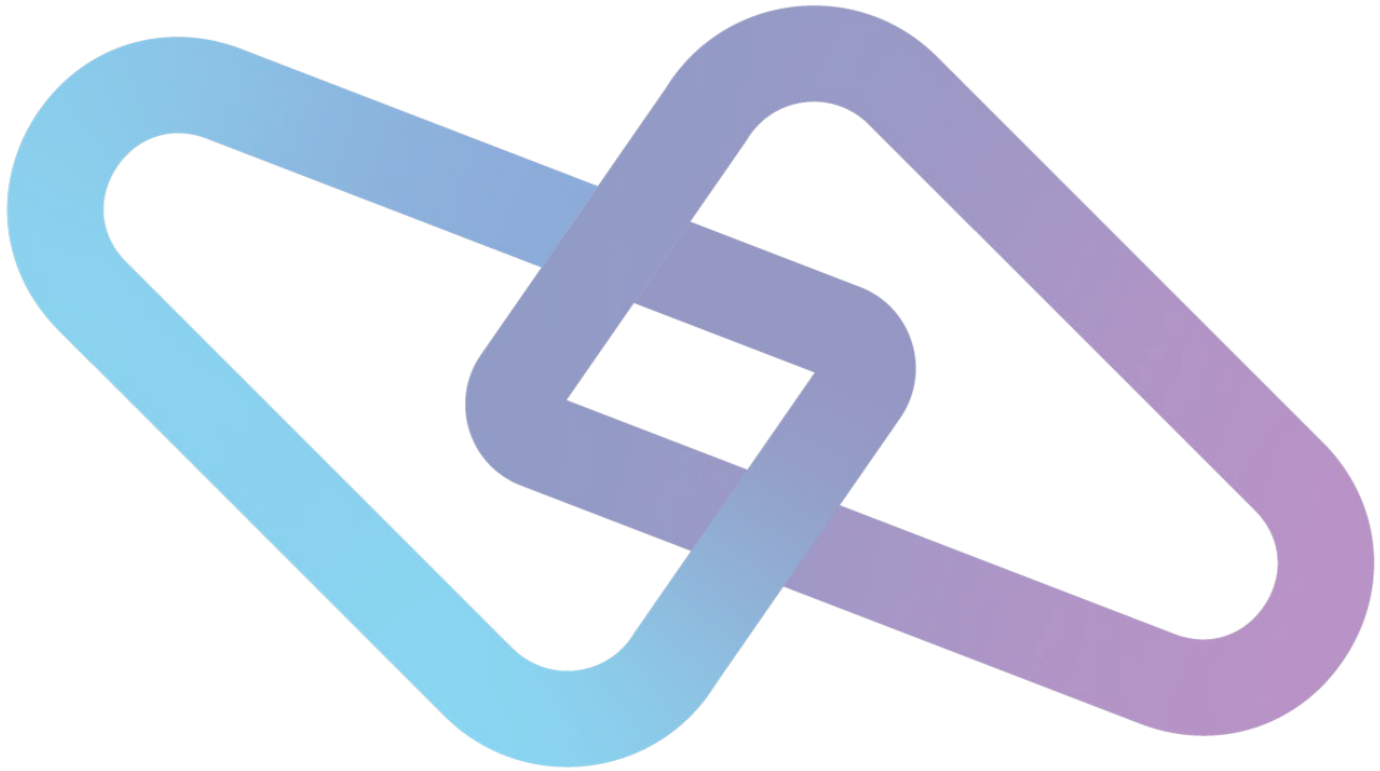


Each Connector Client API is made using Java.

Although implementation may vary, the underlying language remains a consolidation of services, making it highly portable, modular and upgradable.

This is just a few of the connectors.

The connectors are useful on their own. But what if something used all of them together?



VSECode

Modern Development on VSEn

Brief IBM Mainframe IDE History

Previous Attempts...

VSEn Navigator

Designed as a system management GUI, not an IDE. You can open members in an external editor, but there's no compilation, no job submission, no output capture. Also struggles with multiple EBCDIC codepages.

RDz/IBM Developer for z/OS

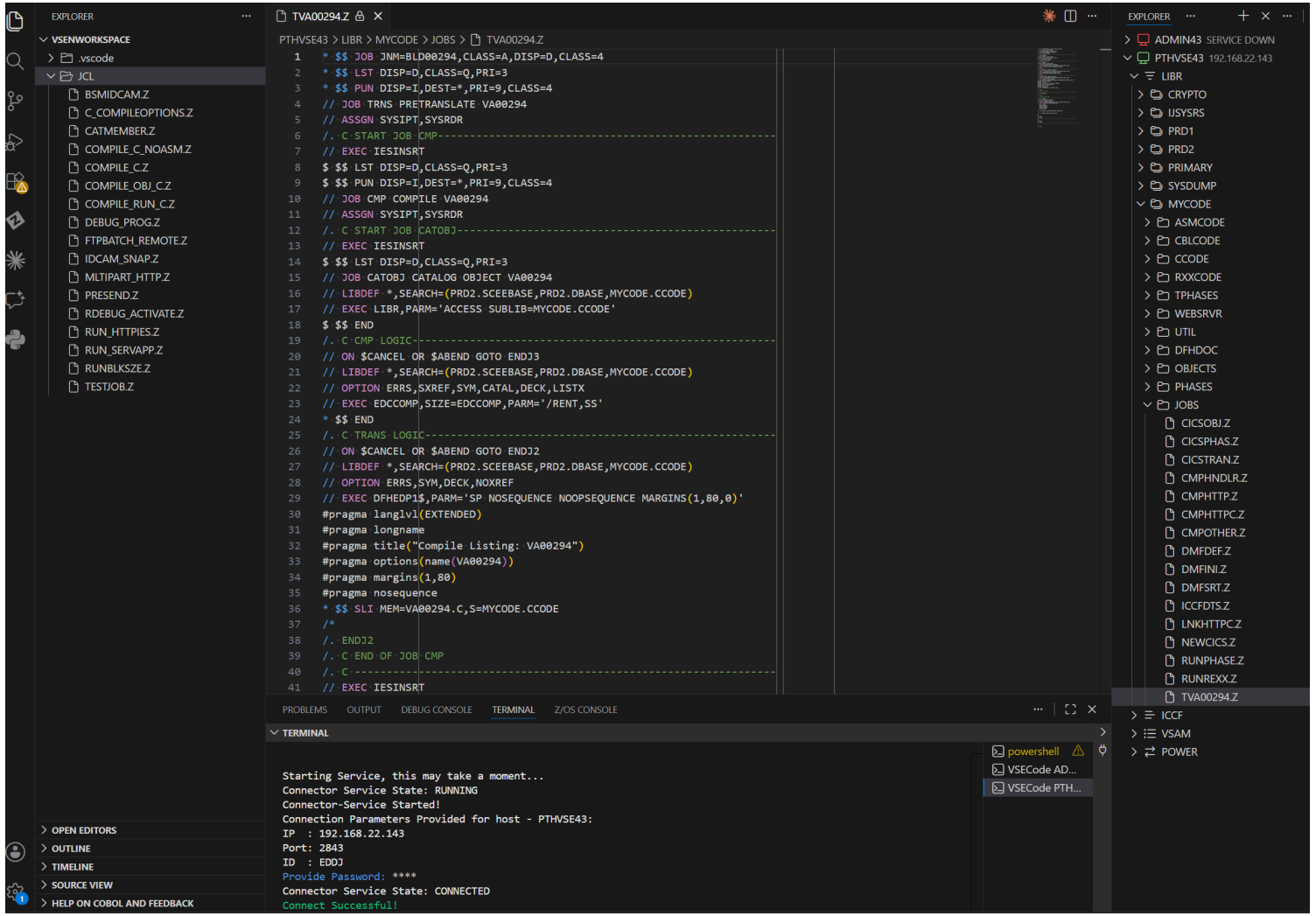
IBM's serious Eclipse-based mainframe IDE. A z/VSE plugin was developed by QGroup.
Base product is large, slow to start, and aimed firmly at z/OS.

IBM Z Open Editor (VSCode)

IBM's modern VS Code extension for mainframe development. Free and well-maintained. A good product.
But it targets z/OS exclusively. VSEn is not supported.

No modern option to select from...

VSCode for VSEn



A VS Code extension bringing modern development tooling to VSEn mainframe systems

Modern Development

What does modern development offer?

- Quality of Life Tooling
  Syntax, Error, Highlighting Checking
- Program Debugger
  Set breakpoints, step through code, inspect variables/memory.
- Version Control
  Pushing data to and from the workstation to VSEn. Diff view and change rollback.
- AI Assistant **Emerging Technology*
  Have AI review source code directly from the system to provide the fastest support and solutions.

All of this depends on one thing — the IDE being able to see VSEn's file system. That's the gap VSECode closes.

Common Mainframe Problems

Things that stop you just opening a file and editing it…

- Round Trip File structures. Records <-> Streams
  Convert binaries between record-based datasets and byte streams.
- Codepage Translation. EBCDIC <-> ASCII
  Work between different source members using different encodings.
- Remote control to Librarian and POWER
  Create, delete, modify and compile source code safely from one place.

Making it easier to leverage VSCode's features…

- Syntax highlighting
- Syntax Checking
- Autocomplete
- Copy/Cut and Paste
- Undo/Redo

**Things every modern editor has had since the 1990s.*

Record Lengths

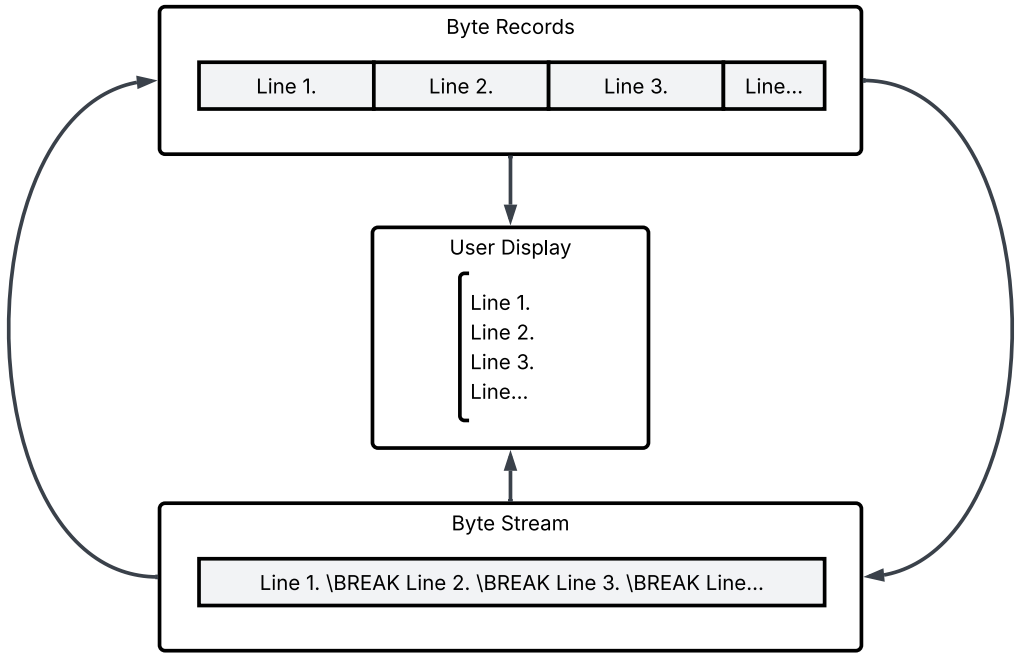
Librarian resource meta data:

- Record Lengths
  Treat each record as a newline; encoding does not help, as what a new line *is* can be fuzzy.
- Formatting
  Librarian and ICCF records are limited to 80 bytes, making each “line” only 80 characters long.

VSEn doesn't store source code as a stream of text with newlines. It stores it as fixed-length records — 80 bytes each, no line endings, no exceptions. An editor doesn't understand that natively. VSECode does.

POWER entries are an exception; But are read-only by design.

Display MUST remain consistent between conversion



Codepage Translation

EBCDIC translation is handled on the workstation side rather than relying on the connector server, due to the connector server being able to use one codepage at a time.

- Transfer of raw binaries only, no translation during transit. EBCDIC in *and* out.
- Translation only occurring on the presentation layer level, no changes to the underlying bytes they represent.
- Create our own encoder on VSCode to deal directly with edge cases/problems with record lengths into newlines.

Per-file codepage selection means you can have members in different encodings in the same workspace simultaneously.

Due limitations of the Librarian, code pages must be managed by the users themselves.

Supported Codepages

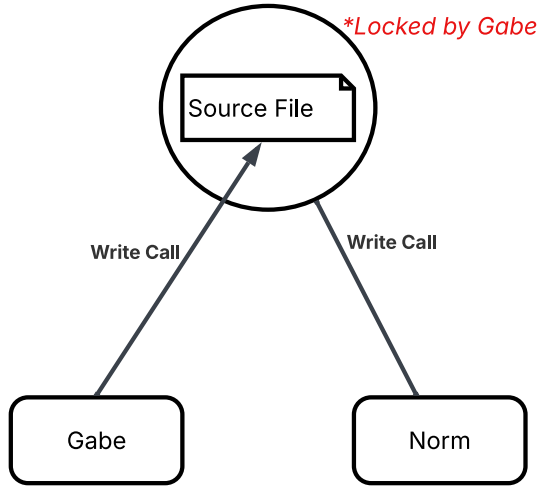
IBM-037
IBM-273
IBM-277
IBM-278
IBM-280
IBM-284
IBM-297
IBM-500
IBM-871
IBM-1047
IBM-1140
IBM-1141
IBM-1142
IBM-1143
IBM-1144
IBM-1145
IBM-1146
IBM-1147
IBM-1148
IBM-1149
ISO8859-1
UTF-8

Resource Locking

Connector Librarian API supports resource locking and creation/modification date tags.

Locking allows for reliable writes even with multiple users accessing datasets at the same time.

A powerful tool for version control. Without locking, two users editing the same member simultaneously would silently overwrite each other's changes — a real problem in shared development environments.



Modification timestamps on each member means the extension can detect whether a local copy is out of date before a write attempt — without comparing the file contents."

Basic IDE Features!

Out of the box with VSECode

- Drag and drop to copy members between libraries
- Syntax error highlighting (red squiggles) for COBOL and PL/I
- Multi-tab editing — work on several members simultaneously
- Integrated terminal panel for submitting jobs
- File explorer tree for Librarian and ICCF members
- Undo/redo, cut/copy/paste, minimap — things every modern editor has had since the 90s

These features come free — VSECode just gives them something to work with.

How VSCode Modernizes Development

VSCode has a few powerful features that can be tapped into.

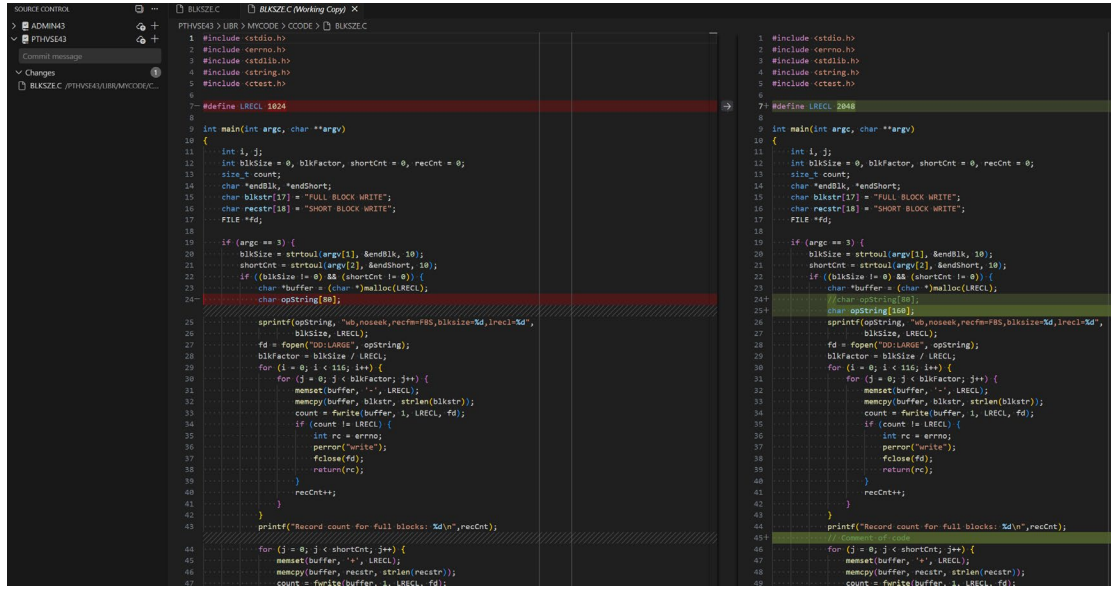
- Source Control Manager (SCM)
- GIT
- AI
- *...anything anyone has ever done with another extension.*

These aren't features we had to build. They're already there. VSECode just gives them something to work with.

VSCode Source Control Manager

VSCode has a powerful SCM API that we can leverage for VSEn development.

The SCM panel shows every modified member as a pending change. You can diff before committing, roll back individual files, and review exactly what's going to the system before it goes. This works independently of Git — it's VSECode tracking your VSEn changes.



GIT

What about GIT?

Implementation/Bridge to GIT is in the works.

Librarian workflows are well-positioned for Git integration — the member model maps cleanly to a file-based repository.

ICCF workflows are still being developed.

VSEn has had no version control for its entire existence. ICCF was designed in the 70s and is tied very close to core components of the operating system. The issue is less about the connector capabilities, but the design of ICCF itself.

ICCF uses a different member structure to Librarian, and getting anything to work with its members is the current technical challenge.

Given how deep the solution of GIT may be to implement. Git integration may not come from this extension.

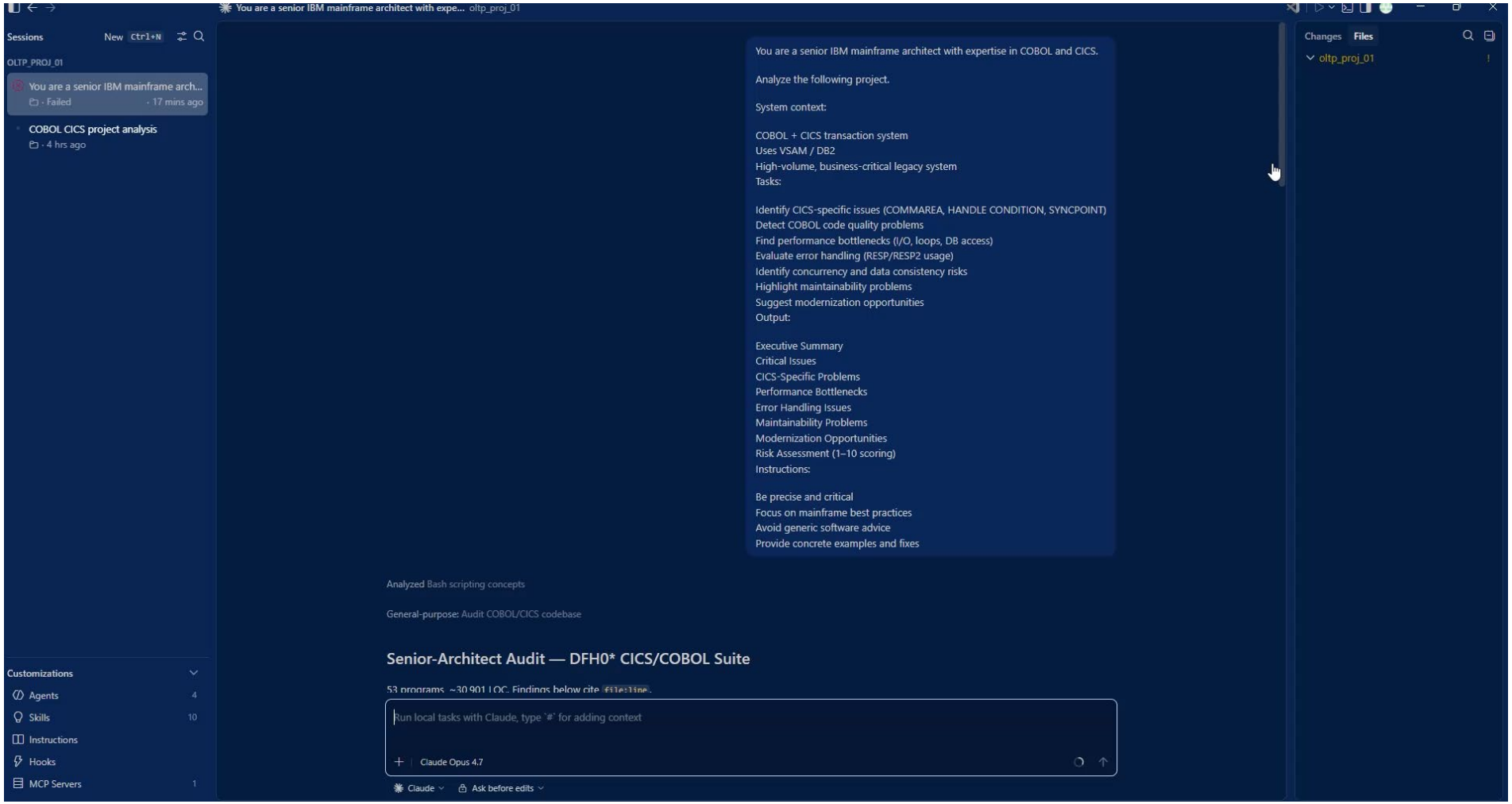
But the things we are learning from this are definitely helping us build a viable solution for VSEn and GIT.

AI

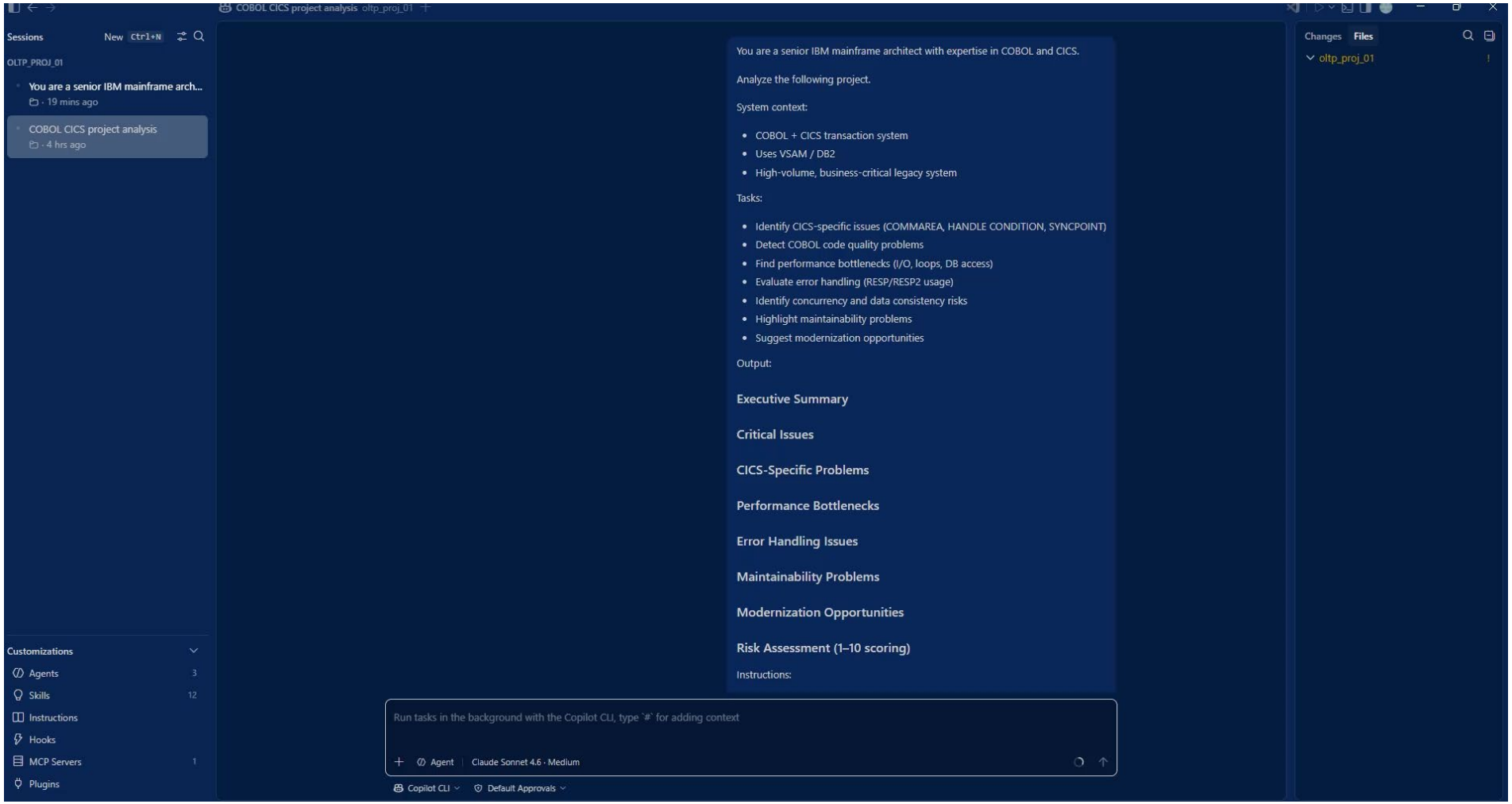
- VSEn systems often have COBOL and PL/I programs that haven't been touched in decades
- Original developers are gone, documentation is sparse or nonexistent
- Understanding what a program does before modifying it is slow and risky

- Source members are right there in the editor, so you can pass them directly to GitHub Copilot, ChatGPT, or any AI extension without copy-pasting out of a terminal
- Ask it to explain what a program does in plain English
- Ask it to identify what a specific paragraph or section is responsible for
- Ask it to spot potential issues before you submit a change

Claude Example



Copilot Example



What VSCode Unlocks with the Connectors

Leveraging the connector API *with* VSCode allows us to take what came before even further…

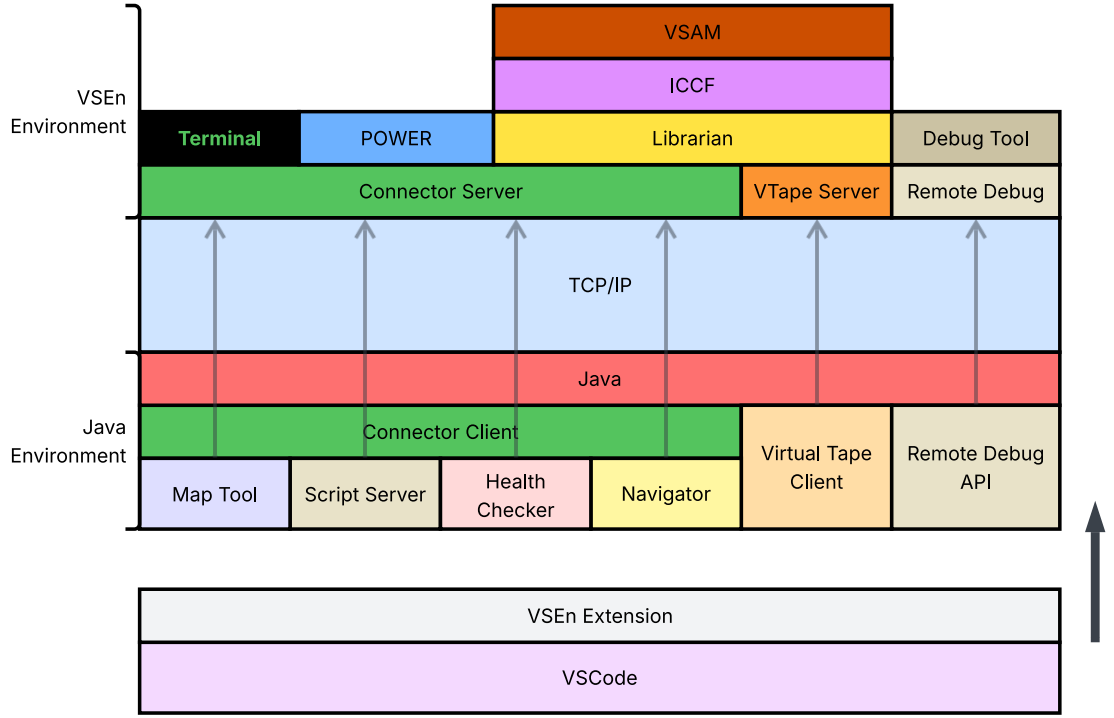
- Navigator: Edit/Delete of source members.
- Health Checker: Remote Submission and read of POWER job queues.
- Script Server: Remote execution of automated processes.
  Potential for native build tools.
- VTape Server: Backup datasets en mass.
  Potential for GIT migrations.

What about other connectors?

- Remote Debugger: Hook into VSCodes debugger API.
  Stepping and breakpoints become instant.
- VSAM Mapping tool: VSAM data viewing.
  Direct dataset viewing.

Summary

The goal isn't to replace these services. It's to leverage what is readily available for customers to adopt it. No additional installation of programs, just what VSEn has out the box.



Taking advantage of what came before to build something now.

Demo

Quick demo. More to show tomorrow

Q & A

Thank you!