CM FIRST
Rethink Modernization

// WHITE PAPER

Transform Legacy Code to Maintainable Java

CM evolveIT metaTX-AI can refactor your legacy app developed in native COBOL, or CA 2E (Synon) generating RPG or COBOL, directly to maintainable runtime-free Java and JS

COBOL is over 60 years old yet still powers 70% of the world's transactions.

And that number shows no sign of slowing down, with an estimated 1 billion lines of new code being generated per year. A recent comprehensive survey commissioned by MicroFocus and published in 2022 estimates that **775-800 billion lines of COBOL code is currently in production.** The previous estimate, before the study, was in the range of **200-300 Billion**[1].

92 percent of respondents of that same survey stated that their organizations' COBOL applications are "strategic to their IT strategy," and application portfolio alignment with new technology is among their key drivers for COBOL modernization.

The problem?

Legacy applications are growing, yet new coders aren't learning legacy languages like COBOL and RPG to keep these systems humming. The average age of the COBOL programmer is well over 40 and they are rapidly leaving the profession. Many important functions will soon reach end-of-life.

The solution:

Transform your legacy application into maintainable, runtime free Java and JS, workload by workload, with state of the art tools like CM evolveIT metaTX that leverages the power of automation and artificial intelligence.

Why Shift from COBOL to Java?

Java is a modern programming language in wide use, offering modern capabilities and performance, supported by a large and growing pool of specialists who can maintain and transform your critical applications.

With a sustainable, iterative approach to modernizing your legacy applications, you can achieve digital transformation and match your IT staffing resources to your application functionality at a pace that is comfortable, risk-free, and sane.

Contents

[1]*OpenText (opentext.com/what-is/cobol*

Making the Case For Java

Legacy languages have been the go-to, trustworthy system for decades, and remain the backbone of many organizations that process large amounts of data. But the tide is rapidly shifting in favor of more modern and versatile languages like Java, and for good reason.

+ **Vast Labor Pool**

Java is versatile, platform-independent, and boasts a large, active community of developers constantly working to improve and update the language.

+ **Boosts Efficiency**

Java allows for writing reusable object-oriented code, which means developers can build upon existing codebases instead of starting from scratch each time.

+ **Builds Resilience**

With Java, you're not tied to a specific hardware or operating system, allowing you to adapt your critical applications to future changes in technology as needed.

+ **Improves Developer Productivity**

IDEs like Eclipse and IntelliJ IDEA provide developers with sophisticated functionalities to identify and fix errors faster, write less code, and automate routine programming tasks.

+ **Streamlines Application Management**

A Java-based system is easier to integrate with modern technologies and platforms, critical for businesses looking to adopt a microservices architecture or expand their suite of applications.

+ **Move your Applications to the Cloud**

Moving to commodity cloud servers can be less expensive to operate than on-premesis midrange or mainframe hardware, with the right amount of preparation and planning.

+ **Enhances Security**

Java is widely recognized for its strong security features, which include a robust permissions system, a security-conscious runtime environment, and standard cryptographic functionalities.

+ **Reduced Cost**

Java can be open sourced, IDE's and compilers are low cost or free, and specialty processors can be utilized on the mainframe to lower processing fees*.

+ **Your Customers Will Love It**

Modern consumers demand fast, efficient, and reliable digital experiences – something that modern Java/JS can deliver much more effectively than COBOL can.

*Cost Savings Case Study: Specialty Processors

Making the move from expensive standard processors to less expensive zIIP processors is just one example of the cost savings that you can realize when moving workloads from COBOL to Java. In the case of IBM monthly licence charges (MLC) costs, any workloads moved to the zIIP aren't counted in the million service units (MSU) metrics that are used to add up your bill, a move that can save a lot of money.

STRATEGIES FOR SUCCESS: ITERATION AND AUTOMATION

Gone are the days of the rip and replace modernization strategy for most companies. According to the previously quoted 2022 MicroFocus survey, 64 percent of respondents have plans to modernize their COBOL applications, and 72 percent see modernization as an "overall business strategy." We find that with most modernization strategies, the two principles that drive a successful project are Iteration and Automation. We have several strategies and tools at our disposal to **iterate** our COBOL to Java projects:

Phased Migration

Migrating your application workload by workload minimizes the risk of disruption to critical operations by giving you the needed breathing room to validate the migration at each stage. It also offers a gradual learning curve for developers new to Java, as they can focus on specific modules rather than dealing with the entire system all at once. This approach provides flexibility, allowing you to prioritize modules based on their criticality or complexity.

Hybrid Environments

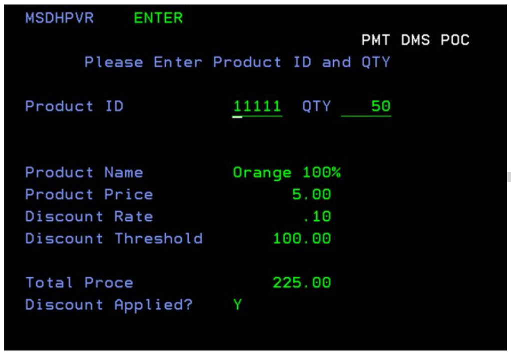
With a hybrid environment, you can leverage the strengths of both programming languages. Existing COBOL code can continue to run smoothly while new functionalities and enhancements can be developed in Java. This approach enables you to take advantage of the robustness and reliability of COBOL while embracing the flexibility and scalability offered by Java.

Software Testing with Automation and AI

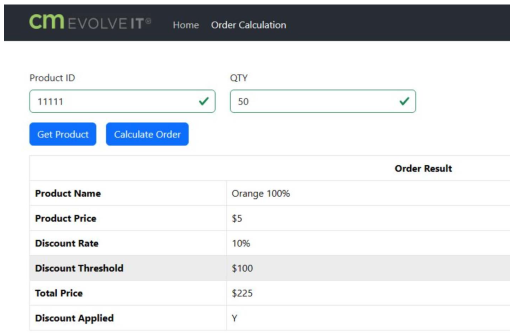
Adopting testing strategies and best practices, powered by automation and AI, at important phases of your COBOL to Java transformation can significantly impact your development process. You'll see improved reliability, quicker issue detection, and smoother deployments. You'll also have a well-oiled machine; everything will run more smoothly, and you can sleep better at night knowing your application won't fall apart at the slightest disturbance in your new transformed environment.

THE POWER OF AUTOMATION

When refactoring your codebase, automation is a powerful ally loaded with high-impact benefits, decreasing costs, time, and risk in one concerted effort. When tuned correctly CM evolveIT meta TX automation will not only minimize end-user training allowing you to utilize your existing staff, it's highly automated at scale and maintainable within standard architectures. You'll be able to plug it into your standard DevOps pipelines and go.

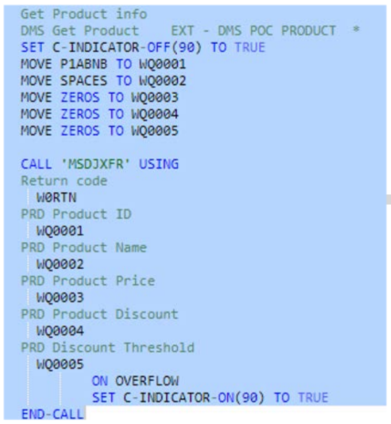


```
MSDHPVR     ENTER
                                          PMT DMS POC
       Please Enter Product ID and QTY

Product ID                11111   QTY    50


Product Name              Orange 100%
Product Price                  5.00
Discount Rate                   .10
Discount Threshold           100.00

Total Proce                  225.00
Discount Applied?         Y
```

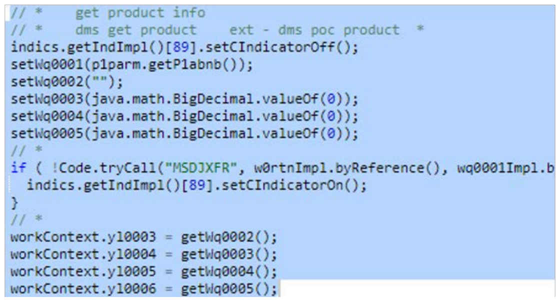


Product ID: 11111
QTY: 50

	Order Result
Product Name	Orange 100%
Product Price	$5
Discount Rate	10%
Discount Threshold	$100
Total Price	$225
Discount Applied	Y



```
Get Product info
DMS Get Product     EXT - DMS POC PRODUCT  *
SET C-INDICATOR-OFF(90) TO TRUE
MOVE P1ABNB TO WQ0001
MOVE SPACES TO WQ0002
MOVE ZEROS TO WQ0003
MOVE ZEROS TO WQ0004
MOVE ZEROS TO WQ0005

CALL 'MSDJXFR' USING
Return code
  W0RTN
PRD Product ID
  WQ0001
PRD Product Name
  WQ0002
PRD Product Price
  WQ0003
PRD Product Discount
  WQ0004
PRD Discount Threshold
  WQ0005
         ON OVERFLOW
         SET C-INDICATOR-ON(90) TO TRUE
END-CALL
```



```
// *    get product info
// *    dms get product    ext - dms poc product  *
indics.getIndImpl()[89].setCIndicatorOff();
setWq0001(p1parm.getP1abnb());
setWq0002("");
setWq0003(java.math.BigDecimal.valueOf(0));
setWq0004(java.math.BigDecimal.valueOf(0));
setWq0005(java.math.BigDecimal.valueOf(0));
// *
if ( !Code.tryCall("MSDJXFR", w0rtnImpl.byReference(), wq0001Impl.b
  indics.getIndImpl()[89].setCIndicatorOn();
}
// *
workContext.y10003 = getWq0002();
workContext.y10004 = getWq0003();
workContext.y10005 = getWq0004();
workContext.y10006 = getWq0005();
```

+ Before Automation

Your green-screen application is trusty and reliable, but lacks the modern features and flexibility you need to compete in today's competitive technical marketplace.

+ After Automation

Your application is refactored into maintainable runtime-free Java accessible by the web, maintainable by a much larger labor pool, and no vendor lock-in is required.

Automated Refactoring Best Practices

Integrate Automation into the Development Process

Rather than treating automation as a separate project, integrate it into your software development process. Continuous monitoring and improvement should be part of this integration. This approach will ensure that automation remains practical and relevant in the long term.

Consider Scalability from the Beginning

Ensure that your refactoring projects can handle future growth. Scalability should be a critical factor in selecting an automated refactoring tool. Choose tools that can handle large databases and continue to perform well as the database grows. Proper database design and performance testing can help prevent scalability issues.

Clearly Define Requirements

In automated software refactoring, distinctly outlining what the system should achieve is crucial. This includes specifying not only the functional aspects but also non-functional ones like performance metrics, security standards, and code maintainability. Unclear requirements can lead to mismatches between the end product and expected outcomes, making it crucial for stakeholders to stay aligned throughout the project lifecycle.

Implement Continuous Integration/Continuous Deployment (CI/CD)

Adopting CI/CD practices ensures that refactoring is a regular part of the development lifecycle. By continuously integrating and deploying code, teams are forced to address refactoring needs more proactively, which helps maintain a clean codebase.

Use Metrics to Track Success

Track the success of your refactoring efforts with appropriate metrics. Measuring factors like the frequency of code integration, the amount of technical debt, and the number of bugs reported can give valuable insights into the effectiveness of your refactoring efforts.

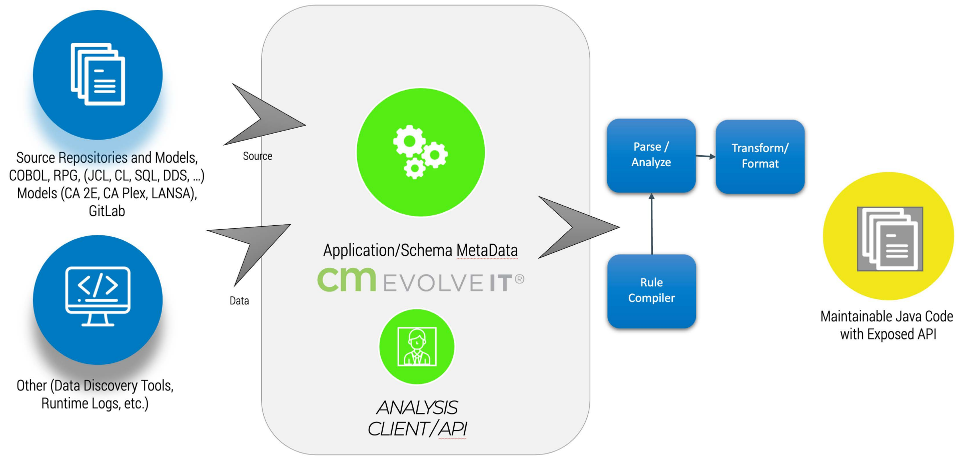
Customize Refactoring Techniques to Fit Project Needs

Understand that not all refactoring techniques suit every project. Tailor your approach to the project's specific needs, taking into account factors such as the size of the codebase, the programming languages in use, and the domain of the application.

Leverage Expertise and Specialized Tools

Engage with experts who have a proven track record in software refactoring and make use of specialized tools designed for refactoring. These resources can provide the necessary expertise and support to tackle complex refactoring tasks effectively.

With the help of CM evolveIT metaTX-AI's automated code refactoring superpower, you can effectively convert your legacy codebases to runtime-free Java and JS. Your applications will operate on the cutting edge, and future developers will be able to get up to speed quickly.


Source Repositories and Models, COBOL, RPG, (JCL, CL, SQL, DDS, ...) Models (CA 2E, CA Plex, LANSA), GitLab
Source
Other (Data Discovery Tools, Runtime Logs, etc.)
Data
Application/Schema MetaData
CM EVOLVE IT®
ANALYSIS CLIENT/API
Parse / Analyze
Transform/ Format
Rule Compiler
Maintainable Java Code with Exposed API

CM evolveIT metaTX-AI provides several features that facilitate refactoring legacy code to Java and JS

Code Analysis

CM evolveIT Meta TX-AI performs an in-depth analysis of the legacy codebase, identifying potential areas for improvement. It examines program structure, data usage, and logic flow, enabling developers to identify opportunities for refactoring.

CodeMetrics

The tool generates comprehensive reports on code quality, complexity, and maintainability. These metrics help developers prioritize refactoring efforts and track the progress of code improvements.

Automated Refactoring

CM evolveIT offers a range of automated refactoring techniques specific to COBOL. From restructuring data structures to improving naming conventions and eliminating dead code, these refactoring actions are applied programmatically, saving valuable development time.

DependencyVisualization

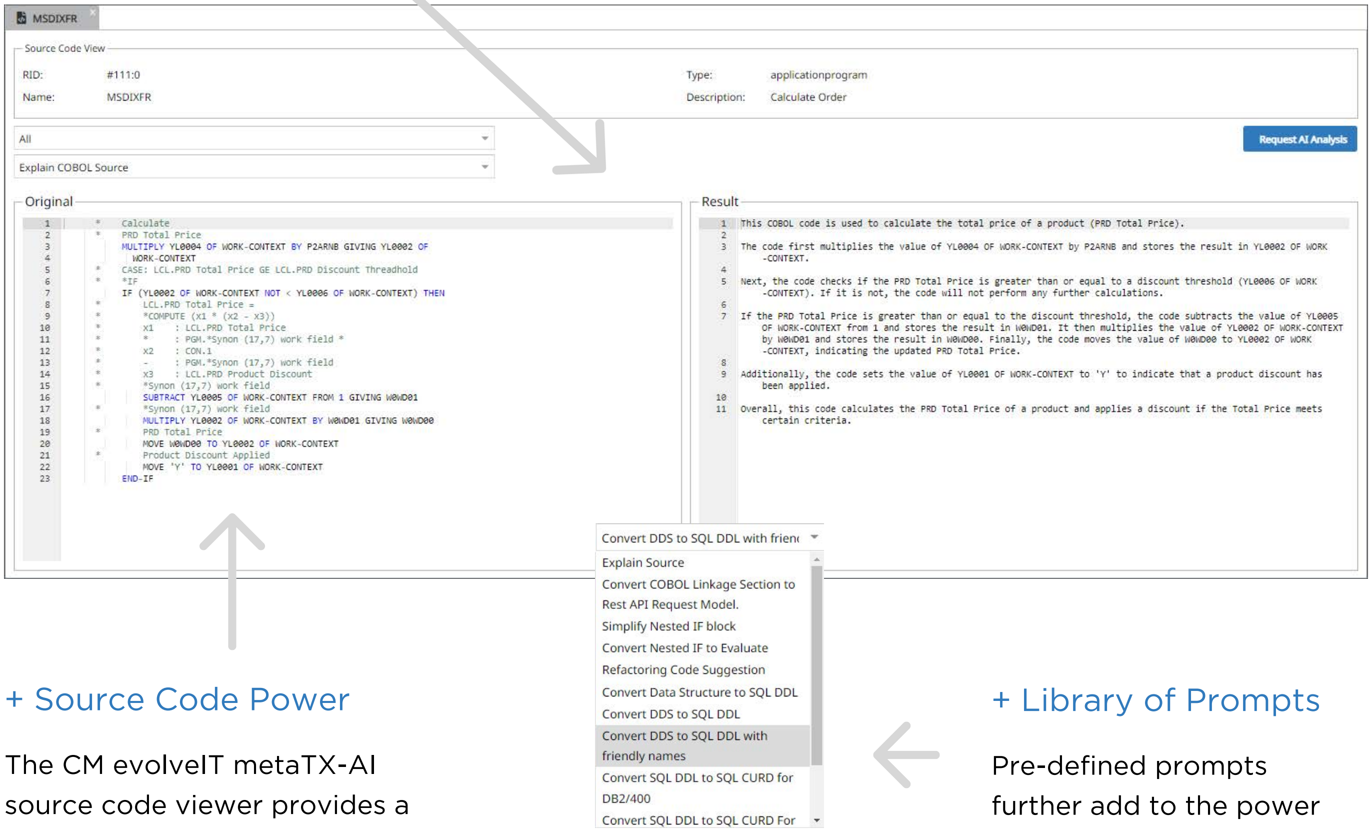
The tool visualizes code dependencies, allowing developers to identify and refactor complex interdependencies. This feature helps to improve code modularity and reduce tight coupling.

NOW WITH CHATGPT INTEGRATION

Generative AI is throwing the software world for a loop with dramatic productivity gains. The promise is real if used intelligently and with skill. The key: inject AI access where the important work is done and focus its power within a carefully selected universe of context-aware prompts.

+ AI Output

Artificial Intelligence output explains and analyzes your existing code, and generates new code as needed.


MSDIXFR
Source Code View
All
Explain COBOL Source
Request AI Analysis
Original
Result
Convert DDS to SQL DDL with frien
Explain Source
Convert COBOL Linkage Section to Rest API Request Model.
Simplify Nested IF block
Convert Nested IF to Evaluate
Refactoring Code Suggestion
Convert Data Structure to SQL DDL
Convert DDS to SQL DDL
Convert DDS to SQL DDL with friendly names
Convert SQL DDL to SQL CURD for DB2/400
Convert SQL DDL to SQL CURD For

RID: #111:0
Name: MSDIXFR
Type: applicationprogram
Description: Calculate Order

```
*    Calculate
*    PRD Total Price
     MULTIPLY YL0004 OF WORK-CONTEXT BY P2ARNB GIVING YL0002 OF
       WORK-CONTEXT
*    CASE: LCL.PRD Total Price GE LCL.PRD Discount Threadhold
*    *IF
     IF (YL0002 OF WORK-CONTEXT NOT < YL0006 OF WORK-CONTEXT) THEN
*        LCL.PRD Total Price =
*        *COMPUTE (x1 * (x2 - x3))
*        x1    : LCL.PRD Total Price
*        *     : PGM.*Synon (17,7) work field *
*        x2    : CON.1
*        -     : PGM.*Synon (17,7) work field
*        x3    : LCL.PRD Product Discount
*        *Synon (17,7) work field
         SUBTRACT YL0005 OF WORK-CONTEXT FROM 1 GIVING W0WD01
*        *Synon (17,7) work field
         MULTIPLY YL0002 OF WORK-CONTEXT BY W0WD01 GIVING W0WD00
*        PRD Total Price
         MOVE W0WD00 TO YL0002 OF WORK-CONTEXT
*        Product Discount Applied
         MOVE 'Y' TO YL0001 OF WORK-CONTEXT
     END-IF
```

```
This COBOL code is used to calculate the total price of a product (PRD Total Price).

The code first multiplies the value of YL0004 OF WORK-CONTEXT by P2ARNB and stores the result in YL0002 OF WORK
    -CONTEXT.

Next, the code checks if the PRD Total Price is greater than or equal to a discount threshold (YL0006 OF WORK
    -CONTEXT). If it is not, the code will not perform any further calculations.

If the PRD Total Price is greater than or equal to the discount threshold, the code subtracts the value of YL0005
    OF WORK-CONTEXT from 1 and stores the result in W0WD01. It then multiplies the value of YL0002 OF WORK-CONTEXT
    by W0WD01 and stores the result in W0WD00. Finally, the code moves the value of W0WD00 to YL0002 OF WORK
    -CONTEXT, indicating the updated PRD Total Price.

Additionally, the code sets the value of YL0001 OF WORK-CONTEXT to 'Y' to indicate that a product discount has
    been applied.

Overall, this code calculates the PRD Total Price of a product and applies a discount if the Total Price meets
    certain criteria.
```

+ Source Code Power

The CM evolveIT metaTX-AI source code viewer provides a direct interface to the vast powers of AI, giving you all the problem-solving benefits of generative artificial intelligence at your fingertips.

+ Library of Prompts

Pre-defined prompts further add to the power by providing the exact queries needed to get the job done quickly and efficiently, focusing the user on the task at hand.

CM First Group Can Help

Our deep experience with legacy enterprise systems puts us in a unique position to help companies reinvent their modernization efforts with RPA. We have the knowledge and real-world experience needed to implement emerging RPA technology effectively and help you target and achieve the highest ROI possible.

For more information, visit **cmfirstgroup.com**.

Contact us for more information or to schedule a demo. Call 888-866-6179 or email us: **info@cmfirstgroup.com.**

About CM First Offerings

CM First's powerful automation tools, augmented by professional services staff with many decades of software engineering and DevOps experience, ensure successful outcomes for even the most demanding modernization projects. Our products and expertise have helped over 400 customers in the public and private sectors reach their desired future state faster and more cost effectively than by using conventional approaches.

CM First software quickly analyzes, documents and re-platforms legacy code bases with minimal errors and rework, including those that are too large and complex for humans to tackle in any reasonable timeframe. The output is immediately usable by all team members, regardless of experience and knowledge of legacy software languages, accelerating application maintenance and modernization projects.

Rethink Modernization

CM First Group
888-866-6179
cmfirstgroup.com

7000 North Mopac Expressway
Plaza 7000, 2nd Floor
Austin, Texas 78731