

Cobol2j

Version 1.7

General Information Manual

Publication number	GC2J-0001-01
Edition	Second Edition (October 2026)
Product	Cobol2j Version 1.7
Language	English

Form-number convention: GC2J-####-## identifies Cobol2j General Information publications. The base number is shared by a language pair; English uses the numeric suffix (GC2J-0001-01); Spanish uses the same base with an ES suffix (GC2J-0001-01ES). Revision of either language increments the trailing two digits together.

Note: Before using this information and the product it supports, read the information in the Notices and Trademarks sections of this edition notice.

Copyright Notice

Copyright (c) 2026 Cobol2j. All rights reserved.

This documentation and the software it describes are proprietary to Cobol2j and are protected by copyright law and applicable international treaties. No license is granted by mere access to this document. Use of the software by a customer is governed exclusively by the license agreement between that customer and Cobol2j. See [LICENSE](#) and [THIRD-PARTY-NOTICES.md](#).

THE SOFTWARE AND THIS DOCUMENTATION ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, EXCEPT TO THE EXTENT THE APPLICABLE LICENSE AGREEMENT PROVIDES OTHERWISE.

Trademarks

IBM, CICS, DFSORT, IMS, and VSAM are trademarks or registered trademarks of International Business Machines Corporation (IBM Corp.) in many jurisdictions worldwide. In this publication those names appear only where the reader must recognize or type them as they are (statements in the customer's programs, utility names, product identifiers). Cobol2j is not affiliated with, sponsored by, or endorsed by IBM. Other product and service names might be trademarks of their respective owners.

Contact: hello@cobol2j.dev

Contents

1. About this manual	1
2. Why migrate	3
3. What Cobol2j is	5
4. Who it is for	7
5. The Mainframe → Cobol2j migration path	9
5.1 Assessment	9
5.2 Migration in the IME	10
5.3 Deployment in the IME	10
5.4 Operation in the new environment	10
5.5 Attached applications (cataloged files as Java repositories)	11
6. The three pillars	13
6.1 EBCDIC byte-image fidelity	13
7. Generated Java and COBOL behavior	15
8. Execution environments and the Kubernetes platform	17
8.1 Batch and Online	17
8.2 Platform 1.7 on Kubernetes	17
8.3 IME and Maven lifecycle	18
9. Internationalization	19
10. Documented product limits	21
11. High-level requirements	23
12. Product modules (summary)	25
13. Related publications	27

1. About this manual

This General Information Manual (GIM) introduces Cobol2j for evaluators, architects, and technical managers. It states what the product is, whom it serves, the **Mainframe → Cobol2j migration path** in five steps, what the product covers at a high level, and where detailed manuals begin.

It is **not** an installation guide or an operations handbook. Those are listed in [§13 Related publications](#).

Facts in this edition reflect Cobol2j **1.7** as documented in the product repository (October 2026).

2. Why migrate

Mainframe applications that still carry core business logic face a structural problem that rewriting from scratch seldom solves cleanly:

- **Skills.** Enterprise COBOL and mainframe operations expertise is scarce and retiring. Hiring and retaining people who can change certified batch, online-transaction, and indexed-file programs at scale has become harder each year.
- **Cost.** Hardware, software licensing, and specialty operations keep rising while the same workloads could run on commodity Java platforms that the organization already operates.
- **Hardware dependency.** The application remains bound to mainframe capacity planning, tooling, and data centers even when the business no longer needs that footprint for those workloads.
- **Risk.** Critical logic lives in programs that few people can safely change. Deferred maintenance, frozen interfaces, and fragile data layouts increase operational and compliance exposure.

Modernizing those applications means moving them to a contemporary runtime **without discarding** the certified logic, the EBCDIC record layouts, or the online-transaction commands and JCL control language the teams already know. Cobol2j is built for that path: rehost on Java 21, Spring Boot, and Kubernetes while preserving mainframe data and program semantics.

3. What Cobol2j is

Cobol2j is a **rehosting platform** for mainframe COBOL online and batch applications on **Java 21 + Spring Boot + Kubernetes**.

It runs, in Java, programs and data that today live on the mainframe: **EBCDIC indexed and sequential files**, **BMS/3270** screens, **COBOL** online and batch programs, and **SQL** data. It is **not** an emulator. It translates mainframe artifacts into native Java and web artifacts while preserving mainframe data semantics (EBCDIC, COMP-3, fixed length, key collation, exact decimal).

Two products share one platform:

Product	Role
Cobol2j Online	Regions that serve terminal transactions (pseudo-conversational online-transaction semantics).
Cobol2j Batch	Job queues that execute original JCL (classes, priorities, spool, console), with Spring Batch under the covers.

COBOL keeps EXEC CICS; JCL stays JCL. That is the syntax the organization already has, and Cobol2j executes it as written.

Version **1.7** delivers the three pillars (data, screens, translator), a multi-user web **IME** (Integrated Migration Environment), both execution environments with transactional **EXEC SQL** unified with file I/O, and installation with **Helm on Kubernetes**.

4. Who it is for

Cobol2j serves organizations that need to:

- Leave mainframe hardware for selected workloads and stop using COBOL as the language of **new** development.
- Keep certified batch, online-transaction, indexed-file, and static SQL logic **without rewriting** it to idiomatic Spring.
- Own the certified Java after migration.
- Query migrated data with ordinary database and BI tools, without a mainframe utility.
- Operate the runtime on **their own** Kubernetes cluster.
- Attach **new** Java applications that read and write the same cataloged files through a Spring Data API and the published data contract.

5. The Mainframe → Cobol2j migration path

The product path from a mainframe portfolio to a running Cobol2j system is five steps. Each step has a documented procedure; this section states the sequence and the boundary of each step. Detail lives in the manuals cited at the end of each subsection.

Step	Name	Where it happens	Outcome
1	Assessment	On premises (<code>java -jar</code> or <code>mvn cobol2j:assess</code>)	Measured migrability of the portfolio; inventory, traffic light, graphs, report and decks
2	Migration in the IME	IME Development perspective (or <code>mvn cobol2j:migrate / certify</code>)	Generated Java and HTML; per-file report; certified ownership of the Java
3	Deployment in the IME	IME Development → Deploy (or <code>mvn cobol2j:deploy</code>)	Classes, screens, JCL, and control cards in the system load library; TRANSID and file catalog on the target region; audited result
4	Operation in the new environment	IME perspectives Systems, Operations, and Security ; Online terminal; batch queue / spool / console; data browser	Day-to-day run of Online, batch, and data without the mainframe
5	Attached applications	New Spring Boot apps using <code>cobol2j-spring-data-vsam</code> and the published data contract	New Java development against the same cataloged files as repositories

The five steps, in order:

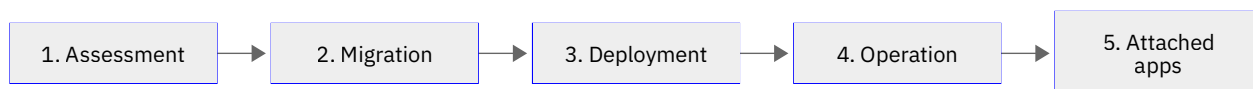


Figure 1.

The Maven goals use the **same migration engine** as the IME (`cobol2j-migration`). After `certify`, the Java belongs to the team and is not regenerated by a later `migrate` unless force options are set. Phase-2 maintenance of certified programs can proceed in any Java IDE; the IME remains the place for **operation** (queue, spool, console, terminal, data).

5.1 Assessment

Before committing to a migration, Cobol2j measures the portfolio **by running the translator in dry-run** on the organization's code — not by heuristic estimation.

Property	Fact
Delivery	Signed <code>java -jar</code> run on premises , or <code>mvn cobol2j:assess</code> .
Network	No network connection is opened. Source never leaves the site for the assessment itself.
Output	Inventory, per-object traffic light (automatic / assisted / blocks / out of scope), graphs, verb census, report and executive/technical decks from one model.
Prose	Deterministic product text from measured fields — not an external language model.

Traffic-light criteria match acceptance used by the project: automatic requires **0 TODO and 0 WARNING**.

Artifacts that define the **environment** (CSD, data-set catalog, SQL DDL, scheduler network, security-manager definitions, SIT, job classes, and similar) are **not** part of the code portfolio and must be requested separately; without them the migrated system does not start. See GC2J-0003-01 Assessment Guide.md.

5.2 Migration in the IME

Migration is performed in the IME **Development** perspective: select the project and run **Migrate**. The same engine used by `mvn cobol2j:migrate` walks copybooks, BMS maps, and COBOL programs and produces the equivalent Java and HTML, plus a per-file report: **GENERATED**, **SKIPPED**, **OVERWRITTEN**, **CERTIFICADO**, or **ERROR**. Unsupported constructs are classified `TODO` (blocks), `WARNING` (translated, may differ from the mainframe), or `INFO`. The full report is written to `migracion.md` / `migracion.json` and shown in the IME migration console. The step-by-step procedure, from preparing the sources to cutover, is in GC2J-0010-01 Application Migration Guide.

The IME also provides:

- A source↔generated comparator (COBOL left, Java right) for program-level review.
- Per-user Git clones with commit, publish, and synchronize.
- Project layout for COBOL, copybooks, BMS, JCL, PROC, and control cards (JCL/PROC/CTL are **not** translated; they are deployed as written).

Certification (`mvn cobol2j:certify` or the equivalent close of phase 1) records ownership of the generated Java. See GC2J-0007-01 User's Manual.md §§5–7 and §13.

5.3 Deployment in the IME

Deploy installs into the **target system** load library (shared volume: local directory or NFS export on Kubernetes): `.class` files, screens, JCL, procedures, and control cards, atomically. Each region and initiator that serves the system detects the change by periodic volume scan and takes it without restart.

Two halves of a deploy do not go to the same place:

- Load-library members are **of the system** (shared by all regions and initiators of that system).
- TRANSID and the file catalog go to the **selected region** only; they must be repeated for a new region that should serve the same transactions.

If a program contains `EXEC SQL`, deploy asks the target system engine to prepare each statement (analyze and resolve names, without executing) — the Cobol2j equivalent of a mainframe `BIND`. Warnings do not block install. Every deploy is audited (who, when, what, to which system and region, with what result). See GC2J-0007-01 User's Manual.md §9 and GC2J-0008-01 System Administration Guide.md §16.3.

5.4 Operation in the new environment

Day-to-day work after deploy uses the IME's **four perspectives** (roles determine which actions appear):

Perspective	Typical roles	Purpose
Development	DESARROLLO, ADMINISTRACION	Project tree, editor, migrate, compare, Git, deploy.
Systems	All four roles; write mainly ADMINISTRACION	Systems, regions and their catalogs, data catalog, initiators.
Operations	OPERACION, ADMINISTRACION; SEGURIDAD read	Start/stop regions and initiators; job queue, spool, console, terminal; denied access.

Perspective	Typical roles	Purpose
Security	Write SEGURIDAD; others read	IME accounts and audit; Cobol2j Security users, groups, and profiles.

Online. From Operations (or the region URL): the HTML terminal with the same protected fields and color attributes as the original BMS map. Keystrokes (ENTER, PF1 – PF24, PA1 – PA3, CLEAR) drive pseudo-conversational online-transaction semantics; session state lives in PostgreSQL (C2JTERMID), so multi-instance regions do not require sticky sessions.

Batch. From Operations: submit a job (paste/upload or internal reader INTRDR); **queue** by CLASS= / PRTY= taken by free initiators; **spool** for DD SYSOUT=; **console** with stable message ids (C2Jnnnn); operator messages (WTOR-style) pause the job until answered from Operations.

Data. The IME **data browser** decodes a catalog file with its view copybook (table or hex modes), loads EBCDIC datasets, exports CSV, and can create ASCII BI views (vascii_*) in the system schema. Cataloged datasets live in the **system catalog**, not in the migration project. See GC2J-0007-01 User's Manual.md §§4, 8, 11, 12 and GC2J-0008-01 System Administration Guide.md §10.

5.5 Attached applications (cataloged files as Java repositories)

After the portfolio runs on Cobol2j, **new** Java/Spring Boot applications can use the same cataloged files without learning COBOL. That path is separate from the Java the translator produces.

Capability	Fact
Dependency	Published module cobol2j-spring-data-vsam (plus cobol2j-data-platform and cobol2j-core).
API shape	VsamRepository is to a cataloged file what JpaRepository is to a table (findById, save, paging types, DataAccessException).
Access path	The application does not open the database. It asks an Online region that already has the file open (configured URL, SYSID, token).
Provisioning	Operations registers the application as a region connection (SYSID + token) in the IME — the same model used for a sister region.
Layout source	Record layout (copybook, length, code page, alternate indexes) comes from the region's file descriptor at startup — not declared in the application class beyond file name and region.
Queries	Key or declared alternate index only. Full SQL analytics use the published data contract : typed relational views in schema bi_<system> (see GC2J-0006-01 §4).
Unit of work	An attached application does not join the region's or batch step's distributed unit of work. Each of its operations commits alone. Inside a region, SQL and the files with RECOVERY (BACKOUTONLY) or ALL commit and roll back together; in a batch step, as on z/OS, SQL is rolled back at an abend and VSAM files are not.

Rule of thumb documented in the Data Access Guide: **the repository to reach and write a record; the published data contract to ask questions.** See GC2J-0005-01 Data Access Guide for Attached Applications.md and GC2J-0006-01 Technical Architecture Guide.md §4.

6. The three pillars

#	Pillar	What it does
1	Data (EBCDIC files)	Stores file records as raw EBCDIC byte images in PostgreSQL (<code>bytea</code>). Decode only on view or export with a copybook.
2	Screens (BMS)	Parses BMS maps and renders HTML5 terminal screens for Online regions.
3	Translator (COBOL → Java)	Ahead-of-time translation: one imperative Java class per program. Unsupported constructs emit a visible <code>// TODO cobol2j</code> marker and fail loudly — never silently wrong Java.

6.1 EBCDIC byte-image fidelity

Migrating mainframe data often fails at the first conversion step. An EBCDIC→ASCII or UTF-8 pass corrupts:

- **Packed decimal (COMP-3)** — two digits per byte, sign in the low nibble.
- **Zoned decimal** — sign embedded in the high nibble of the last byte.
- **Binary (COMP / COMP-4 / COMP-5)** and hexadecimal floating point (HFP, COMP-1 / COMP-2).
- **KSDS key order** — EBCDIC collation differs from ASCII; digits sort differently.

Cobol2j does not convert those fields on ingest. Each file record is stored as the **exact byte image** in PostgreSQL `bytea`. Collation for KSDS keys follows unsigned `bytea` order, which matches EBCDIC key order. Decoding is deferred until a copybook is applied for display, export, or a BI view (`vascii_*`). The same bytes can be re-read under another copybook (`REDEFINES`) without re-migrating the file.

Supporting capabilities at product level include:

- Global data catalog (IDCAMS-style): organization, `RECFM`, record length, key, and alternate indexes independent of the copybook.
- ASCII BI views that decode `bytea` on read.
- The **attached** Spring Data repository API described in §5.5.
- Commit and rollback of SQL and recoverable files together inside a region. In a batch step, SQL is rolled back at an abend and VSAM files are not, as on z/OS.

7. Generated Java and COBOL behavior

The translator emits ordinary imperative Java that continues to behave like COBOL where the business logic depends on it. The following characteristics are documented in the Technical Architecture Guide (GC2J-0006-01) and the product README; they are not aspirational claims.

Characteristic	Behavior in generated Java / runtime
Exact decimal	Arithmetic uses <code>CobolDecimal</code> (scale of the receiving field, truncate or <code>ROUNDED</code>). Generated code does not use <code>double</code> for COBOL numerics.
COMP-3 and related USAGE	Packed decimal, zoned, binary, and hexadecimal floating point (COMP-1/COMP-2) are handled by the EBCDIC codecs; values surface as <code>BigDecimal</code> at the Java boundary.
Physical record image	Field writes keep the physical byte image synchronized; group <code>MOVE</code> works on raw bytes with mainframe padding semantics.
1-based table indexes	<code>OCCURS</code> subscripts follow COBOL indexing (1-based). Out-of-range access follows mainframe <code>NOSSRANGE</code> -style behavior within the data area, with a logged warning.
Online-transaction semantics	Translated programs retain <code>EXEC CICS</code> (BMS, <code>COMMAREA</code> pseudo-conversation, <code>EIB</code> , <code>LINK/XCTL</code> , file I/O, <code>TS/TD</code> queues, <code>ENQ/DEQ</code> , <code>SYNCPOINT</code> , and related commands).
JCL semantics	Batch runs the original JCL : <code>DD</code> , <code>SYSIN</code> , <code>PARM</code> , <code>IF/COND</code> , <code>PROC/INCLUDE</code> /symbolics, <code>GDG</code> , and common utilities, with a mainframe-style job queue, spool, and console.
EXEC SQL	Static and dynamic SQL with host-variable fidelity; <code>COMMIT/ROLLBACK</code> atomic with the I/O of recoverable files inside a region; in a batch step the SQL is rolled back at an abend and VSAM files are not, as on z/OS.

Unsupported constructs are marked and fail visibly: the translator emits a `TODO`, `WARNING`, or `INFO` marker and the per-file migration report lists every one. What is supported, construct by construct, is in the Cobol2j Reference, GC2J-0009-01 Cobol2j Reference.md; its §14 lists the limits.

8. Execution environments and the Kubernetes platform

8.1 Batch and Online

Environment	Summary
Batch	Executes original JCL: DD, SYSIN, PARM, IF / COND , PROC/INCLUDE/symbolics, GDG, utilities (IEFBR14, IEBGENER, DFSORT, IDCAMS, and others). Provides a mainframe-style job queue, spool, and operator console.
Online	Tomcat-based region: web terminal, TRANSID transactions, hot program/file catalogs, pseudo-conversational state in PostgreSQL (multi-instance capable; sticky session not required for that state).

8.2 Platform 1.7 on Kubernetes

A production-style installation uses Helm:

- Chart `cobol2j-plataforma` (once per install) and chart `cobol2j-sistema` (once per system).
- Namespace per system; end-to-end TLS at Ingress; shared `ReadWriteMany` volume as PDS truth.
- The system agent acts as a Kubernetes controller (Server-Side Apply for region and initiator Deployments).
- Value overlays for EKS, AKS, and GKE.
- Regions default to elastic HPA (typically 1–3 replicas).

How the platform chart, system charts, and shared volume relate:

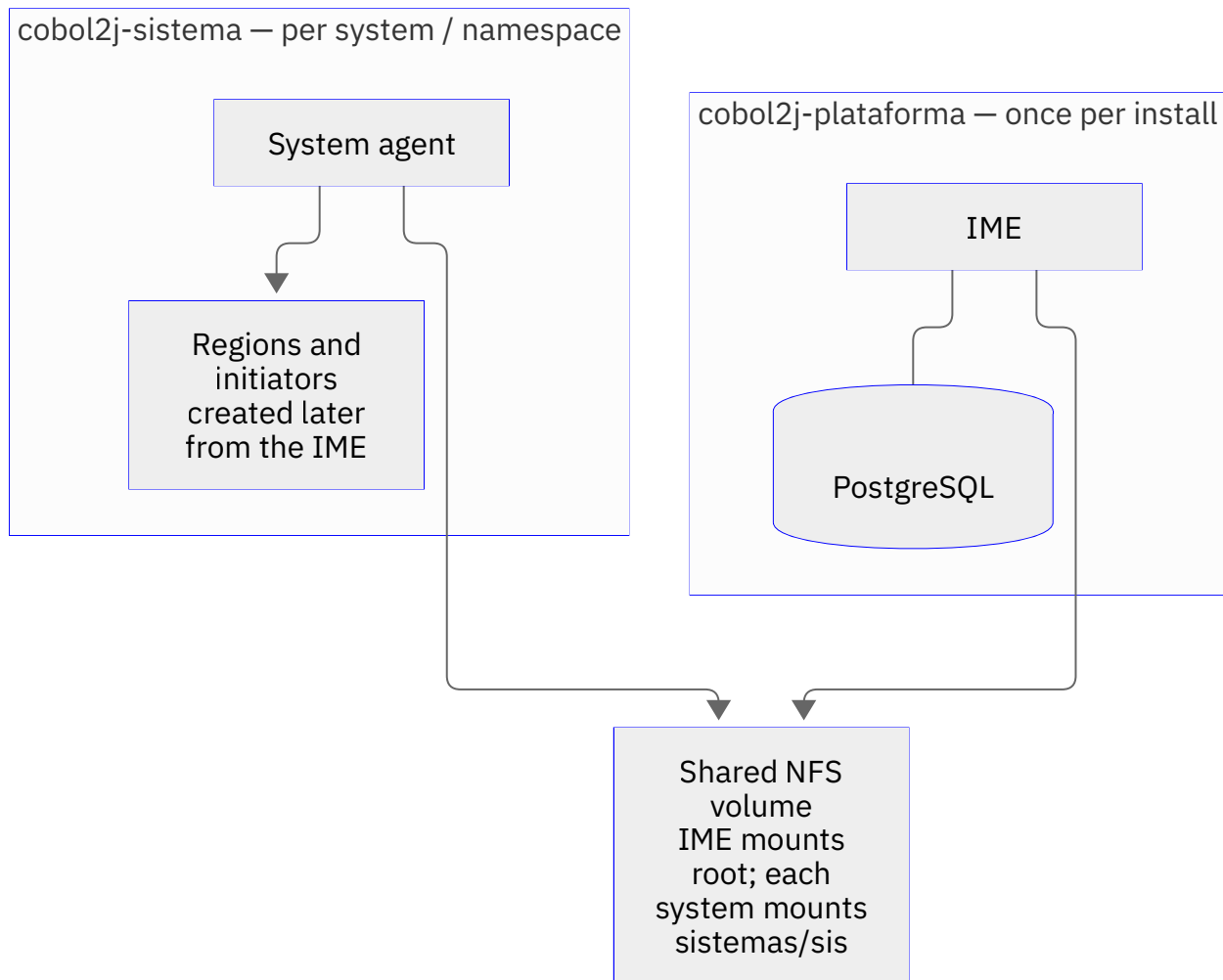


Figure 2.

Day-to-day install and operations are described in the Installation Guide (GC2J-0002-01) and the System Administration Guide (GC2J-0008-01), not here. The migration path in §5 assumes a platform already installed.

8.3 IME and Maven lifecycle

The **IME** is the multi-user web migration and operations environment whose four perspectives are summarized in §5.4.

The Maven plugin covers `cobol2j:assess`, `cobol2j:migrate`, `cobol2j:certify`, and `cobol2j:deploy` for CLI/CI without the IME UI. After `certify`, the Java is owned by the team and is not regenerated by `migrate`.

9. Internationalization

Product surface language is **English by default, Spanish on request** (`--lang es` / `-Dcobol2j.lang=es` / `assessment -Dcobol2j.assess.lang=es`). One language dimension (`en` | `es`) covers assessment deliverables, translator markers forwarded into reports, and IME UI where localized.

Level names `WARNING` and `INFO` are not renamed (legacy assessment runs that used `AVISO` still open).

This GIM is published in both languages as form numbers GC2J-0001-01 (EN) and GC2J-0001-01ES (ES).

10. Documented product limits

The following are **documented product decisions**:

Limit	Meaning
IMS	Hierarchical database and transaction manager: out of scope (0%) unless a specific engagement requires it.
FTP	Utility not implemented; decks that require it do not run (explicit failure / out of scope).
Customer Assembler	Subprograms in Assembler are out of scope by decision (for example COBDATFT and MVSWAIT in the CardDemo sample application).
Attached application ≠ same UOW	A new Spring Data attached application talks to a region for file I/O; it does not join the region's or batch step's unit of work. SQL and recoverable files commit together inside a region (GC2J-0009-01 §7.4).
Long-tail constructs	Unsupported items emit TODO/WARNING/INFO, listed in the per-file migration report.

Multi-instance Online regions also document limits around process-local CWA, `START ... TERMID` screen push, and non-migrating conversational tasks — see the Technical Architecture Guide (GC2J-0006-01).

11. High-level requirements

Minimum environment to build and run Cobol2j (detail in the Installation Guide, GC2J-0002-01):

Component	Level
JDK	21 (full JDK; deploy compiles with <code>javac</code>)
Maven	3.9+
PostgreSQL	16+
Docker	For integration tests (Testcontainers) and image builds
Kubernetes	Cluster with Helm for a full install; <code>kind</code> / Docker Desktop suffice for local trials

Stack summary: Java 21, Spring Boot 4 / Spring Framework 7, PostgreSQL, Kubernetes (Helm, fabric8 client), cert-manager, shared RWX volume, Thymeleaf + HTMX (IME).

Reactor groupId `dev.cobol2j`, version **1.7.0** for this edition.

12. Product modules (summary)

Sixteen Maven modules. High-level map:

Area	Modules (examples)
Data & codecs	cobol2j-core, cobol2j-data-platform, cobol2j-dataview, cobol2j-spring-data-vsam
Screens	cobol2j-screen-platform
Translate & migrate	cobol2j-translator, cobol2j-migration, cobol2j-maven-plugin
Runtime	cobol2j-runtime, cobol2j-batch-runtime, cobol2j-online-runtime
Platform	cobol2j-kubernetes
Assess	cobol2j-assessment, cobol2j-assessment-cli
IME / examples	cobol2j-ime, cobol2j-examples

Full module responsibilities: GC2J-0006-01 Technical Architecture Guide.md.

13. Related publications

Form / path	Title	Audience
GC2J-0001-01ES / GC2J-0001-01ES Manual de Información General.md	Manual de Información General (Spanish)	Same audience, Spanish
GC2J-0002-01 / GC2J-0002-01 Installation Guide.md	Installation Guide	Platform installers
GC2J-0002-01ES / GC2J-0002-01ES Guía de instalación.md	Guía de instalación (Spanish)	Same audience, Spanish
GC2J-0003-01 / GC2J-0003-01 Assessment Guide.md	Assessment Guide	Who runs the assessment
GC2J-0003-01ES / GC2J-0003-01ES Guía de assessment.md	Guía de assessment (Spanish)	Same audience, Spanish
GC2J-0004-01 / GC2J-0004-01 Security Administrator's Guide.md	Security Administrator's Guide	Security administrators
GC2J-0004-01ES / GC2J-0004-01ES Guía del administrador de seguridad.md	Guía del administrador de seguridad (Spanish)	Same audience, Spanish
GC2J-0005-01 / GC2J-0005-01 Data Access Guide for Attached Applications.md	Data Access Guide for Attached Applications	Authors of new Java/Spring Boot attached apps
GC2J-0005-01ES / GC2J-0005-01ES Guía de acceso a datos para aplicaciones adjuntas.md	Guía de acceso a datos para aplicaciones adjuntas (Spanish)	Same audience, Spanish
GC2J-0006-01 / GC2J-0006-01 Technical Architecture Guide.md	Technical Architecture Guide	Maintainers and extenders
GC2J-0006-01ES / GC2J-0006-01ES Guía de arquitectura técnica.md	Guía de arquitectura técnica (Spanish)	Same audience, Spanish
GC2J-0007-01 / GC2J-0007-01 User's Manual.md	User's Manual	Programmers and operators (IME)
GC2J-0007-01ES / GC2J-0007-01ES Manual de usuario.md	Manual de usuario (Spanish)	Same audience, Spanish
GC2J-0008-01 / GC2J-0008-01 System Administration Guide.md	System Administration Guide	System administrators and system programmers
GC2J-0008-01ES / GC2J-0008-01ES Guía de administración del sistema.md	Guía de administración del sistema (Spanish)	Same audience, Spanish
GC2J-0009-01 / GC2J-0009-01 Cobol2j Reference.md	Cobol2j Reference	Migrators, programmers of the generated Java, and JCL operators
GC2J-0009-01ES / GC2J-0009-01ES Referencia de Cobol2j.md	Referencia de Cobol2j (Spanish)	Same audience, Spanish
GC2J-0010-01 / GC2J-0010-01 Application Migration Guide.md	Application Migration Guide	Application programmers and migration engineers
GC2J-0010-01ES / GC2J-0010-01ES Guía de migración de aplicaciones.md	Guía de migración de aplicaciones (Spanish)	Same audience, Spanish

Form / path	Title	Audience
GC2J-0011-01 / GC2J-0011-01 Messages and Codes.md	Messages and Codes	Operators, programmers, and administrators who look up a message or a code
GC2J-0011-01ES / GC2J-0011-01ES Mensajes y códigos.md	Mensajes y códigos (Spanish)	Same audience, Spanish
deploy/README.md	Deploy charts	Cluster preparation

IBM, CICS, DFSORT, IMS, and VSAM are trademarks or registered trademarks of International Business Machines Corp.

End of GC2J-0001-01