ibi Summit 2026

Connecting Modern Applications to the Mainframe

Walter Brengel, Product Manager
Angie Hildack, Director of Product Management



© Copyright 2026 Cloud Software Group, Inc.

Confidentiality & Disclaimer

The information in this document is confidential information of Cloud Software Group, Inc. and/or its affiliates. Use, duplication, transmission, or republication for any purpose without the prior written consent of Cloud Software Group, Inc. is expressly prohibited.

This document (including, without limitation, any product roadmap or statement of direction data) illustrates the planned testing, release and availability dates for Cloud Software Group, Inc. products and services. This document is provided for informational purposes only and its contents are subject to change without notice. Cloud Software Group, Inc. makes no warranties, express or implied, in or relating to this document or any information in it, including, without limitation, that this document, or any information in it, is error-free or meets any conditions of merchantability or fitness for a particular purpose.

The material provided is for informational purposes only, and should not be relied on in making a purchasing decision. The information is not a commitment, promise or legal obligation to deliver any material, code, or functionality. The development, release, and timing of any features or functionality described for our products remains at our sole discretion.

During the course of this presentation, Cloud Software Group, Inc. or its representatives may make forward-looking statements regarding future events Cloud Software Group, Inc.'s future results or our future financial performance. These statements are based on management's current expectations. Although we believe that the expectations reflected in the forward-looking statements contained in this presentation are reasonable, these expectations or any such forward-looking statements could prove to be incorrect and actual results or financial performance could differ materially from those stated herein. Cloud Software Group, Inc. does not undertake to update any forward-looking statement that may be made from time to time or on its behalf.

© Copyright 2024 Cloud Software Group, Inc.

Connecting Modern Applications to the Mainframe

A JDBC walkthrough with ibi Open Data Hub from a 77-line demo to production Java & Python.

Key Highlights:

- **77 Lines of Java:** Standard java.sql, no framework required.
- **1 Driver JAR:** jlink_standalone.jar handles the connection.
- **ODH on z/OS:** The mainframe data access layer supporting VSAM, DB2, and IMS.
- **Reachable via one contract:** Standard SQL over mainframe data sources.

© Copyright 2026 Cloud Software Group, Inc.

The Reference Implementation Executes in Five Steps

The java_to_odh.java program demonstrates the essential life cycle of a mainframe data request.

The process follows the exact shape you would embed inside any enterprise Java service.

1. **Register the Driver**
   The JLink driver is registered so JDBC recognizes jdbc:jlink:// URLs.

2. **Connect to ODH**
   A connection is opened to Open Data Hub on ibimvs.tibco.com:15120 using standard credentials.

3. **Execute SQL**
   A single SQL query is executed against the qavsm01 synonym.

4. **Iterate Results**
   The program fetches 50 rows, iterating the ResultSet and reading the F1SSN column.

5. **Clean Up**
   Resources are closed in reverse order within a finally block.

© Copyright 2026 Cloud Software Group, Inc.

The Data Access Stack Consists of Three Distinct Layers

The architecture between the Java line of code and the physical VSAM record is broken down into three specific layer responsibilities.

01

Client Layer

Java App

This runs on any JVM, whether on a laptop, in a container, or within an application server.

JLink Driver

The ibi.jlink.EdaDriver provided by jlink_standalone.jar. This layer opens the TCP connection and speaks the EDA wire protocol.

02

Gateway Layer

ODH on z/OS

Open Data Hub on the mainframe. It listens on port 15120, authenticates the user, parses the SQL, and invokes the correct adapter.

03

Storage Layer

Data Source

The physical VSAM dataset that the qavsm01 synonym ultimately resolves to.

© Copyright 2026 Cloud Software Group, Inc.

Open Data Hub Acts as a Bridge for Standard SQL Queries

Open Data Hub (ODH) is the ibi data access server that lives natively on the z/OS mainframe. It is the bridge that lets a modern Java or Python client on a laptop query legacy mainframe data using standard SQL, without needing any mainframe-native runtime on the client side.

Core Architecture:

- **SQL Surface:** The runtime host on ibimvs.tibco.com features a TCP listener on port 15120 that accepts JLink and EDA clients, allowing them to speak standard SQL over mainframe data.
- **Adapters:** ODH manages the specific connections to underlying data stores, including VSAM, DB2, IMS, and ADABAS.
- **Execution:** It authenticates against the mainframe security profile, resolves table names through a metadata layer, and routes the query to the right data-source adapter. Everything on the mainframe side of the JDBC call happens strictly inside ODH.

© Copyright 2026 Cloud Software Group, Inc.

Open Data Hub Provides Four Core Capabilities for Data Access

Platform Value

Open Data Hub serves as the point of leverage by turning the mainframe into a first-class, SQL-addressable data source.

Data engineers can publish synonyms once, and any downstream client can consume them through the same contract.

Unified SQL Surface: Provides one SQL dialect over many sources, meaning clients do not need to know if the target is VSAM, DB2, or IMS.

Broad Adapters: Offers extensive coverage across mainframe stores, including VSAM, DB2 for z/OS, IMS, ADABAS, CA-Datacom, IDMS, and sequential files.

Metadata Layer: Utilizes master files and synonyms to map friendly table names to physical datasets and their respective record layouts. This absorbs schema and physical-storage changes, insulating the clients.

Enterprise Security: Ties directly into RACF, ensuring every session is authorized against the trusted mainframe security profile.

© Copyright 2026 Cloud Software Group, Inc.

Open Data Hub Deploys as a Standard z/OS Server

Operationally, ODH behaves like a well-behaved z/OS server rather than a foreign runtime. It is managed by JES and operators with a predictable footprint.

Deployment Shape

Runs as a started task and address space. It requires only one long-running server address space per ODH instance, making it easy to size, monitor, and restart.

Concurrency Model

Operates as a multi-user threaded environment, supporting many concurrent client sessions within one address space.

Native Integration

Respects Workload Manager (WLM) priorities and utilizes UNIX System Services (USS).

Configuration

Uses an EDACONF/EDAHOME layout that separates the product installation from customer configuration. Server activity is written to EDAPRINT logs, making sessions, timings, and errors highly observable.

© Copyright 2026 Cloud Software Group, Inc.

The Metadata Layer Isolates Clients from Physical Storage

The metadata layer is the point of leverage that makes SQL over VSAM possible. It ensures one authoritative definition per data source, meaning every client sees the same fields, types, and semantics.

Synonym

The Contract

qavsm01

The friendly name that clients actually query. In this demonstration, the synonym is **qavsm01**. It acts as the contract.

Master File

The Schema

Record Layout

Defines the record layout, acting as the schema. It specifies field names, data types, positions, and formats. It is governed by data owners on the mainframe.

Access File

The Pointer

Physical Binding

The physical binding that serves as the pointer. It contains the actual dataset name and adapter-specific options. If the dataset name changes, only the Master and Access files need updating.

© Copyright 2026 Cloud Software Group, Inc.

Adapters Isolate Mainframe-Specific Access Logic

Adapter Architecture

Open Data Hub uses a specific adapter architecture to speak to each mainframe data source.

When an SQL request arrives, a router performs a lookup by synonym and hands off to the appropriate adapter, which handles the mainframe-native call.

Supported Adapters

- **VSAM:** Keyed and entry-sequenced files (used for qavsm01 demo).
- **DB2 for z/OS:** Relational adapter; pushes SQL directly to DB2.
- **IMS:** Provides DB and DC access to IMS segments.
- **Additional:** ADABAS, IDMS, flat/sequential files, VSAM RRDS, and generation datasets.

Provides uniform SQL semantics across sources, allowing federated queries to join data (e.g., VSAM & DB2) via a single SQL statement.

© Copyright 2026 Cloud Software Group, Inc.

Open Data Hub Supports Multiple Client Protocols

While JDBC is used in the reference implementation, ODH features one engine with many client contracts.

All protocols land at the same ODH engine, resolve the same synonyms, and invoke the same adapters.

The choice of protocol is entirely a client-side concern; the mainframe view of the data remains unchanged.

Supported Client Protocols

JDBC

Used for Java clients. This demo uses the JLink driver over TCP.

ODBC

Used for Windows and BI tools, including Excel, Power BI, Tableau, and .NET applications.

REST / HTTP

Designed for modern integrations like web apps, mobile, and cloud services, requiring no driver installation.

© Copyright 2026 Cloud Software Group, Inc.

The Demo Executes SQL Against a VSAM Synonym

In this example, the **qavsm01** synonym routes directly to the VSAM adapter. The Java driver never touches VSAM directly; it only ever talks to ODH.

```
static final String DB_URL = "jdbc:jlink://ibimvs.tibco.com:15120";
String sql = "select F1SSN from qavsm01 fetch first 50 rows only;";
```

01. SQL Parsing

ODH receives the SQL on port 15120 and parses it securely.

02. VSAM Binding

It looks up qavsm01 in the master-file catalog and finds it bound to the VSAM adapter.

03. Data Fetch

The VSAM adapter opens the underlying dataset, walks the records, and returns the first 50 rows.

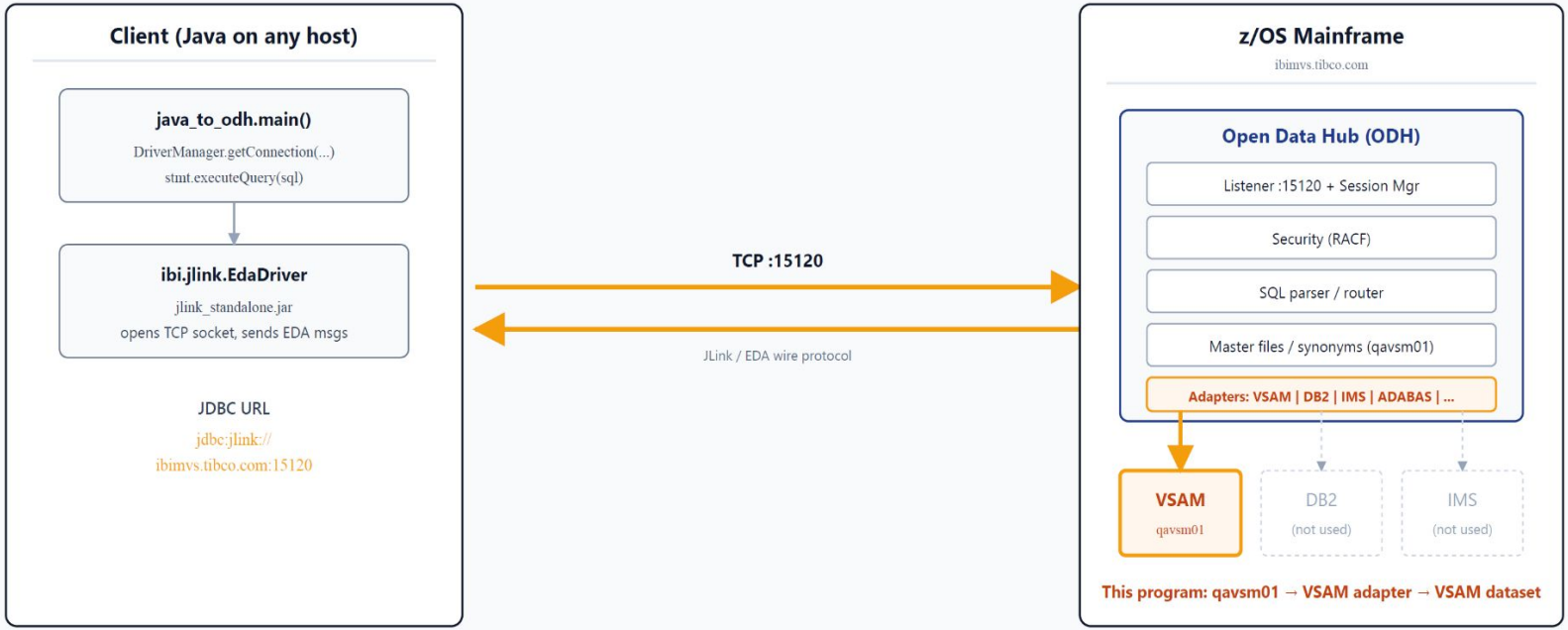
04. Projection

F1SSN is one field in the record layout; ODH projects only that specific field back to the client.

© Copyright 2026 Cloud Software Group, Inc.

Architecture — How It All Flows

From the Java main() on a laptop to the VSAM dataset on z/OS



© Copyright 2026 Cloud Software Group, Inc.

The Architecture Separates Client Calls from Mainframe I/O

The architecture flows from the Java main() method on a laptop to the physical VSAM dataset on z/OS.
All mainframe I/O happens inside ODH; the client only ever talks to the listener socket.

1. Client Side

Environment
Java application running on local laptop.

Execution
Executes java_to_odh.main() to establish standard programmatic flow.

Database Engine
Calls DriverManager.getConnection(...) and invokes query via stmt.executeQuery(sql).

Driver Details
ibi.jlink.EdaDriver loaded from jlink_standalone.jar opens a TCP socket to transmit EDA protocol messages.

2. Network Layer

Protocol
Communication travels end-to-end via the high-performance JLink / EDA wire protocol.

Port Configuration
Establishes communication directly over TCP port **15120**.

Connection Endpoint
Target URL connection string:
jdbc:jlink://ibimvs.tibco.com:15120

3. Server Side

Target Platform
z/OS Mainframe (ibimvs.tibco.com)

Ingress Processing
JES started task executes the ODH Listener and Session Manager to receive requests.

Security & Parsing
Validates credentials via RACF, runs the SQL parser/router, and uses Master Files to resolve target table qavsm01.

Physical Storage I/O
The dedicated VSAM adapter completes direct read operations on the physical VSAM dataset.

© Copyright 2026 Cloud Software Group, Inc.

Setup Relies on Standard Java SQL Imports

Class-Level Declarations

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class java_to_odh {
    static final String JDBC_DRIVER = "ibi.jlink.EdaDriver";
    static final String DB_URL      = "jdbc:jlink://ibimvs.tibco.com:15120";
    static final String USER        = "admin";
    static final String PASS        = "admin";
```

Portable Java SQL Integration

The application code remains completely portable. Every import is resolved from standard **java.sql.*** packages, meaning vendor-specific code lives entirely within the database driver JAR.

Implementation Details

- **URL Structure:** Follows the `jdbc:` prefix, followed by the **jlink:** sub-protocol claimed by EdaDriver, targeting the mainframe host and port.
- **Authentication:** Credentials are authenticated by ODH against a mainframe security profile, entirely transparent to the local JVM.
- **Production Security:** While hard-coded credentials are used in this scratch demo, production deployments should utilize a secrets vault or connection pool.

© Copyright 2026 Cloud Software Group, Inc.

Setup Relies on Standard Java SQL Imports

Class Declarations

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class java_to_odh {
static final String JDBC_DRIVER = "ibi.jlink.EdaDriver";
static final String DB_URL = "jdbc:jlink://ibimvs.tibco.com:15120";
static final String USER = "admin";
static final String PASS = "admin";
}
```

Portable Java SQL Integration

The application code remains completely portable. Every import is resolved from standard `java.sql.*` packages, meaning vendor-specific code lives entirely within the database driver JAR.

Implementation Details

URL Structure: Follows the `jdbc:` prefix, followed by the `jlink:` sub-protocol claimed by EdaDriver, targeting the mainframe host and port.

Authentication: Credentials are authenticated by ODH against a mainframe security profile, entirely transparent to the local JVM.

Production Security: While hard-coded credentials are used in this scratch demo, production deployments should utilize a secrets vault or connection pool.

© Copyright 2026 Cloud Software Group, Inc.

Registering the Driver and Opening the Session

The application establishes a TCP session with the mainframe through a single connection call inside the main() method.

CONNECTION CODE

```
Class.forName(JDBC_DRIVER);

System.out.println(
"Connecting to database..."
);

conn = DriverManager.getConnection(
DB_URL, USER, PASS
);
```

UNDER THE HOOD

1. **Class.forName** forces the JVM to load **EdaDriver**. Its static initializer calls *DriverManager.registerDriver(this)*.
2. **DriverManager.getConnection** walks registered drivers, matches *jdbc:jlink://*, and delegates the call.
3. The driver opens a TCP socket to **ibimvs.tibco.com:15120**, performs the ODH handshake, sends credentials, and receives a session token.
4. The returned **Connection** object wraps the socket and session for creating future database Statements.

© Copyright 2026 Cloud Software Group, Inc.

Executing Queries and Iterating the ResultSet

Query execution and result iteration use standard JDBC methods. The exact same code shape would work against any ODH synonym, regardless of the underlying data source.

Java Code Snippet

```
stmt = conn.createStatement();
String sql = "select F1SSN from qavsm01
fetch first 50 rows only;";
rs = stmt.executeQuery(sql);
while (rs.next()) {
  String firstName = rs.getString("F1SSN");
  System.out.println("F1SSN: " +
  firstName);
}
```

Synonym Querying

qavsm01 is queried as an ODH synonym bound to a VSAM dataset, not as a raw dataset name.

DB2 Fetch Limits

fetch first 50 rows only is DB2-style SQL that caps the result server-side to save network bandwidth.

Data Types Mapping

F1SSN is mapped seamlessly directly to a standard Java String upon retrieval.

Row Prefetching

rs.next() pulls the next row from the driver's fetch buffer, minimizing network overhead and adding round-trips only as needed.

© Copyright 2026 Cloud Software Group, Inc.

Deterministic Cleanup Maintains Healthy Session Counts

Proper error handling and resource release are critical.

Closing the Connection tears down the TCP socket to ODH and releases the mainframe session.

CLEANUP CODE STRUCTURE

```
} catch (SQLException se) {
se.printStackTrace();
} catch (Exception e) {
e.printStackTrace();
} finally {
try { if (rs != null) rs.close(); }
catch (SQLException e) { ... }
try { if (stmt != null) stmt.close(); }
catch (SQLException e) { ... }
try { if (conn != null) conn.close(); }
catch (SQLException e) { ... }
}
```

IMPLEMENTATION NOTES

Dual Catch Scopes

SQLException handles relational database/server issues, while **Exception** safely intercepts generic framework errors like ClassNotFoundException.

Sequential Resource Tear-Down

Resources are closed in precise reverse-allocation order: **ResultSet Statement Connection**. Wrapping each step independently ensures a single cleanup failure never breaks downstream closures.

Modern try-with-resources Alternative

For production environments, refactoring the block to use Java's native **try-with-resources** statement achieves the same robust safety guarantees in significantly fewer, cleaner lines of code.

© Copyright 2026 Cloud Software Group, Inc.

The Core JDBC Contract Scales to Modern Java Stacks

Unified Integration Contract

From a 77-line demo to a Spring Boot microservice, the ODH connection contract is identical. The framework choices on the client side do not change the mainframe contract.

SHARED CONNECTION SPECIFICATION

URL: `jdbc:jlink://ibimvs.tibco.com:15120`

Artifact: `jlink_standalone.jar`

Enterprise Java Patterns

01

Spring Boot + JdbcTemplate

Used in enterprise applications to configure ODH as a DataSource and inject repositories into any service.

02

HikariCP Connection Pooling

Essential for production performance. Because ODH sessions are expensive to open, pooling is required at scale.

03

Virtual Threads (Java 21+)

Supports high-concurrency services by offloading blocking JDBC calls without exhausting platform threads.

04

Kafka Connect & Streaming

Enables real-time integration to publish query results to topics and feed downstream analytics.

© Copyright 2026 Cloud Software Group, Inc.

Data Science Teams Access the Mainframe via the Same Endpoint

Data science and analytics teams reach the mainframe through the exact same door as application developers. They use the same ODH endpoint and standard Python tools.

■ JayDeBeApi

Reuses the `jlink_standalone.jar` by launching a JVM in-process to provide JDBC from Python.

■ pyodbc

Acts as an ODBC bridge that talks to ODH natively without requiring a JVM.

■ pandas.read_sql()

Loads a ResultSet directly into an analytics-ready DataFrame for exploration or export.

■ Jupyter Notebooks

Enables interactive workflows with live mainframe data using the same synonym and VSAM records.

mainframe_connection.py

```
conn = jaydebeapi.connect(
"ibi.jlink.EdaDriver",
"jdbc:jlink://ibimvs.tibco.com:15120",
["admin", "admin"],
"jlink_standalone.jar"
)

df = pd.read_sql(
"select F1SSN from qavsm01
 fetch first 50 rows only",
conn
)
```

© Copyright 2026 Cloud Software Group, Inc.

Running the Example

One command line to compile, one to run

run_example.sh

```
# COMPILE & RUN
javac -cp "jlink_standalone.jar;." java_to_odh.java
java -cp "jlink_standalone.jar;." java_to_odh

# EXPECTED OUTPUT
Connecting to database...
Creating statement...
Fetching data...
F1SSN: 123-45-6789
F1SSN: ...
Goodbye!
```

THE TAKEAWAY

ODH is the Point of Leverage

One well-defined synonym on the mainframe is consumable by any modern client (Java, Python, BI, notebooks) via standard SQL.

Unified Connection Contract

The 77-line example on your laptop and a production Spring Boot service use the exact same connection contract.

Portable Architecture

Data engineers keep the metadata; developers keep the code portable; the mainframe stays authoritative.

© Copyright 2026 Cloud Software Group, Inc.



© Copyright 2024 Cloud Software Group, Inc.