



Raincode 360 User Guide

Version 4.0.0

14-06-2026



Raincode 360 topics

1. [Developer Experience](#)
2. [Raincode Legacy Compilers](#)
3. [Raincode Console](#)
4. [Raincode Batch: JCL Interpreter](#)
5. [zBridge](#)
6. [CICS Emulation](#)
7. [Caching](#)
8. [IMSql](#)
9. [Cloud Native Deployment](#)
10. [Expert mode](#)

What is Raincode 360?

- An advanced showcase
- A non-trivial application
 - Showcasing advanced features of the Raincode product suite
- Includes:
 - Full source code and development projects
 - Data
 - Tooling
 - This presentation
- Deployed as an Azure Resource Manager Template
- Demos are hosted in a GitLab repository (<https://gitlab.phidani.be/raincodepublic/raincode-360.git>)
- Raincode 360 demos are located at C:\Raincode360\Repositories\Demos
- Raincode Product Manuals are available at <https://www.raincode.com/docs/>

List of installed resources on Raincode 360

- Raincode product suite
 - Raincode Legacy Compilers
 - Raincode Visual Studio Integration
 - Raincode QIX Emulator
 - Raincode Batch
 - Raincode IMSql
 - Raincode RadaR
 - Raincode zTrieve
- System tools
 - [PowerShell](#)
 - [GNU Make](#)
- Development tools
 - [Visual Studio](#) 2026 Community Edition
 - [Git](#) for Windows
- Terminal emulators
 - [wc3270](#)
- PowerShell Modules
 - [Az](#)
 - [SqlServer](#)
 - [VSSetup](#)
- Analysis tools
 - Raincode Insight
 - [DB Browser for SQLite](#)
 - [SQLite ODBC](#)
 - [HxD](#) hex editor
 - [ILSpy](#) .NET decompiler
 - [Meld](#)
 - [SQL Server Management Studio](#)

Raincode 360 Licensing

- Raincode 360 is licensed under the Raincode 360 Software License Agreement.
 - The latest version of the license is available at: <https://www.raincode.com/legal/TLA.pdf>
 - A local copy is available on the VM at: C:\Raincode360\Cache\TLA.pdf
- Raincode 360 includes third-party products which are covered by their own respective licenses.
 - Third-party licenses are available on the VM at: C:\Raincode360\Cache\licenses.html

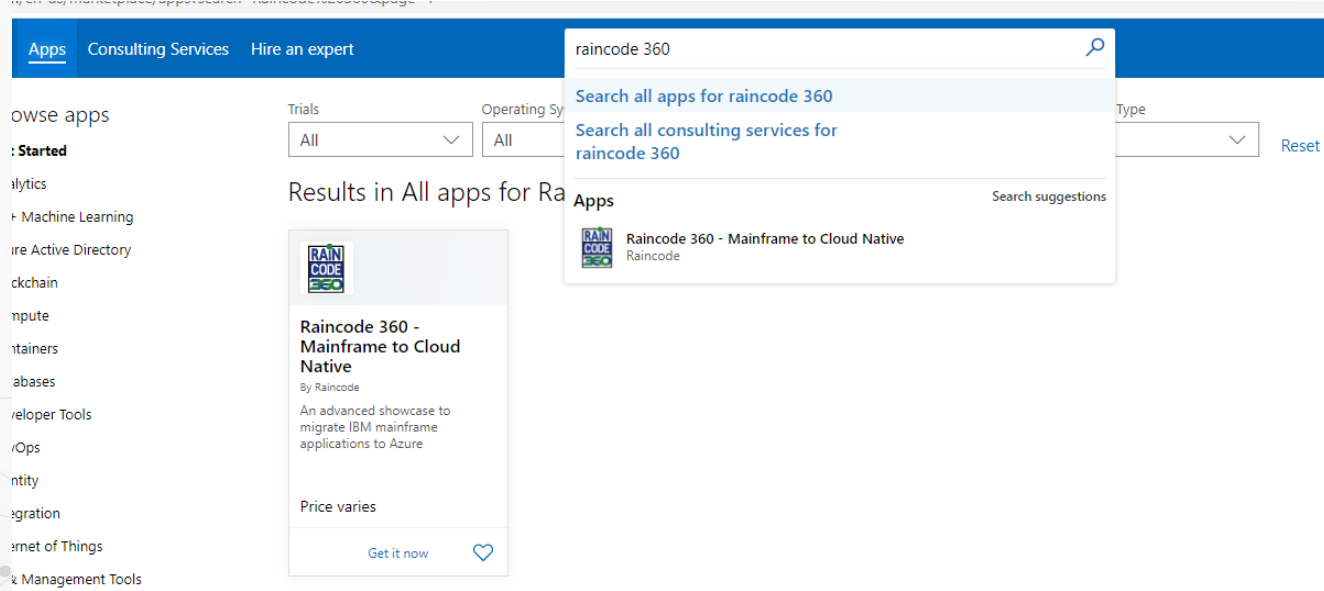
Raincode 360 on Azure Marketplace



Steps to download Raincode 360 from Marketplace



- Step 1: Login to the Azure Marketplace (<https://azuremarketplace.microsoft.com/>)
- Step 2: Search for **Raincode 360**

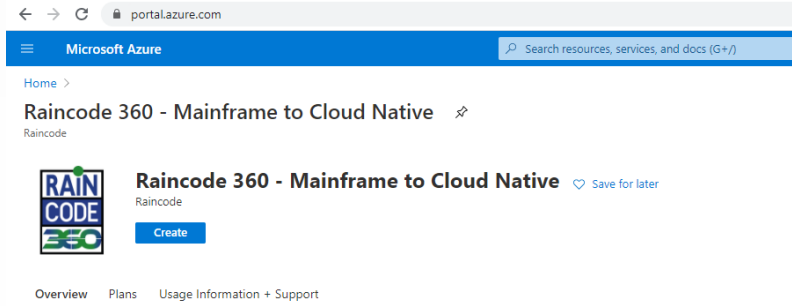


- Step 3: Click **Get it now** and fill in the required details. It will redirect you to the Azure Portal.

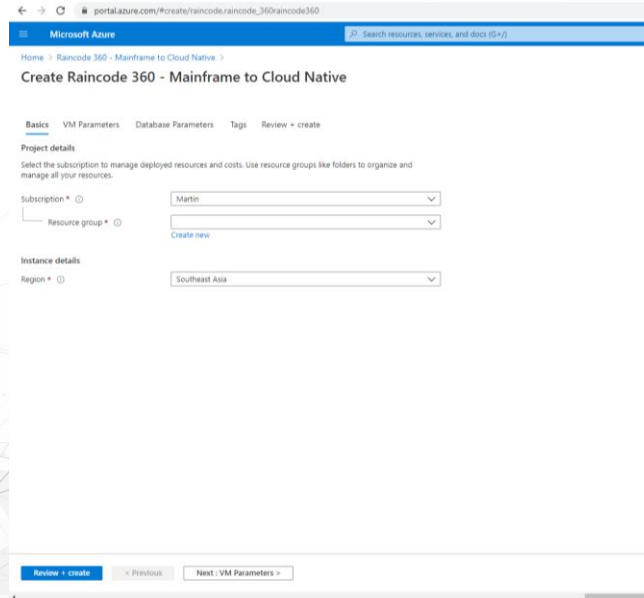
Steps to create Raincode 360 from Marketplace



- Step 4: Click Create



- Step 5: Enter Resource group name and choose Region



Steps to create Raincode 360 from Marketplace



- Step 6: Enter the details:
 - Size
 - Username and password
 - VM access type:
 - RDP through public IP: the VM is made available on a public IP address with open access only for the RDP protocol (TCP port 3389).
 - Bastion: the VM is secured using [Azure Bastion](#). The VM is not exposed on the public Internet and can only be accessed through the Azure portal. Allows fine-grained access control using Azure role assignments.

Note: Azure Bastion greatly improves security, but it also incurs supplementary resource costs. See [Azure Bastion pricing](#) for details.

Microsoft Azure

Home > Raincode 360 - Mainframe to Cloud Native >

Create Raincode 360 - Mainframe to Cloud Native

Basics VM Parameters Database Parameters Tags Review + create

i Raincode 360 provides a virtual machine which houses the Raincode Development Environment and our product showcases.

Customize your VM

Size * ⓘ

1x Standard B2ms
2 vcpus, 8 GB memory
[Change size](#)

Admin Username * ⓘ shipra ✓

Password * ⓘ ✓

Confirm password * ✓

VM access type * ⓘ

RDP through public IP ^
RDP through public IP
Azure Bastion

[Review + create](#) < Previous Next : Database Parameters >

Steps to create Raincode 360 from Marketplace



- Step 7: Enter details:
 - SQL server admin username and password
- Step 8: Enter tags (optional):
 - Name
 - Value
 - Resource types

portal.azure.com/#create/raincode.raincode_360raincode360

Microsoft Azure

Search resources, services, and docs (G+/I)

Home > Raincode 360 - Mainframe to Cloud Native >

Create Raincode 360 - Mainframe to Cloud Native

Basics VM Parameters **Database Parameters** Tags Review + create

i Raincode 360 provides an Azure SQL Server instance which houses the databases shared by our various showcases.

SQL Server Admin Username * ⓘ

Password * ⓘ

Confirm password * ⓘ

SQL Server and Databases location * ⓘ

Review + create < Previous Next : Tags >

portal.azure.com/#create/raincode.raincode_360raincode360

Microsoft Azure

Search resources, services, and docs (G+/I)

Home > Raincode 360 - Mainframe to Cloud Native >

Create Raincode 360 - Mainframe to Cloud Native

Basics VM Parameters Database Parameters **Tags** Review + create

Tags are name/value pairs that enable you to categorize resources and view consolidated billing by applying the same tag to multiple resources and resource groups. [Learn more about tags](#)

Note that if you create tags and then change resource settings on other tabs, your tags will be automatically updated.

Name ⓘ Value ⓘ Resource

: S selected

Review + create < Previous Next : Review + create >

Steps to create Raincode 360 from Marketplace



- Step 9: Click Review + Create
- Step 10: Once the validation is passed, click Create

After the successful creation, VM will appear under Virtual Machines (Azure Services)

portal.azure.com/#create/raincode.raincode_360raincode360

Microsoft Azure Search resources, services, and docs (G+)

Home > Raincode 360 - Mainframe to Cloud Native >

Create Raincode 360 - Mainframe to Cloud Native

✓ Validation Passed

transactional information with the provider(s) of the offering(s) for support, billing and other transactional activities. Microsoft does not provide rights for third-party offerings. See the [Azure Marketplace Terms](#) for additional details.

Basics

Subscription	Martin
Resource group	demo1
Region	East US

VM Parameters

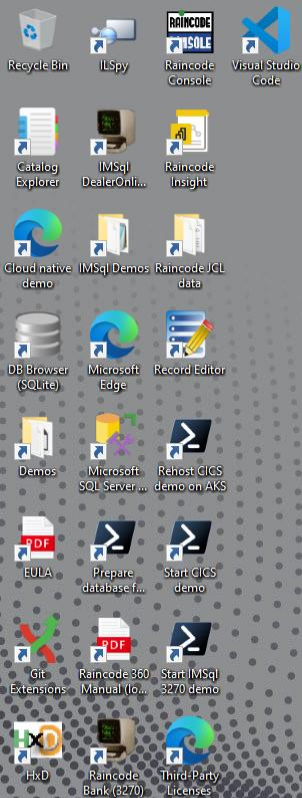
Size	Standard_B2ms
Admin Username	shipra
Password	*****
VM access type	RDP through public IP

Database Parameters

SQL Server Admin Username	shipra
Password	*****
SQL Server and Databases location	East US

Create < Previous Next Download a template for automation

Welcome to Raincode 360!



B
CROSS
W



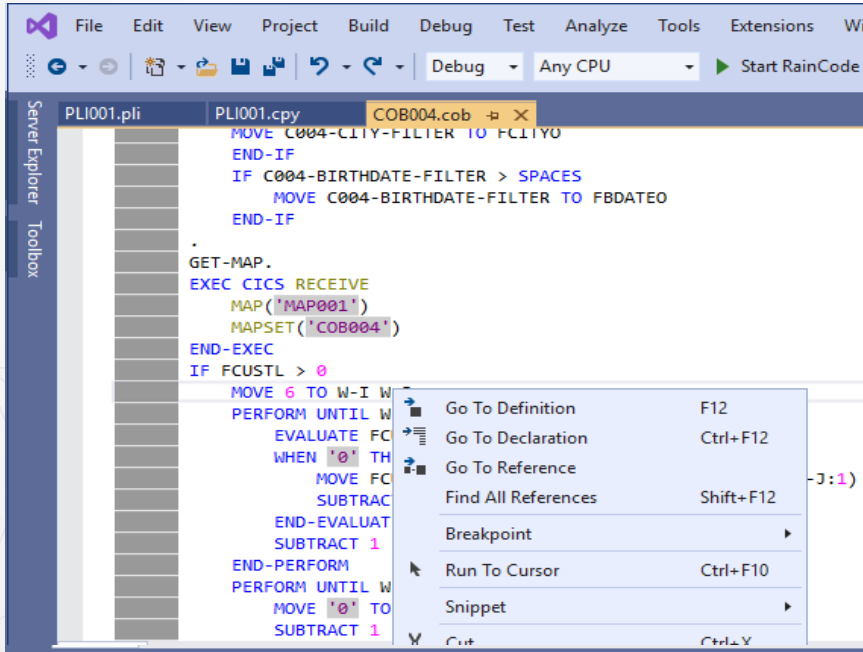


- Demos:
 - [Visual Studio Integration](#)
 - [Repository Database](#)
 - [Raincode Insight](#)
 - [Raincode Insight – Call Graph Tools](#)

Visual Studio Integration



- Raincode products are fully integrated with the Microsoft Visual Studio development environment and provide developer productivity tools, such as:



Some more features to name:

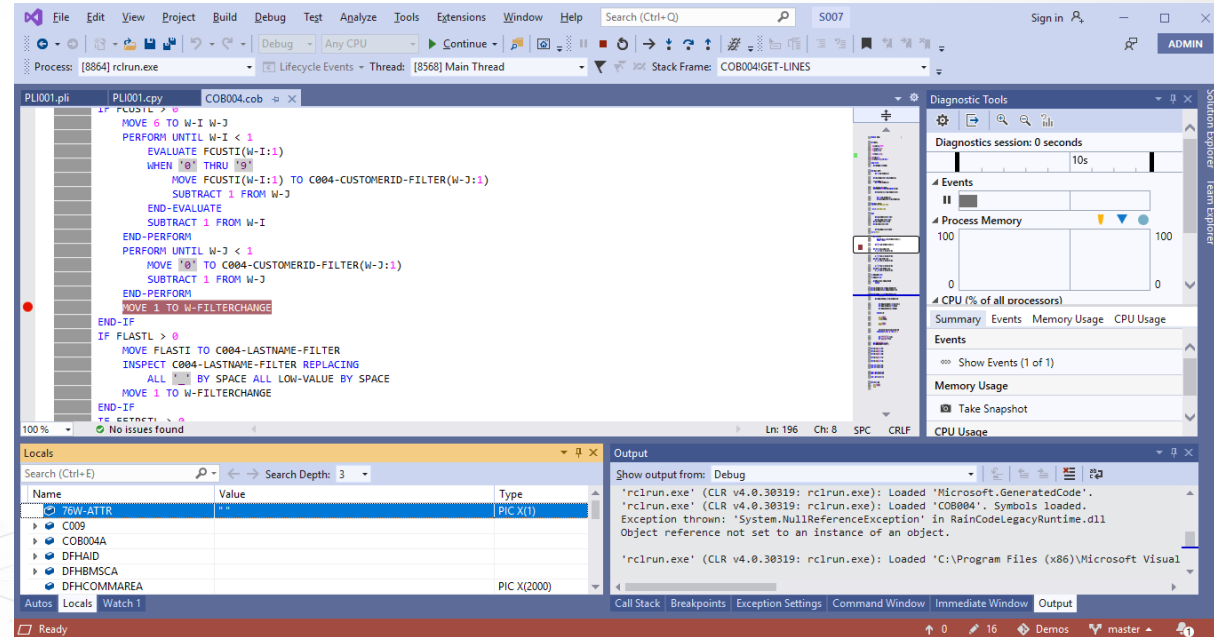
- Visuals:
 - Color coding
- Assistance in editing:
 - Code folding
 - IntelliSense
 - Automatic completion (variables, labels)
- Navigation
 - Declaration (jump to definition)

Hovering over SQL statements shows the converted SQL statement sent to the database.

Visual Studio Integration – Symbolic Debugging



- Raincode compilers are fully integrated with the development environment debugger.
- It allows users to:
 - set breakpoints in the source code
 - perform step-by-step debugging



Repository Database



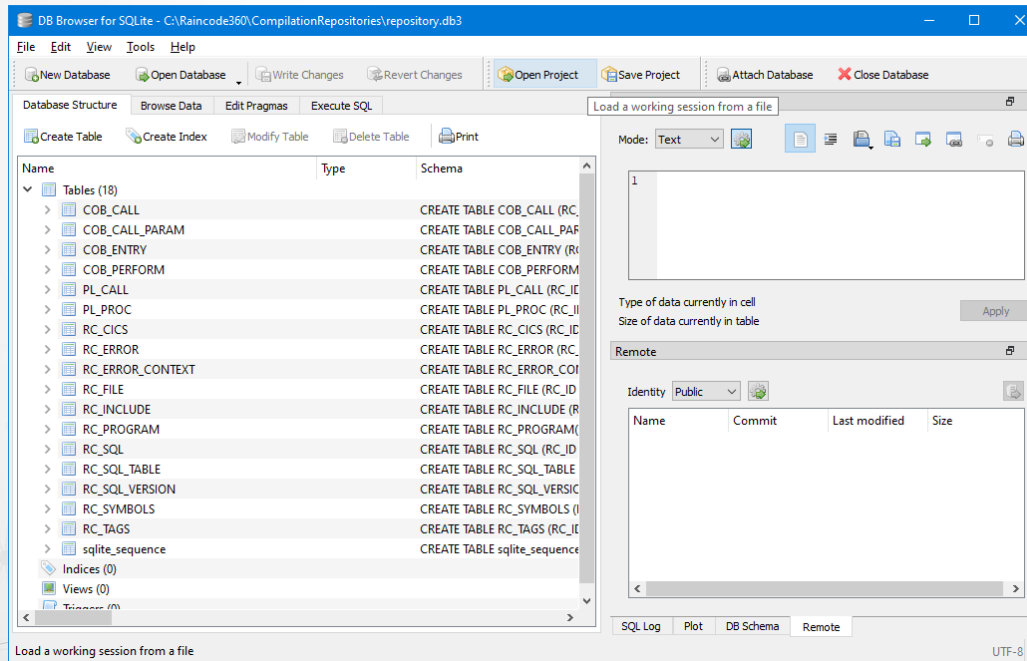


- The Raincode Legacy compilers can populate a repository database when compiling
- This repository describes various artifacts found in the source code:
 - Programs, including size statistics, compilation time, compiler version, and compiler return code.
 - Procedures
 - CICS statements
 - SQL statements
 - Compilation errors
 - Call graph, with discrimination between local and cross-program calls
 - Etc.
- Essential asset for any non-trivial migration projects

Repository Database – How to demonstrate



- Step 1: Open the solution from folder Qix Emulator – Bank Application
- Step 2: Select Build -> Rebuild Solution
- Step 3: Open **repository.db3** to browse the repository, placed at C:\Raincode360\CompilationRepositories



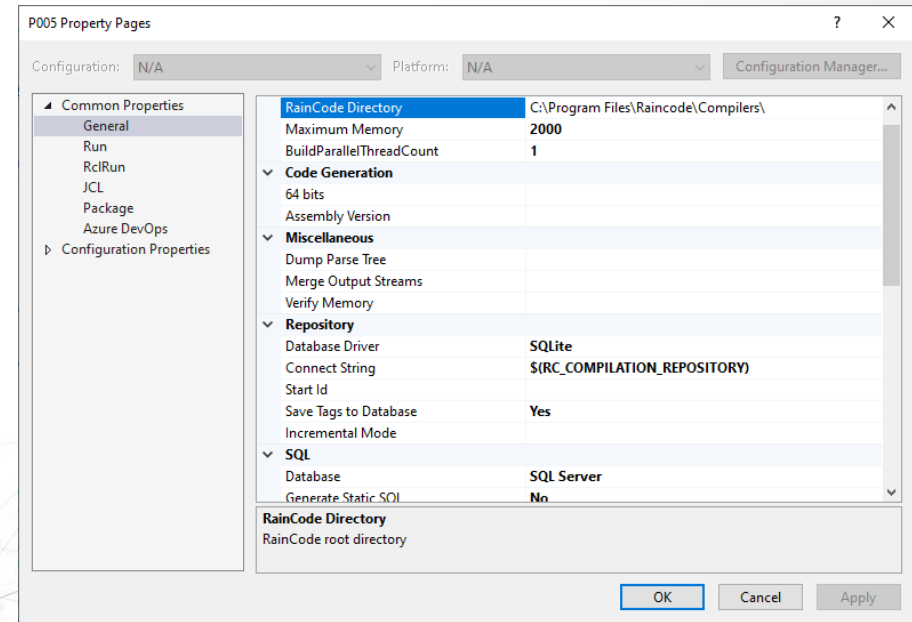
We are using DB Browser for browsing the SQLite database. This program is also provided with Raincode 360.

<https://sqlitebrowser.org/dl/>

Repository Database – How to demonstrate



- The repository database made by compiling all the demos within Raincode 360, located at C:\Raincode360\CompilationRepositories\repository.db3 is already prepared and can be used directly
- If you want to configure the compilation repository, right-click on P005 in the Solution Explorer and select Properties to access the Compilation Repository configuration.
- Under Repository Configuration, you can:
 - Configure Database Driver: SQLite/ODBC. SQLite is easier (SQL Server requires an external database to be set up)
 - Set Connect String

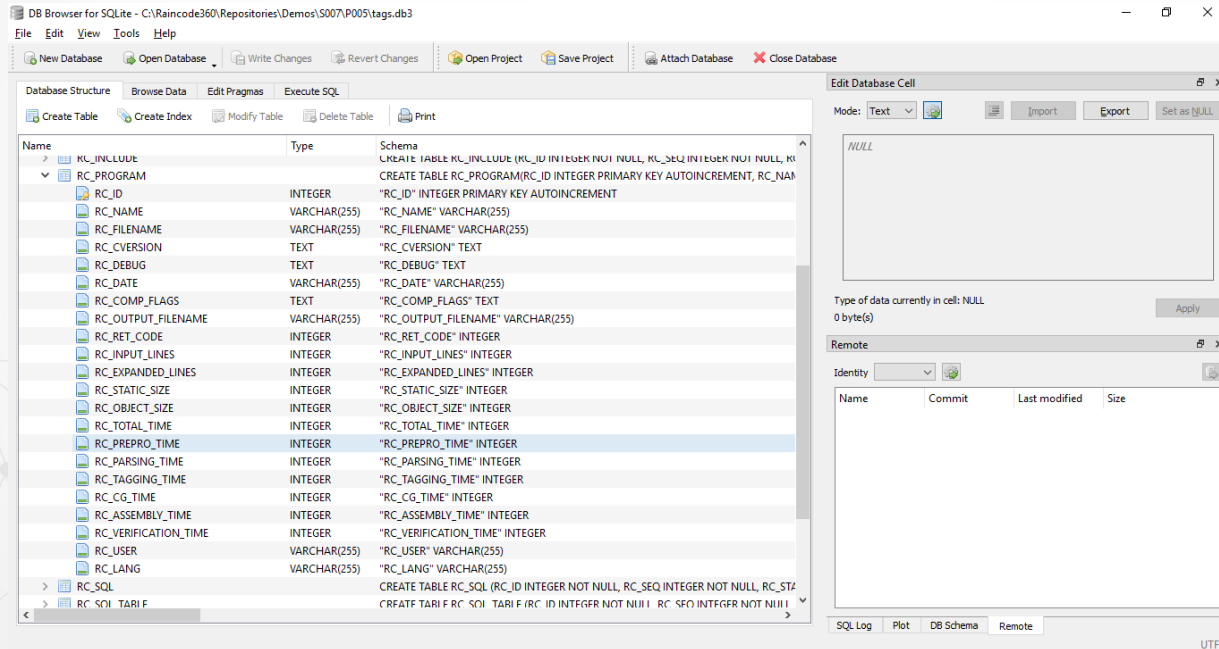


There are several tables created in the compilation repository. For details on the compiler artifacts repository, see [Artifacts Repository](#)

Browse Table



- A few tables explained:
 - **RC_PROGRAM:** The RC_PROGRAM table holds information about program files analyzed by the Raincode Legacy Compiler



It includes the program name, the compiler version used to build it; compiled in debug mode, compilation date, compilation options used, compilation return code, etc.

Browse Table



- In the **RC_PROGRAM** table, each row represents a COBOL, or PL/I program in the solution:

DB Browser for SQLite - C:\Raincode360\Repositories\Demos\S007\P005\tags.db3

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

Table: RC_PROGRAM

	RC_ID	RC_NAME	RC_FILENAME	RC_CVERSION	RC_DEBUG	RC_DATE	RC_COMP_FLAGS	OUTPUT_FILEN	RC_RE
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	COB004	COB004.COB	4.0.261.0, Bui...	Y	2020:06:02 1...	:Dependency...	bin\Debug\CO...	0
2	2	COB004A	COB004A.COB	4.0.261.0, Bui...	Y	2020:06:02 1...	:Dependency...	bin\Debug\CO...	0
3	3	COB005	COB005.COB	4.0.261.0, Bui...	Y	2020:06:02 1...	:Dependency...	bin\Debug\CO...	0
4	4	COB005A	COB005A.COB	4.0.261.0, Bui...	Y	2020:06:02 1...	:Dependency...	bin\Debug\CO...	0
5	5	COB006	COB006.COB	4.0.261.0, Bui...	Y	2020:06:02 1...	:Dependency...	bin\Debug\CO...	0
6	6	COB007	COB007.COB	4.0.261.0, Bui...	Y	2020:06:02 1...	:Dependency...	bin\Debug\CO...	0
7	7	COB008	COB008.COB	4.0.261.0, Bui...	Y	2020:06:02 1...	:Dependency...	bin\Debug\CO...	0
8	8	COB009	COB009.COB	4.0.261.0, Bui...	Y	2020:06:02 1...	:Dependency...	bin\Debug\CO...	0
9	9	COB010	COB010.COB	4.0.261.0, Bui...	Y	2020:06:02 1...	:Dependency...	bin\Debug\CO...	0
10	10	COB011	COB011.COB	4.0.261.0, Bui...	Y	2020:06:02 1...	:Dependency...	bin\Debug\CO...	0
11	11	PLI001	PLI001.PLI	4.0.261.0, Bui...	Y	2020:06:02 1...	:Dependency...	bin\Debug\PLI...	0

1 - 11 of 11

Go to: 1

Edit Database Cell

Mode: Text

Import Export Set as NULL

1

Type of data currently in cell: Text / Numeric
1 char(s)

Apply

Remote

Identity

Name	Commit	Last modified	Size
------	--------	---------------	------

SQL Log Plot DB Schema Remote

UTF-8

Browse Table

- RC_RET_CODE=0 indicates that compilation was successful

DB Browser for SQLite - C:\Raincode360\Repositories\Demos\S007\P005\tags.db3

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

Table: RC_PROGRAM

	_DATE	RC_COMP_FLAGS	RC_OUTPUT_FILENAME	RC_RET_CODE	RC_INPUT_LINES	EXPANDED_LIN	RC_STATIC_SIZE	RC_O
		Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	:02 11:46:53	:Dependency...	bin\Debug\COB004.dll	0	310	848	5600	2949
2	:02 11:46:53	:Dependency...	bin\Debug\COB004A.dll	0	196	269	2272	9420
3	:02 11:46:54	:Dependency...	bin\Debug\COB005.dll	0	349	765	3104	2467
4	:02 11:46:54	:Dependency...	bin\Debug\COB005A.dll	0	103	168	544	6656
5	:02 11:46:55	:Dependency...	bin\Debug\COB006.dll	0	401	903	3712	2892
6	:02 11:46:56	:Dependency...	bin\Debug\COB007.dll	0	206	561	3520	2063
7	:02 11:46:57	:Dependency...	bin\Debug\COB008.dll	0	365	684	2808	2181
8	:02 11:46:58	:Dependency...	bin\Debug\COB009.dll	0	66	118	336	5939
9	:02 11:46:59	:Dependency...	bin\Debug\COB010.dll	0	472	995	3864	3066
10	:02 11:46:59	:Dependency...	bin\Debug\COB011.dll	0	42	174	1840	1054
11	:02 11:47:00	:Dependency...	bin\Debug\PLI001.dll	0	40	67	32	4966

1 - 11 of 11

Go to: 1

Edit Database Cell

Mode: Text

Import Export Set as NULL

1

Type of data currently in cell: Text / Numeric
1 char(s)

Apply

Remote

Identity

Name	Commit	Last modified	Size
------	--------	---------------	------

SQL Log Plot DB Schema Remote

UTF-8

RC_RET_CODE gives the Raincode Legacy Compiler's return code

Browse Table

- **COB_CALL:** keeps track of all the CALL and EXEC CICS CALL and XCTL statements found when compiling COBOL source programs. CALL statements can be static, as in CALL "MYPRG", or dynamic, referring to a variable that holds the program's name to call, as in CALL PRG-NAME (COBOL only).

The screenshot displays the DB Browser for SQLite application. The main window shows the 'Database Structure' tab with the 'COB_CALL' table selected. The table has the following columns: RC_ID, RC_SEQ, COB_FILE, COB_START_LIN, COB_LOCAL, COB_ARITY, COB_TO, and COB_VERB. The data is displayed in a table with 18 rows. The 'Edit Database Cell' dialog is open on the right, showing the first row of data.

	RC_ID	RC_SEQ	COB_FILE	COB_START_LIN	COB_LOCAL	COB_ARITY	COB_TO	COB_VERB
1	1	1076	C:\Raincode360\Repositories\Demos\S007\P005\tags.db3	70	0	1	COB005	CICS XCTL
2	1	1192	C:\Raincode360\Repositories\Demos\S007\P005\tags.db3	196	0	1	COB004A	CICS LINK
3	3	809	C:\Raincode360\Repositories\Demos\S007\P005\tags.db3	46	0	1	COB004	CICS XCTL
4	3	814	C:\Raincode360\Repositories\Demos\S007\P005\tags.db3	50	0	1	COB006	CICS XCTL
5	3	823	C:\Raincode360\Repositories\Demos\S007\P005\tags.db3	58	0	1	COB006	CICS XCTL
6	3	830	C:\Raincode360\Repositories\Demos\S007\P005\tags.db3	65	0	1	COB007	CICS XCTL
7	3	837	C:\Raincode360\Repositories\Demos\S007\P005\tags.db3	72	0	1	COB007	CICS XCTL
8	3	844	C:\Raincode360\Repositories\Demos\S007\P005\tags.db3	79	0	1	COB007	CICS XCTL
9	3	871	C:\Raincode360\Repositories\Demos\S007\P005\tags.db3	107	0	1	COB005A	CICS LINK
10	5	990	C:\Raincode360\Repositories\Demos\S007\P005\tags.db3	66	0	1	COB005	CICS XCTL
11	5	1002	C:\Raincode360\Repositories\Demos\S007\P005\tags.db3	80	0	1	COB005	CICS XCTL
12	6	725	C:\Raincode360\Repositories\Demos\S007\P005\tags.db3	72	0	1	COB008	CICS XCTL
13	6	731	C:\Raincode360\Repositories\Demos\S007\P005\tags.db3	80	0	1	COB005	CICS XCTL
14	6	738	C:\Raincode360\Repositories\Demos\S007\P005\tags.db3	85	0	1	COB008	CICS XCTL
15	7	670	C:\Raincode360\Repositories\Demos\S007\P005\tags.db3	56	0	1	COB007	CICS XCTL
16	7	679	C:\Raincode360\Repositories\Demos\S007\P005\tags.db3	64	0	1	COB010	CICS XCTL
17	7	715	C:\Raincode360\Repositories\Demos\S007\P005\tags.db3	93	0	1	COB007	CICS XCTL
18	7	928	C:\Raincode360\Repositories\Demos\S007\P005\tags.db3	348	0	1	PI 1001	CICS LINK

Browse Table



- **RC_CICS:** lists all the CICS statements encountered in the portfolio

DB Browser for SQLite - C:\Raincode360\Repositories\Demos\S007\P005\tags.db3

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragmas Execute SQL

Table: RC_CICS

	RC_ID	RC_SEQ	RC_LINE	RC_COMMAND	RC_USED
	Filter	Filter	Filter	Filter	Filter
1	1	1075	608	XCTL PROGR...	Y
2	1	1088	622	SEND CONTR...	Y
3	1	1089	624	RETURN	Y
4	1	1090	627	SEND CONTR...	Y
5	1	1092	630	RETURN TRA...	Y
6	1	1113	656	RECEIVE MAP...	Y
7	1	1191	734	LINK PROGRA...	Y
8	1	1241	759	SEND MAP M...	Y
9	1	1243	766	SEND MAP M...	Y
10	1	1359	842	SEND MAP M...	Y
11	2	291	216	RETURN	Y
12	3	808	462	XCTL PROGR...	Y
13	3	813	466	XCTL PROGR...	Y
14	3	822	474	XCTL PROGR...	Y
15	3	829	481	XCTL PROGR...	Y
16	3	836	488	XCTL PROGR...	Y
17	3	843	495	XCTL PROGR...	Y
18	3	866	515	RETURN TRA...	Y

1 - 18 of 69

Go to: 1

Edit Database Cell

Mode: Text

Import Export Set as NULL

1

Type of data currently in cell: Text / Numeric
1 char(s)

Apply

Remote

Identity

Name	Commit	Last modified	Size
------	--------	---------------	------

SQL Log Plot Remote

UTF-8

Browse Table

- **RC_ERROR**: summarizes the errors encountered while compiling the program

The screenshot displays the DB Browser for SQLite interface. The main window shows the 'Database Structure' tab with a list of tables. The 'RC_ERROR' table is selected, and its structure is shown in the 'Table' tab. The table has the following columns:

Name	Type	Schema
RC_ID	INTEGER	"RC_ID" INTEGER NOT NULL
RC_SEQ	INTEGER	"RC_SEQ" INTEGER NOT NULL
RC_LINE	INTEGER	"RC_LINE" INTEGER
RC_COLUMN	INTEGER	"RC_COLUMN" INTEGER
RC_MODULE	TEXT	"RC_MODULE" TEXT
RC_MESSAGE	TEXT	"RC_MESSAGE" TEXT

The 'Edit Database Cell' dialog is open, showing the 'Text' mode. The 'Type of data currently in cell: Text / Numeric' is set to 'Text'. The 'Remote' tab is selected, and the 'Identity' is set to 'RC_MODULE' TEXT. The 'Name' column is highlighted, and the 'Commit' button is visible.

Browse Table



- **RC_SQL_VERSION:** This table holds the information about the automatic SQL translation process. If a translation process is applied, an entry holds the **original** statement (DB2), whereas another entry holds the **translated** statement (SQL Server).

DB Browser for SQLite - C:\Raincode360\Repositories\Demos\S007\P005\tags.db3

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragma Execute SQL

Table: RC_SQL_VERSION

ID	C_SEI	TATEI	RC_DBNAME	RC_VERSION
			Filter	Filter
1	313	312	Original	DECLARE CUR-DOWN CURSOR FOR SELECT CustomerId, FirstName, LastName, C.ZipCode, BirthDate, Z.City
2	314	312	SQLServer	DECLARE "C08004A_CUR-DOWN" CURSOR FOR SELECT "CustomerId", "FirstName", "LastName", C."ZipCode
3	318	317	Original	DECLARE CUR-UP CURSOR FOR SELECT CustomerId, FirstName, LastName, C.ZipCode, BirthDate, Z.City FRC
4	319	317	SQLServer	DECLARE "C08004A_CUR-UP" CURSOR FOR SELECT "CustomerId", "FirstName", "LastName", C."ZipCode", C
5	1092	1091	Original	SELECT CAST(City AS CHAR(60)), Country INTO :C005-CITY, :C005-COUNTRY FROM dbo.zipcode WHERE zi
6	1093	1091	SQLServer	BEGIN SELECT @V0=CONVERT(CHAR(60), "City"), @V1="Country" FROM dbo.zipcode WHERE("zipcode"=@V
7	1096	1095	Original	UPDATE dbo.Customer SET LastName = :C005-LASTNAME, FirstName = :C005-FIRSTNAME, ZipCode = :C0
8	1097	1095	SQLServer	BEGIN UPDATE dbo.Customer SET "LastName"=@V0, "FirstName"=@V1, "zipcode"=@V2, "BirthDate"=CONV
9	215	214	Original	DECLARE CUR-CUST CURSOR FOR SELECT FirstName, LastName, C.ZipCode, BirthDate, Z.City, A.AccountNu
10	216	214	SQLServer	DECLARE "C08005A_CUR-CUST" CURSOR FOR SELECT "FirstName", "LastName", C."ZipCode", CONVERT(CH
11	1471	1470	Original	DECLARE CUR-F8 CURSOR FOR SELECT ZipCode, City FROM dbo.ZipCode WHERE ZIPCODE > :C006-LAST A
12	1472	1470	SQLServer	DECLARE "C08006_CUR-F8" CURSOR FOR SELECT "ZipCode", "City" FROM dbo.ZipCode WHERE(("ZipCode">
13	1475	1474	Original	DECLARE CUR-F7 CURSOR FOR SELECT ZipCode, City FROM dbo.ZipCode WHERE ZIPCODE < :C006-FIRST A
14	1476	1474	SQLServer	DECLARE "C08006_CUR-F7" CURSOR FOR SELECT "ZipCode", "City" FROM dbo.ZipCode WHERE(("ZipCode"<
15	859	858	Original	DECLARE CUR-F8 CURSOR FOR SELECT PaymentDate, DebitAccountNumber, DescriptionText, Amount FROM
16	860	858	SQLServer	DECLARE "C08007_CUR-F8" CURSOR FOR SELECT "Alias\$0", "Alias\$1", "Alias\$2", "Alias\$3" FROM(SELECT C
17	867	866	Original	SELECT Amount INTO :W-AMOUNT-NP3 FROM dbo.Account WHERE Accountnumber = :C007-ACCOUNTNUM

1 - 18 of 40

Go to: 1

Edit Database Cell

Mode: Text Import Export Set as NULL

1

Type of data currently in cell: Text / Numeric
1 char(s)

Apply

Remote

Name	Commit	Last modified	Size
------	--------	---------------	------

SQL Log Plot Remote

UTF-8

Repository Database – SQL Editor



- In Execute SQL Tab:
- You can Execute SQL Statement to explore the repository
 - For example, the query **SELECT * FROM RC_PROGRAM WHERE RC_RET_CODE = 0** will yield all successful compilations.

The screenshot shows the 'DB Browser for SQLite' application. The 'Execute SQL' tab is active, displaying the query: `SELECT * FROM RC_PROGRAM WHERE RC_RET_CODE = 0`. The results are shown in a table with 6 rows and 7 columns: RC_ID, RC_NAME, RC_FILENAME, RC_VERSION, RC_DEBUG, RC_DATE, and RC_RET_CODE. The results show 6 rows of data, all with RC_RET_CODE = 0.

RC_ID	RC_NAME	RC_FILENAME	RC_VERSION	RC_DEBUG	RC_DATE	RC_RET_CODE
1	COB004	COB004.COB	4.0.261.0, Build nb_v4.0.260.0-2-gee9a8c2a9b	Y	2020:06:02 11:46:53	:DependencyMode=
2	COB004A	COB004A.COB	4.0.261.0, Build nb_v4.0.260.0-2-gee9a8c2a9b	Y	2020:06:02 11:46:53	:DependencyMode=
3	COB005	COB005.COB	4.0.261.0, Build nb_v4.0.260.0-2-gee9a8c2a9b	Y	2020:06:02 11:46:54	:DependencyMode=
4	COB005A	COB005A.COB	4.0.261.0, Build nb_v4.0.260.0-2-gee9a8c2a9b	Y	2020:06:02 11:46:54	:DependencyMode=
5	COB006	COB006.COB	4.0.261.0, Build nb_v4.0.260.0-2-gee9a8c2a9b	Y	2020:06:02 11:46:55	:DependencyMode=
6	COB007	COB007.COB	4.0.261.0, Build nb_v4.0.260.0-2-gee9a8c2a9b	Y	2020:06:02 11:46:56	:DependencyMode=

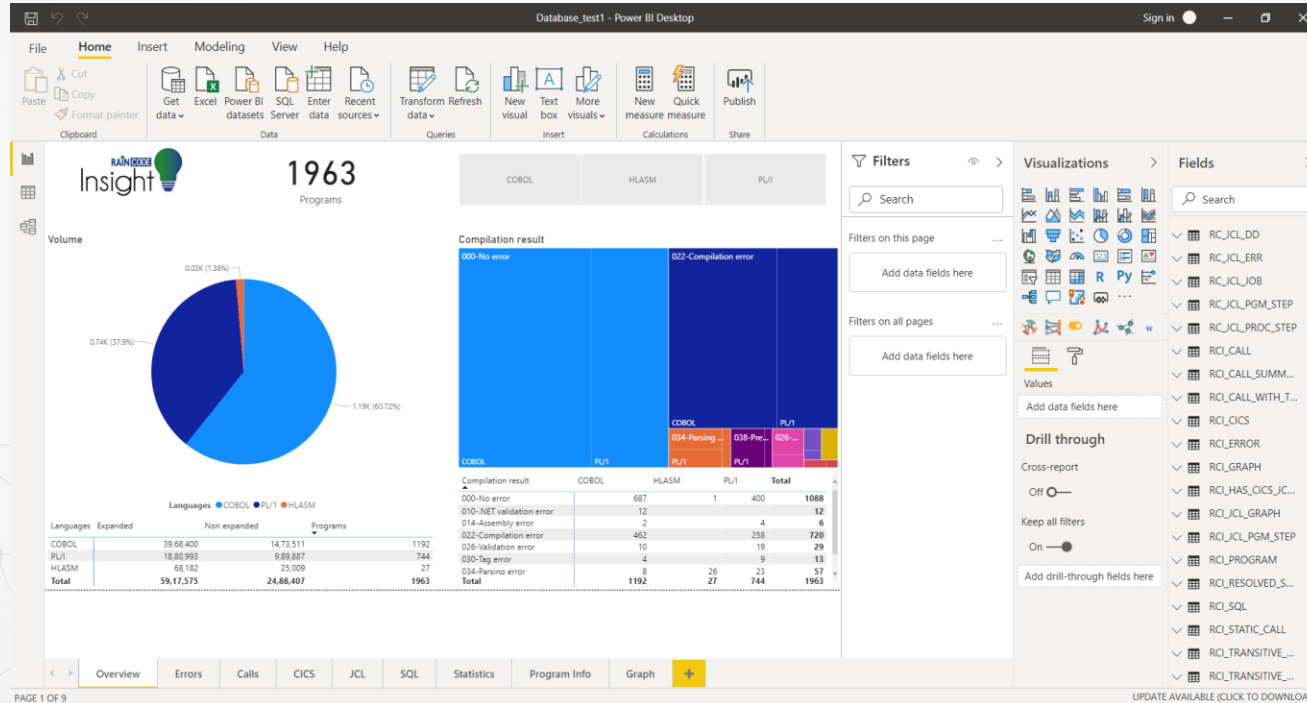
Result: 11 rows returned in 23ms
At line 1:
SELECT * FROM RC_PROGRAM WHERE RC_RET_CODE = 0

Raincode Insight

Raincode Insight – Introduction



- **Raincode Insight** is a Power BI desktop-based data visualization and analysis tool.

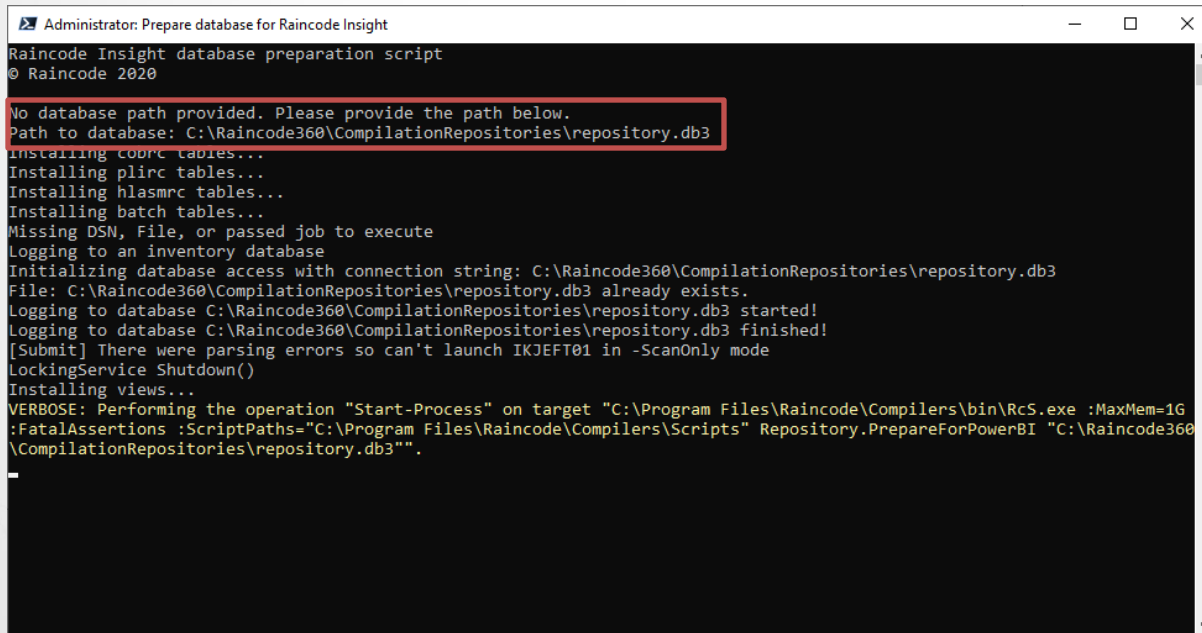


- It provides users with an interactive dashboard on top of the [Raincode Legacy Compilers repository database](#).

Raincode Insight – Install Power BI Desktop and prepare repository



- Step 1: The “Prepare Database for Raincode Insight” shortcut must be executed once to prepare a given compilation repository for Raincode Insight use. This is a one-shot operation that must be done once on a given repository database. The program will ask for the path to the repository database that should be prepared for Raincode Insight.



```
Administrator: Prepare database for Raincode Insight
Raincode Insight database preparation script
© Raincode 2020

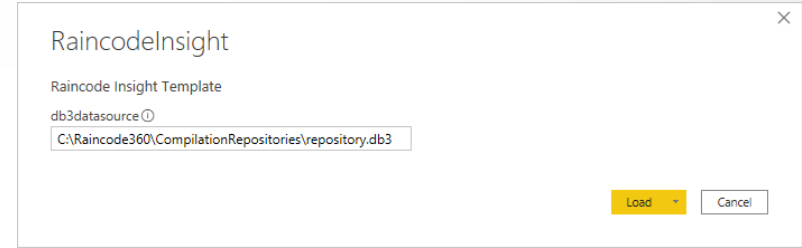
No database path provided. Please provide the path below.
Path to database: C:\Raincode360\CompilationRepositories\repository.db3
Installing coorc tables...
Installing plirc tables...
Installing hlasmrc tables...
Installing batch tables...
Missing DSN, File, or passed job to execute
Logging to an inventory database
Initializing database access with connection string: C:\Raincode360\CompilationRepositories\repository.db3
File: C:\Raincode360\CompilationRepositories\repository.db3 already exists.
Logging to database C:\Raincode360\CompilationRepositories\repository.db3 started!
Logging to database C:\Raincode360\CompilationRepositories\repository.db3 finished!
[Submit] There were parsing errors so can't launch IKJEFT01 in -ScanOnly mode
LockingService Shutdown()
Installing views...
VERBOSE: Performing the operation "Start-Process" on target "C:\Program Files\Raincode\Compilers\bin\RcS.exe :MaxMem=1G
:FatalAssertions :ScriptPaths="C:\Program Files\Raincode\Compilers\Scripts" Repository.PrepareForPowerBI "C:\Raincode360\
\CompilationRepositories\repository.db3"".
```

Note: the repository database made by compiling all the demos within Raincode 360, located at C:\Raincode360\CompilationRepositories\repository.db3 is already prepared and can be used directly in Raincode Insight.

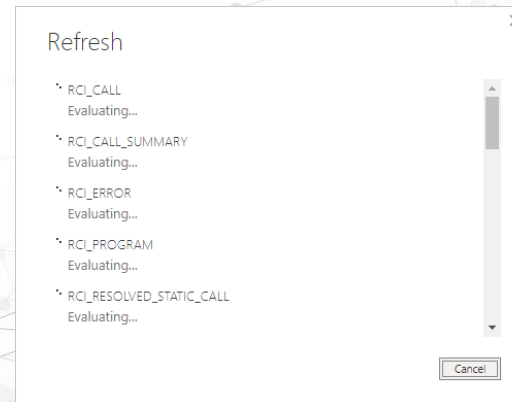
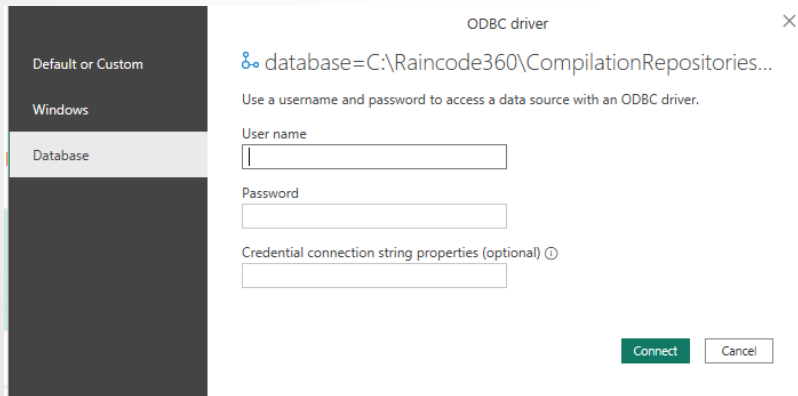
Raincode Insight – How to load database



- Step 2: Once the repository database is prepared for Raincode Insight, double click on the “Raincode Insight” shortcut placed at the desktop. Power BI will launch, and a pop-up will appear on the main window



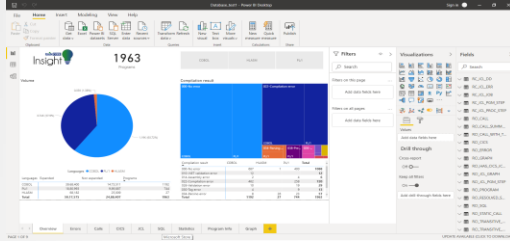
- Step 3: Provide the path to the repository database in the “db3datasource” text box and click on Load.
- Step 4: Provide a username and press Connect. A pop-up progress window with the status of database loading will appear.



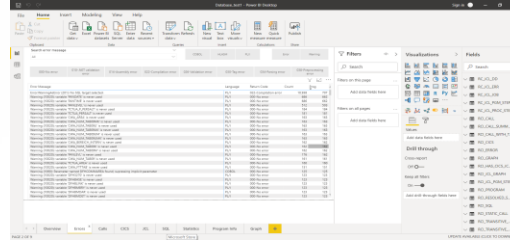
Raincode Insight – Dashboard



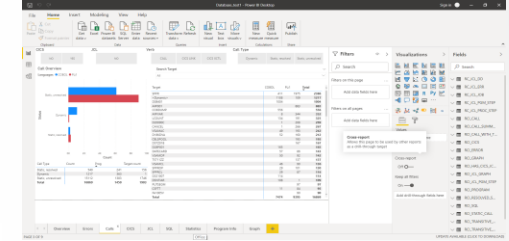
- Once the database is loaded, the Raincode Insight dashboard (9 tabs) is available to the user.



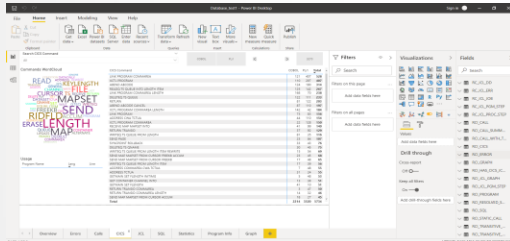
Overview



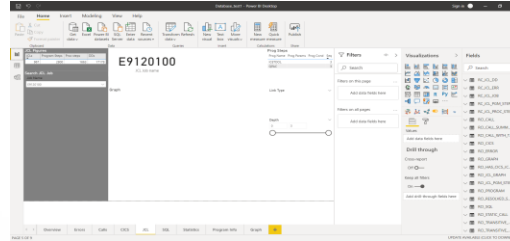
Errors



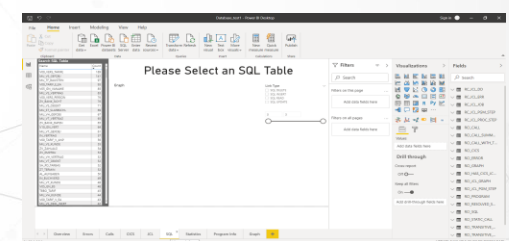
Calls



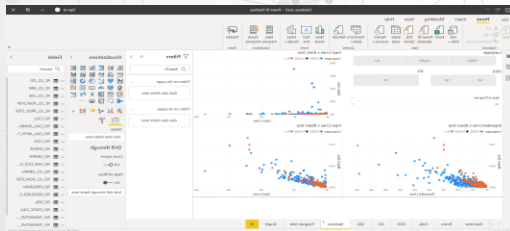
CICS



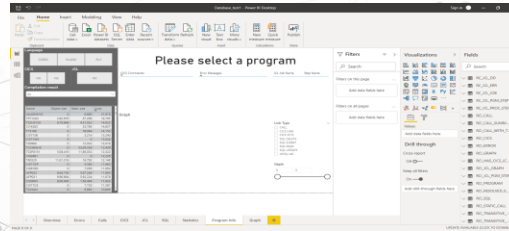
JCL



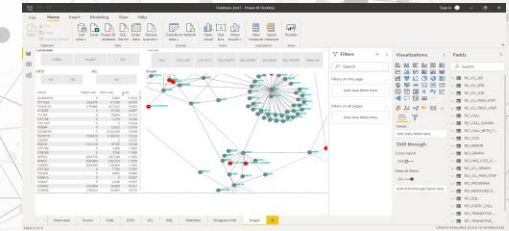
SQL



Statistics



Program Info



Graph

For details on the Raincode Insight dashboard, refer to the [Raincode Insight](#) documentation.

Raincode Insight – Call Graph Tool



Raincode Insight – Call Graph Tool



Raincode Insight -Call Graph Tool is a C# project. This tool explores call graphs that are generated from the repositories created by the Raincode compilers

Raincode Insight - Call Graph Tool Home Repositories

Welcome

Go to repositories: [Repositories](#).

Raincode Insight - Call Graph Tool Home Repositories

Repositories

[Add a new Repository](#)

Name	Path	Options
DB	C:\Nuernberger.db3	Edit Delete

You can choose to [Go to repositories](#); if a repository has been added previously, you can see a list of the repositories. In case you are using the Call Graph Tool for the first time, you have to [Add a new Repository](#)

Click on the repository name to go into details of the call graph

Raincode Insight – Call Graph Tool



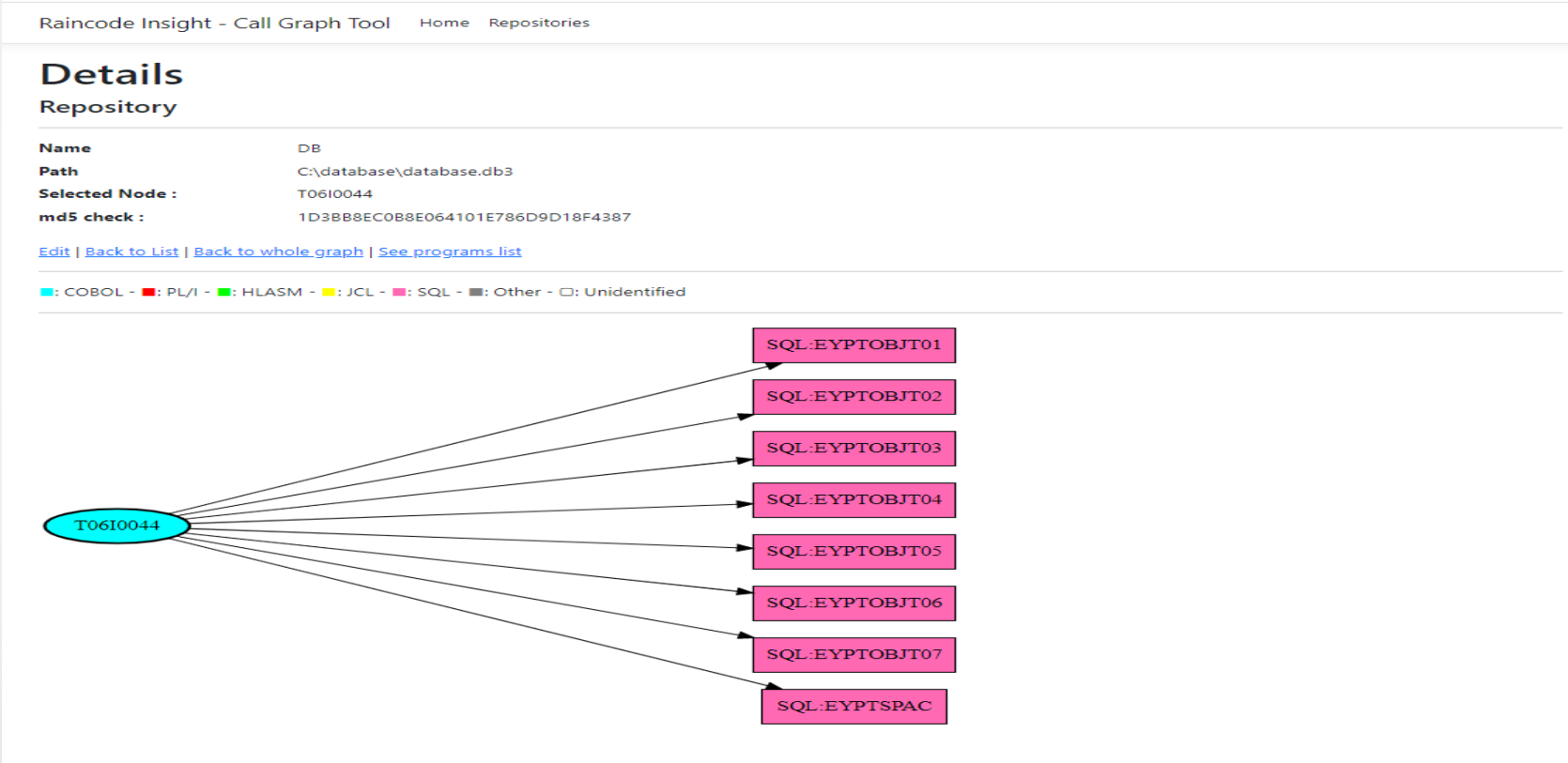
The Call Graph Tool details page shows you the calling relationships between subroutines and provides you with the options of [Edit](#), [Back to List](#), [Back to the whole graph](#), and [See program list](#). It also shows you the different color coding of the language used in the program.



Raincode Insight – Call Graph Tool



Once you click on any of the program, it shows the relationships of the subroutines of that program.



Raincode Insight – Call Graph Tool – Add a new repository



Step 1: To add a new repository, click on Add a new Repository

Step 2: Provide the name and full path of the SQLite database location and click Add repository

Raincode Insight - Call Graph Tool Home Repositories

Add new repository

In order to generate the full call graph of a repository generated by Raincode Compilers, please provide the full path of the SQLite database location.

Example:

`C:\Path\To\repository.db3`

Before you can generate the graph, the SQLite database must be prepared. To do so, please execute the script

`PrepareRaincodeInsight.ps1`

placed at

`C:\Program Files\Raincode\Compilers\scripts\Repository\`

[Back to repository list](#)

Name

Path

Add repository



- Demos:
 - [An Introduction to HLASM](#)
 - [Legacy calling Assembler](#)
 - [SQL Rewriting](#)
 - [Dump Processor](#)
 - [Custom logger](#)
 - [Legacy with database access](#)
 - [Standalone SQL Conversion](#)
 - [File Access via FileConfig](#)
 - [National Character Output on Console](#)
 - [User Defined Functions](#)
 - [Timeshift](#)
 - [RadaR](#)

An introduction to HLASM



A true compiler for ●ASM 370

Not a translator

Not a converter

Not an emulator

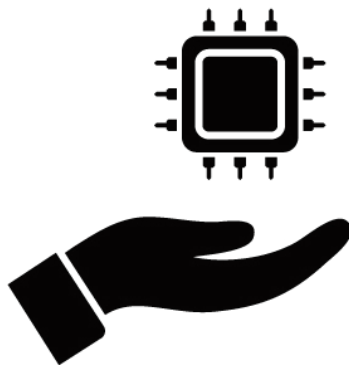
Comprehensive coverage

Self-modifying code

CICS verbs

Macro processing

Most commonly used system macros



Seamless integration

Called from ●COBOL and ●PL/I

Visual Studio (Debugging)

Single source:
execute the same ASM code on .NET and
mainframe

Optimized modes available

HLASM – Visual Studio Plugin



CallAsm (Debugging) - Microsoft Visual Studio

File Edit View Project Build Debug Team Tools Architecture Test Analyze Window Help

Debug Any CPU Continue Code Map Application Insights

INFREV01.COB WWNFB41.ASM Class1.cs CallAsm INIOS.MAC

EJECT
WWNFB41 INIOS
BALR R10,0
USING *,R10,R11,R12
LA R11,4095(R10)
LA R11,1(R11)
LM
MVI
MVI
CLC
BE TRAIT
MVI AIG1,C'1'
EJECT

* GENERATION BLOCS VSAM EXLST ET ACB *

GENCB BLK=EXLST,
SYNAD=ERPHY,
LERAD=ERLOG,
EODAD=FINAL,
AM=VSAM
LTR R15,R15
BNZ ERGENEXL
ST R1,ADEXLST
*
CLC 0(2,R6),=C'CS'
BNE STRN06

00430000
00440000
00450000
00460000
00470000
00480000
00490000
00500000
00510000
00520000
00530000
00540000
00550000
00560000
00570000
00580000
00590000
00600000
00610000
*00620000
*00630000
*00640000
*00650000
00660000
00670000
00680000
00690000
00700000
00710000
00720000

Locals

Name	Value	Type
Section Start	16960	int
Program Count	17150	int
ConditionCode	0	byte
R0	0	uint
R1	22006	uint
R2	0	uint
R3	0	uint
R4	0	uint
R5	0	uint
R6	0	uint
R7	0	uint
R8	0	uint
R9	0	uint
R10	2147500782	uint
R11	2147504878	uint
R12	2147508974	uint
R13	17032	uint
R14	21934	uint
R15	2147500780	uint

Diagnostic Tools

Diagnostics session: 1 seconds (1,881...)

1,872s

Events

Process Memory (MB)ate Bytes

174 174

CPU (% of all processors)

100 100

Summary Events Memory Usage

Events

Show Events (17 of 17)

Exceptions (0 of 0)

IntelliTrace Events (0 of 0)

Memory Usage

Take Snapshot

CPU Usage

Call Stack

Name	Lang
WWNFB41! Line 61	Rain
[External Code]	Rain
INFREV01!proc\$PROCEDURE_DIVISION Line 67	Rain
[External Code]	

Called directly from a COBOL module

In essence, treating assembler as yet another high-level language

Call Stack Breakpoints Exception Settings Immediate Window Output

Ready Ln 61 Col 1 Ch 1 INS Add to Source Control

Legacy Calling Assembler

Legacy Calling Assembler – Introduction

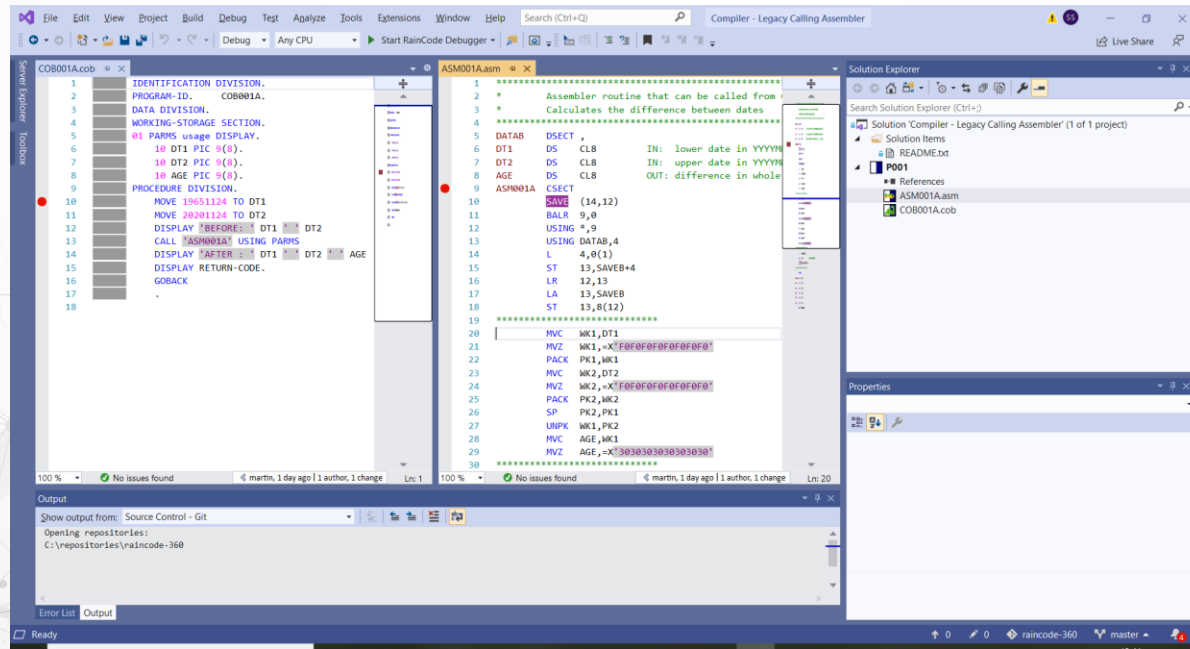


- An example of how to run a COBOL program with the rclrun (the Raincode runner) from the Visual Studio
- Demonstrates how to call a subroutine written in the assembler
- The assembler routine calculates the difference in years between two dates, that are passed as parameters by the COBOL main program

Legacy Calling Assembler – How to demonstrate



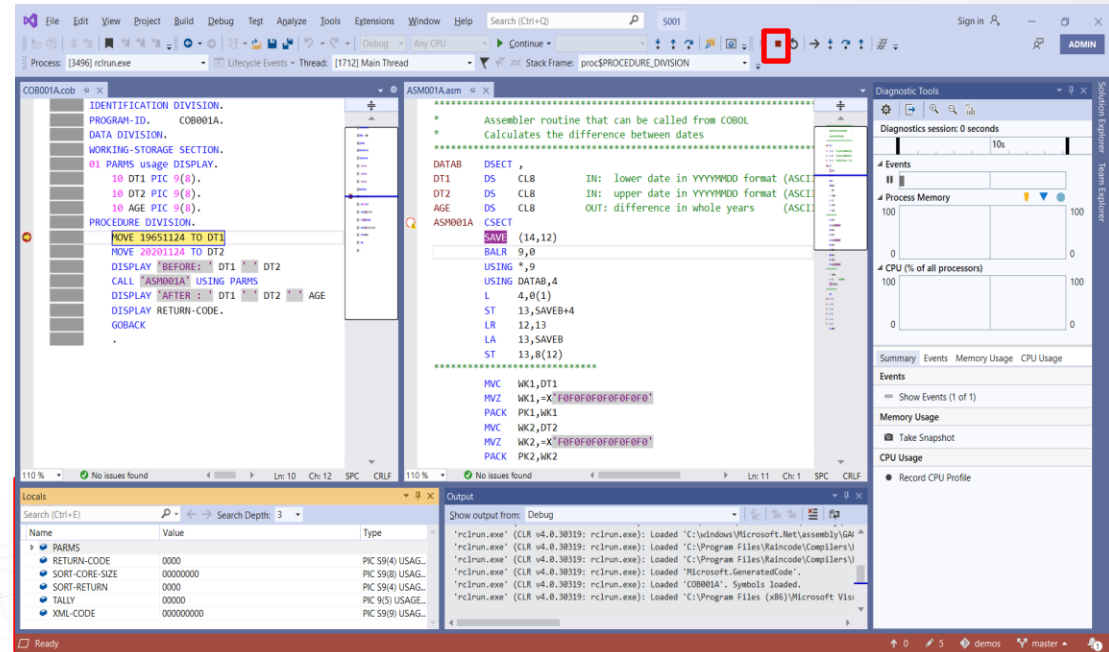
- Step 1: Open the solution from the folder Compiler - Legacy Calling Assembler
- Step 2: Put breakpoints in COB001A.cob program and ASM001A.asm



Legacy Calling Assembler – How to demonstrate



- Step 3: Click on Start Raincode Debugger
 - It activates solution in the debugging mode, and execution will stop at the breakpoint
 - Use the step buttons to step through the program
 - Check the locals windows to see a list of local values and their corresponding values
 - Stop debugging by pressing the stop icon in the debug toolbar



SQL Rewriting

Compile-time DB2 to SQL Server translation



Original DB/2:

```
EXEC SQL DECLARE C1 CURSOR FOR
  SELECT CONCAT(CONCAT(STRIIP(UPPER(LastName),
    BOTH), ', '), STRIP(FirstName, TRAILING))
  FROM People WHERE BirthDate + 30 YEARS > :Today
  WITH UR
  FETCH FIRST ROW ONLY;
```



Converted SQL Server:

```
SELECT TOP (1)
  ((dbo.STRIIP(UPPER(LastName), 'BOTH', DEFAULT)) +
    (', ')) + (dbo.STRIIP(FirstName, 'TRAILING', DEFAULT))
  FROM People WITH (READUNCOMMITTED)
  WHERE (DATEADD(YEAR, 30,
    BirthDate) > CONVERT(DATE, @V0))
```



SQL Rewriting



- SQL statements get transformed during the compilation process, but the source code remains intact.

The screenshot shows the Visual Studio IDE with a COBOL project. The main editor displays COBOL code for a program named 'W-BIRTHDATE'. A tooltip is visible over the 'EXEC SQL' statement, showing the transformed SQL query. The tooltip text is as follows:

```
DECLARE "COB004A_CUR-DOWN" CURSOR FOR SELECT
"CustomerId", "FirstName",
"LastName", C."ZipCode", CONVERT(CHAR(10),
"BirthDate") AS
"BirthDate", Z."City" FROM dbo.Customer C,
dbo.ZipCode Z
WHERE(("CustomerId">@V0) AND("CustomerId"
BETWEEN(@V1)
AND(@V2) AND "LastName" LIKE(RTRIM(@V3))
+('%') AND "FirstName"
LIKE(RTRIM(@V4))+('%') AND C."ZipCode" LIKE
(RTRIM(@V5))+('%')
AND "BirthDate" LIKE(RTRIM(@V6))+('%') AND
Z."City"
LIKE(RTRIM(@V7))+('%') AND
(Z."ZipCode"=C."ZipCode")) ORDER BY
"CustomerId"

LIKE RTRIM(:COB004A-BIRTHDATE-FILTER) || '%' AND
Z.City
LIKE RTRIM(:COB004A-CITY-FILTER) || '%' AND
Z.ZipCode = C.ZipCode
ORDER BY CustomerId
END-EXEC.
```

The Visual Studio plug-in allows you to see the transformed SQL statement by hovering the mouse on the original statement.

Dump Processor

Dump Processor – Introduction



- When a program crashes, it is possible to obtain a report with the following information:
 - The .NET exception that stopped the execution
 - A COBOL stack trace, with the file and the line number
 - the list of variables in all loaded modules and their value
- Dump Processor consists of the following demos:
 - Plugin
 - CSharp Main
 - Plugin-JSON
- Installs a dump processor based on the abstract BaseDumpProcessor class.
- The Dump Processor will produce a formatted report when a program crash.
- Refer to [Dump Processor](#) for more details.

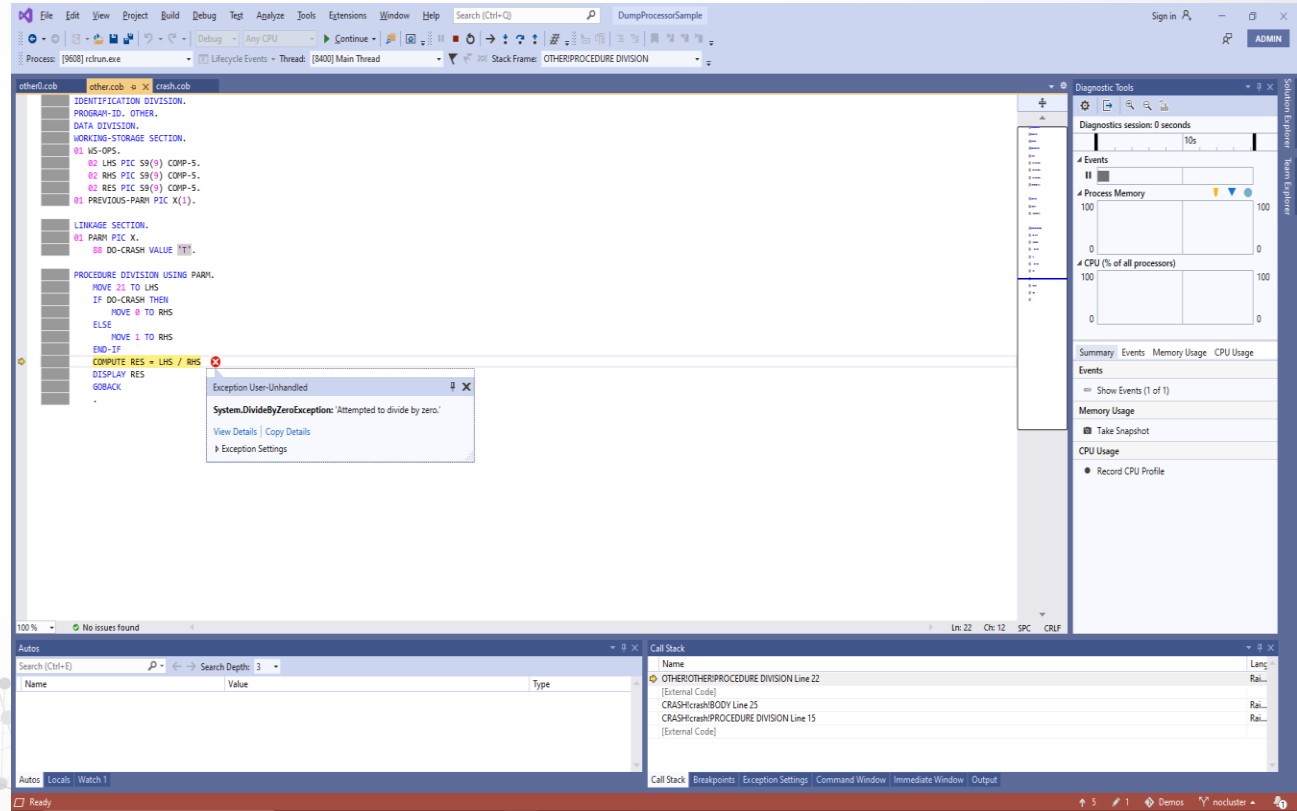
Dump Processor – Plugin: How to demonstrate



- Step 1: Open the solution from the folder Compiler - Dump Processor - Plugin
- Step 2: Start the Raincode debugger

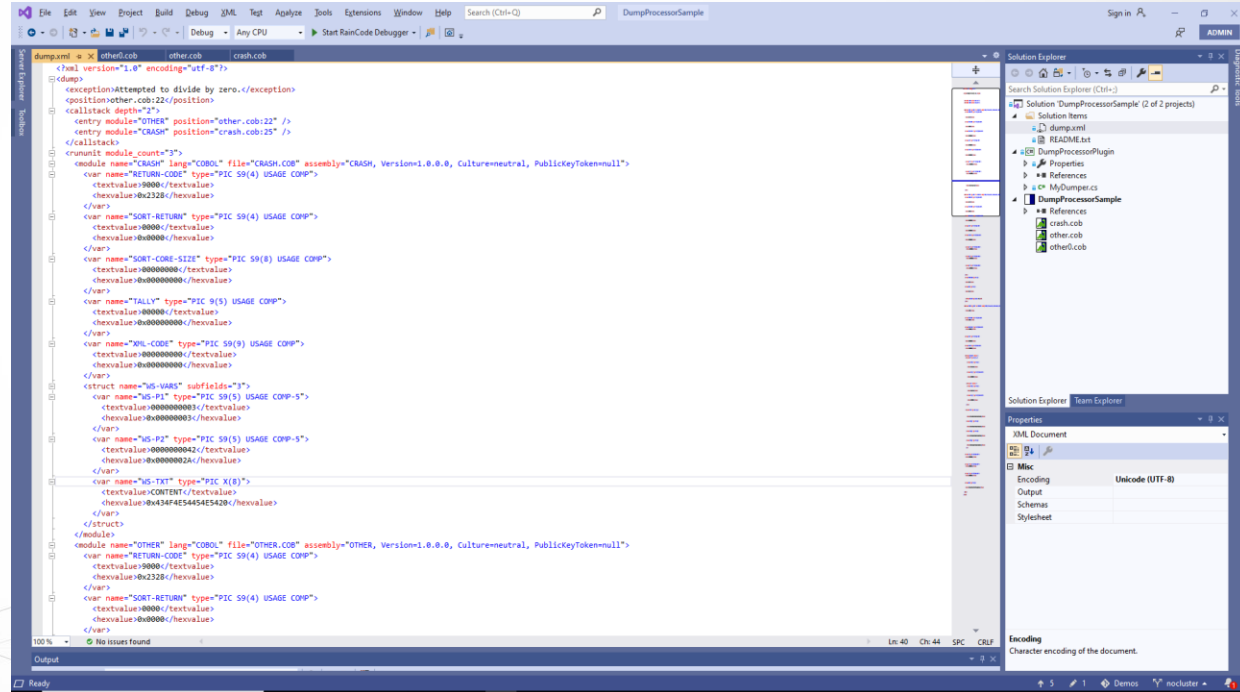
Execution will hit a breakpoint in other.cob caused by a DivideByZeroException.

- Step 3: Click continue or close the application window to exit the debugger



Dump Processor – Plugin: How to see output

- Open the dump.xml from solution items
- The XML contains the working storage of the three COBOL programs involved at the time of the crash.



Dump Processor – C# Main: How to demonstrate

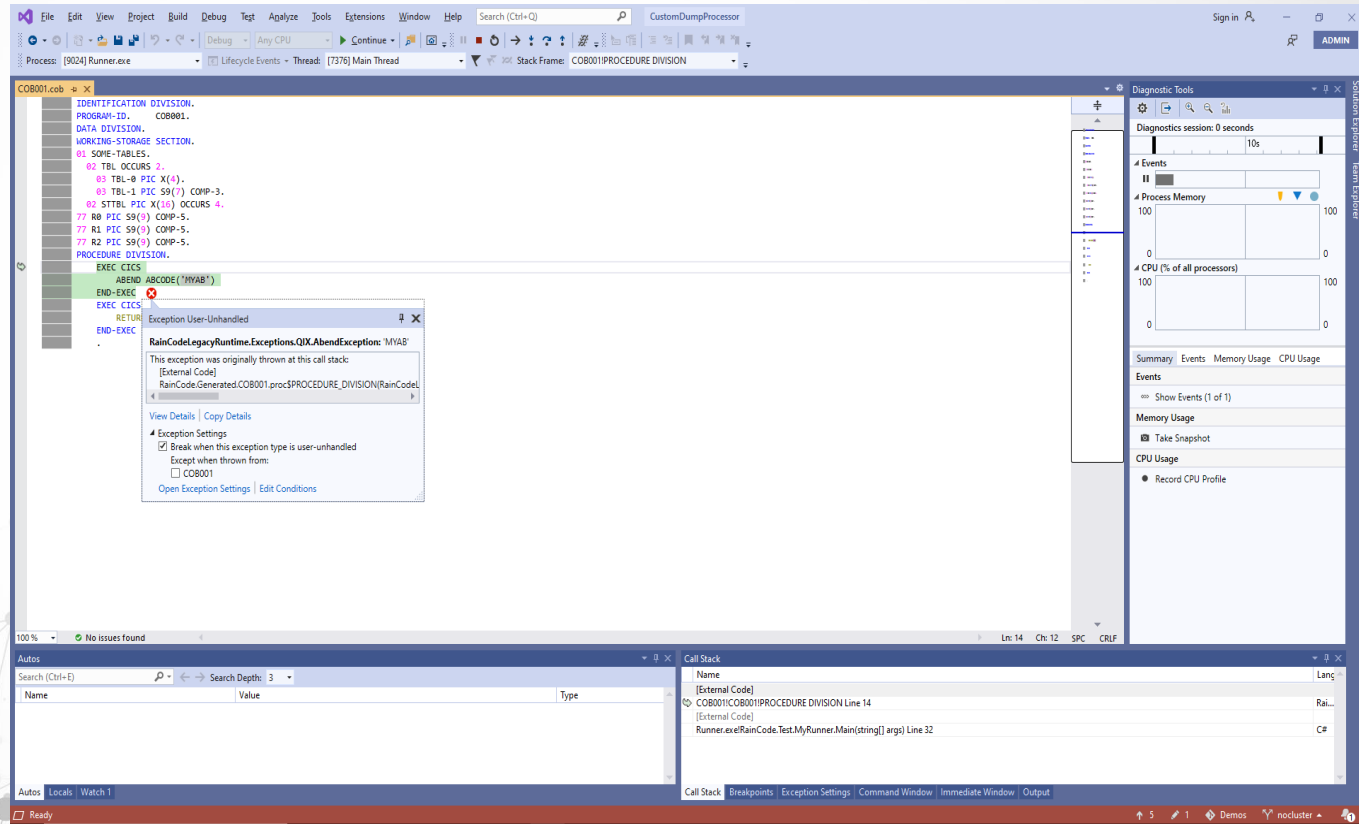


- Step 1: Open the solution from the folder Compiler - Dump Processor – Csharp Main

- Step 2: Start the Raincode debugger

Execution will hit a breakpoint in COB001.cob on the CICS ABEND statement

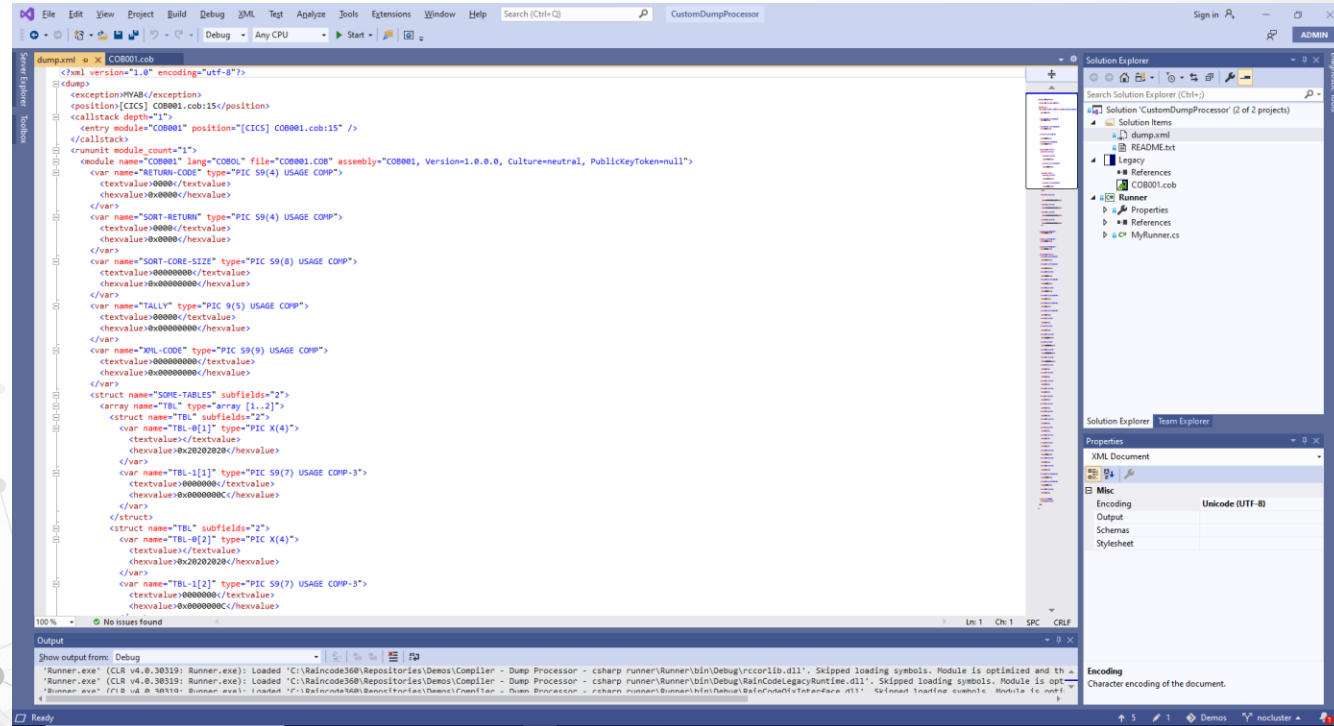
- Step 3: Click continue or close the application window to exit the debugger



Dump Processor – C# Main: How to see output



- Open the dump.xml from solution items
- The file contains the working storage of COB001.cob time of the crash plus the variables of the CICS Exec Interface Block (EIB)

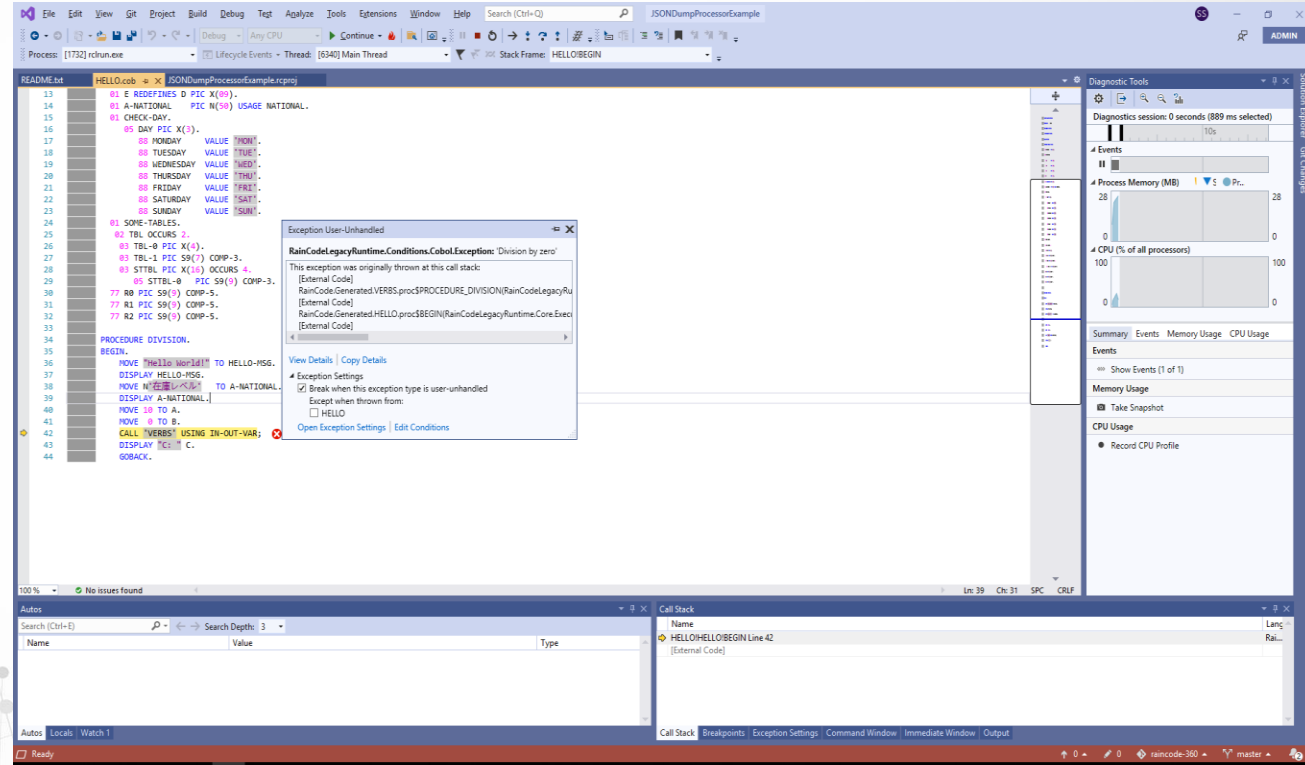


Dump Processor – Plugin - JSON: How to demonstrate



- Step 1: Open the solution from the folder Compiler - Dump Processor - Plugin - JSON

- Step 2: Start the Raincode debugger

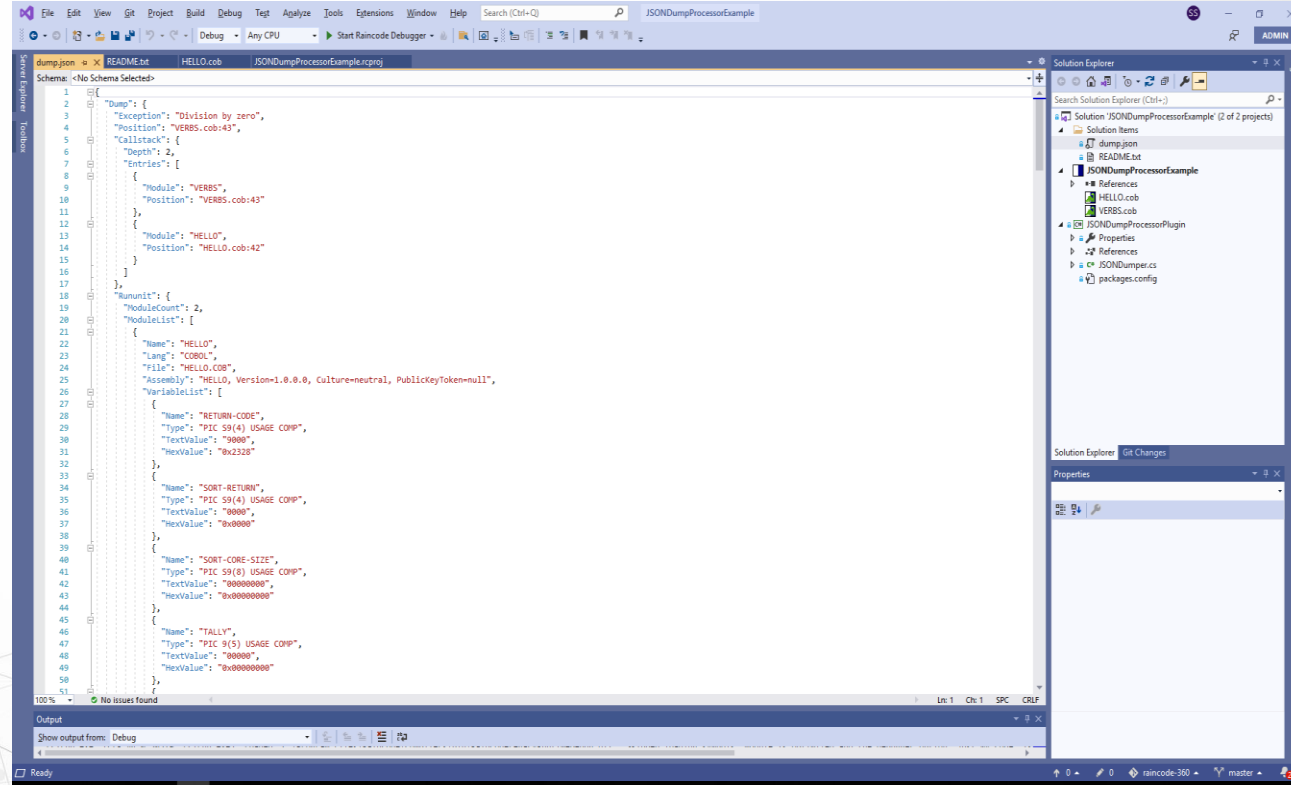


The execution will hit a breakpoint in HELLO.cob caused by a DivideByZeroException in the called module VERBS.cob.

- Step 3: Click continue or close the application window to exit the debugger

Dump Processor – Plugin - JSON: How to see output

- Open the dump.json from the solution items
- The JSON contains the working storage of the two COBOL programs involved at the time of the crash



Custom Logger



- Demonstrates the use of loggers in a C# runner.
 - Loggers are used to write formatted log messages to a particular target, such as the console or a dedicated log file.
 - The Log Level attribute of a logger can be set to show messages to that level or higher.
- Demonstrates how to create and register multiple loggers.
- Demonstrates how to set the log level and how to issue the additional messages.

For more details, see [Logging](#).

Custom Logger – How to demonstrate



- Step 1: Open the solution from the folder Compiler – Custom Logger

- Step 2: Select Start without debugging(Ctrl + F5).

```
mylog.txt  README.txt  hello.cob
IDENTIFICATION DIVISION.
PROGRAM-ID.      HELLO.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
DATA DIVISION.
WORKING-STORAGE SECTION.
    01 HELLO-MSG PIC X(12).
LINKAGE SECTION.
PROCEDURE DIVISION.

[TRACE] : DriverParameters =
[TRACE] : Creating IORuntime RecordBasedFile associated with SYSIN file
[TRACE] : SYSIN: set_open_mode Input
[TRACE] : SYSIN id = 2
[TRACE] : Creating file SYSOUT for instance
[TRACE] : get_handle logical_name=SYSOUT is -3
[TRACE] : SYSOUT not defined, creating default SYSOUT
[TRACE] : Configuring the file driver: [RainCodeLegacyFileDriver]RainCodeLegacyFileDriver.Driver.DotNetFHRRecordBaseFile
[TRACE] : configuring file
[TRACE] : file_name=SYSOUT path=SYSOUT
[TRACE] : file_organization=Sequential
[TRACE] : recording_mode=Fixed
[TRACE] : access_mode=Sequential
[TRACE] : lock_mode=Manual
[TRACE] : file_line_end_CRLF=True
[TRACE] : Printer ControlCharacters=ASA
1>----- [TRACE] : DriverParameters =
1>Restor [TRACE] : Creating IORuntime RecordBasedFile associated with SYSOUT file
1>Determ [TRACE] : SYSOUT: set_open_mode Output
1>All pro [TRACE] : SYSOUT id = 3
1>cobol - [TRACE] : HELLO: Static memory: 4312, 88
1>Skip [TRACE] : HELLO: Initializing static vars
[TRACE] : HELLO: Initializing static vars, done.
=====Hello World!
[TRACE] : Issuing STOP. RETC=0
Program 'hello' ended

C:\Raincode360\Repositories\Demos\Compiler - Custom Logger\main\bin\Debug\net8.0\main.exe (process 11576) exited with code 0 (0x0).
Press any key to close this window . . .
```

It will avoid automatic breakpoints and will keep the console window open after the program ends.

Custom Logger – How to see output



- You can also check the contents of mylog.txt. It should match what is shown in the console window.

The screenshot shows the Visual Studio IDE with the 'mylog.txt' file open in the editor. The file contains a series of log entries, each starting with a timestamp and a trace level, followed by a message. The messages describe the configuration of the CustomLogger, including the lock mode, file line end, printer control characters, driver parameters, and the creation of the IORuntime RecordBasedFile. The log also shows the initialization of static memory and the issuance of a STOP command.

```
11/18/2024 1:50:38 PM [TRACE] : lock_mode=Manual
11/18/2024 1:50:38 PM [TRACE] : file_line_end_CRLF=True
11/18/2024 1:50:38 PM [TRACE] : Printer ControlCharacters=ASA
11/18/2024 1:50:38 PM [TRACE] : DriverParameters =
11/18/2024 1:50:38 PM [TRACE] : Creating IORuntime RecordBasedFile associated with SYSIN file
11/18/2024 1:50:38 PM [TRACE] : SYSIN: set_open_mode Input
11/18/2024 1:50:38 PM [TRACE] : SYSIN id = 2
11/18/2024 1:50:38 PM [TRACE] : Creating file SYSOUT for instance
11/18/2024 1:50:38 PM [TRACE] : get_handle logical_name=SYSOUT is -3
11/18/2024 1:50:38 PM [TRACE] : SYSOUT not defined, creating default SYSOUT
11/18/2024 1:50:38 PM [TRACE] : Configuring the file driver: [RainCodeLegacyFileDriver]RainCodeLegacyFileDriver.Driver.DotNetFHRRecordB
11/18/2024 1:50:38 PM [TRACE] : configuring file
11/18/2024 1:50:38 PM [TRACE] : file_name=SYSOUT path=SYSOUT
11/18/2024 1:50:38 PM [TRACE] : file_organization=Sequential
11/18/2024 1:50:38 PM [TRACE] : recording_mode=Fixed
11/18/2024 1:50:38 PM [TRACE] : access_mode=Sequential
11/18/2024 1:50:38 PM [TRACE] : lock_mode=Manual
11/18/2024 1:50:38 PM [TRACE] : file_line_end_CRLF=True
11/18/2024 1:50:38 PM [TRACE] : Printer ControlCharacters=ASA
11/18/2024 1:50:38 PM [TRACE] : DriverParameters =
11/18/2024 1:50:38 PM [TRACE] : Creating IORuntime RecordBasedFile associated with SYSOUT file
11/18/2024 1:50:38 PM [TRACE] : SYSOUT: set_open_mode Output
11/18/2024 1:50:38 PM [TRACE] : SYSOUT id = 3
11/18/2024 1:50:38 PM [TRACE] : HELLO: Static memory: 4312, 88
11/18/2024 1:50:38 PM [TRACE] : HELLO: Initializing static vars
11/18/2024 1:50:38 PM [TRACE] : HELLO: Initializing static vars, done.
11/18/2024 1:50:38 PM [TRACE] : Issuing STOP. RETC=0
```

The Solution Explorer on the right shows the project structure, including the 'main' project and the 'mylog.txt' file. The 'mylog.txt' file is highlighted in the Solution Explorer.

Legacy with database access - Csharp main

Legacy with database access - Csharp main – Introduction



- C# calling Legacy with database access using a COBOL call
- Demonstrates how a COBOL program can be called from a C# main program (a proxy)
- The C# program uses the regular COBOL call mechanism
- Demonstrates how to prepare an execution context with a database connection at the C# level
- In this demo, the legacy program fetches an item (maximum zip code) from the database, displays it on the console and exits

Legacy with database access - Csharp main – How to demonstrate



- Step 1: Open the solution from the folder Compiler - Legacy with Database Access - Csharp Main
- Step 2: Select Start without debugging (Ctrl + F5)

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\WINDOWS\system32\cmd.exe'. The window has a black background with white text. The text inside the window reads: 'Maximum zipcode: 9992' followed by 'Press any key to continue . . .' on the next line. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

```
C:\WINDOWS\system32\cmd.exe
Maximum zipcode: 9992
Press any key to continue . . .
```

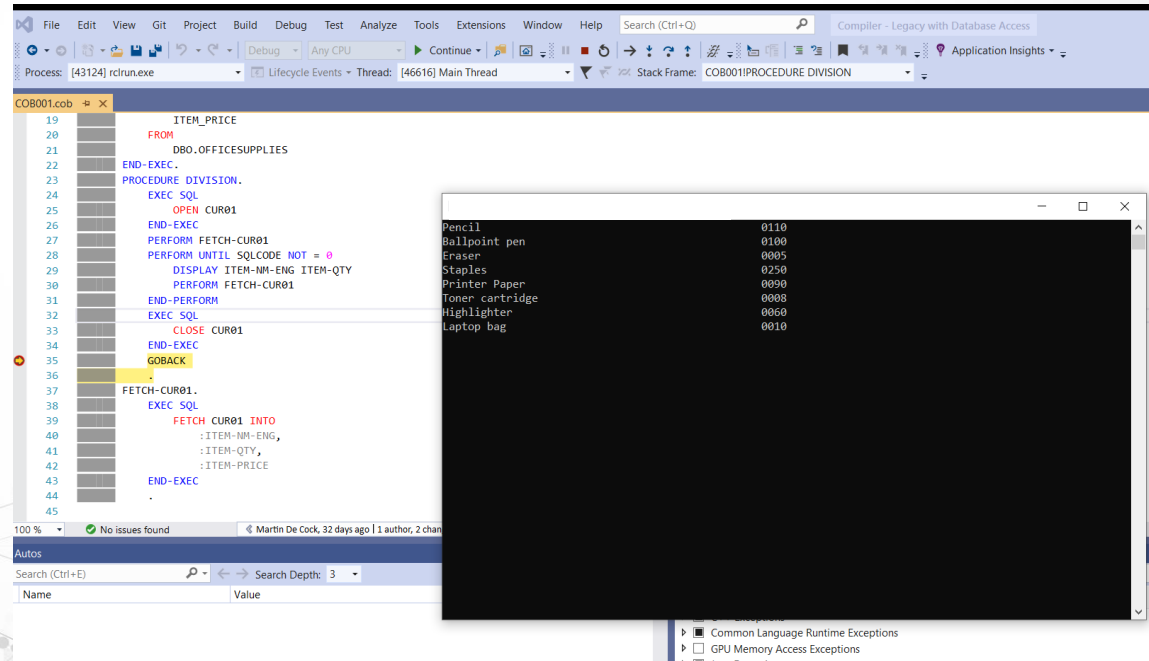
It will avoid automatic breakpoints and will keep the console window open after the program ends.

Legacy with database access

Legacy with database access



- Demonstrate how to connect a COBOL program directly to a database
- Step 1: Open the solution from the folder Compiler - Legacy with Database Access
- Step 2: Put a breakpoint on the GOBACK statement
- Step 3: Start the Raincode Debugger



Legacy calling Csharp with Helper class

Legacy calling CSharp with helper class



- Demonstrates how a COBOL program can transparently call C# routine (as if it were COBOL).
- Demonstrates how the C# routine can use a helper class to make sense of the data being passed to it by the COBOL program.
- This example needs a CSharp main because the CSharp routines (cshello.cs and cssquare.cs) must be added to the module dictionary before the COBOL programs can find them.
- HELLO.cob calls CSHELLO.cs without a helper class. The parameter being passed is a simple string.
- COBSQUARE.cob calls CSSQUARE.cs. The parameters are passed via the copybook W-STRUCT of COBSQUARE.cob. W-STRUCT has a complex structure.
- CSSQUARE.cs uses the generated helper class WStruct.cs to access the parameters.

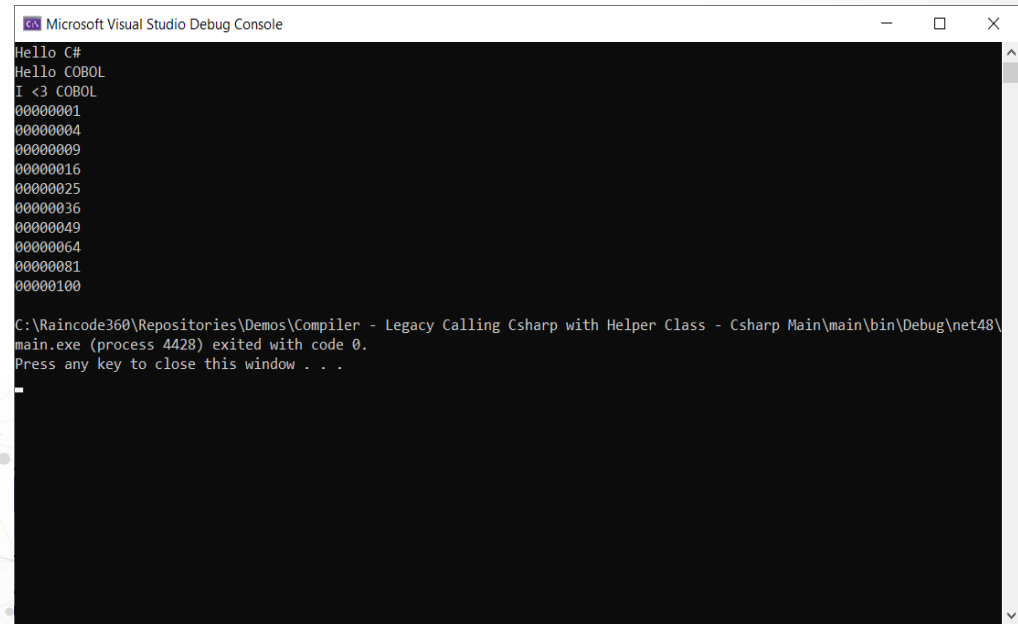
Legacy calling C# with helper class – How to demonstrate



- Step 1: Open the solution from the folder Compiler - Legacy calling CSharp with helper class
- Step 2: Start without debugging (Ctrl-F5)

Optionally, run Helper.bat in the solution directory to re-generate the helper class (WStruct.cs in the CSharp folder)

Output



```
Microsoft Visual Studio Debug Console
Hello C#
Hello COBOL
I <3 COBOL
00000001
00000004
00000009
00000016
00000025
00000036
00000049
00000064
00000081
00000100
C:\Raincode360\Repositories\Demos\Compiler - Legacy Calling Csharp with Helper Class - Csharp Main\main\bin\Debug\net48\
main.exe (process 4428) exited with code 0.
Press any key to close this window . . .
```

Standalone SQL Conversion

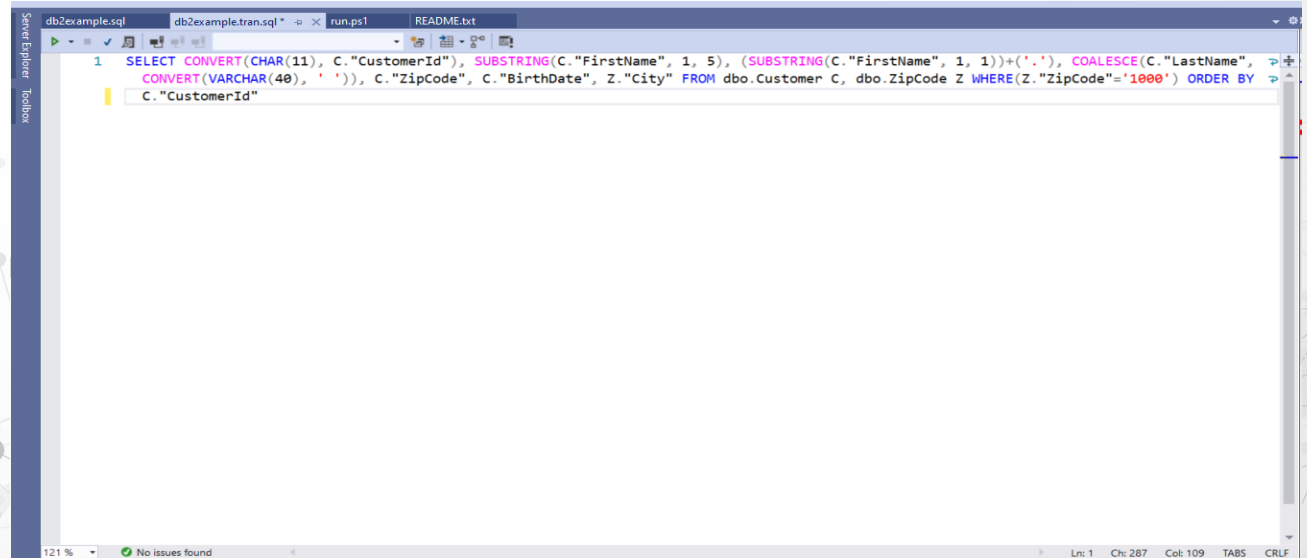
Standalone SQL Conversion



- Illustrates how the compiler can be used to convert SQL statements that are not part of a program
- Useful for SQLs that appear inside scripts or JCLs
- Includes an example of script-based transformation
- Input is taken from the file db2example.sql

- The output is written to db2example.tran.sql

- The rewrite procedure's impact is not visible in the output file as they convert the input before conversion.



```
1 SELECT CONVERT(CHAR(11), C."CustomerId"), SUBSTRING(C."FirstName", 1, 5), (SUBSTRING(C."FirstName", 1, 1))+('.'), COALESCE(C."LastName",  
CONVERT(VARCHAR(40), ' '), C."ZipCode", C."BirthDate", Z."City" FROM dbo.Customer C, dbo.ZipCode Z WHERE(Z."ZipCode"='1000') ORDER BY  
C."CustomerId"
```

File Access via FileConfig

File Access via FileConfig - Introduction

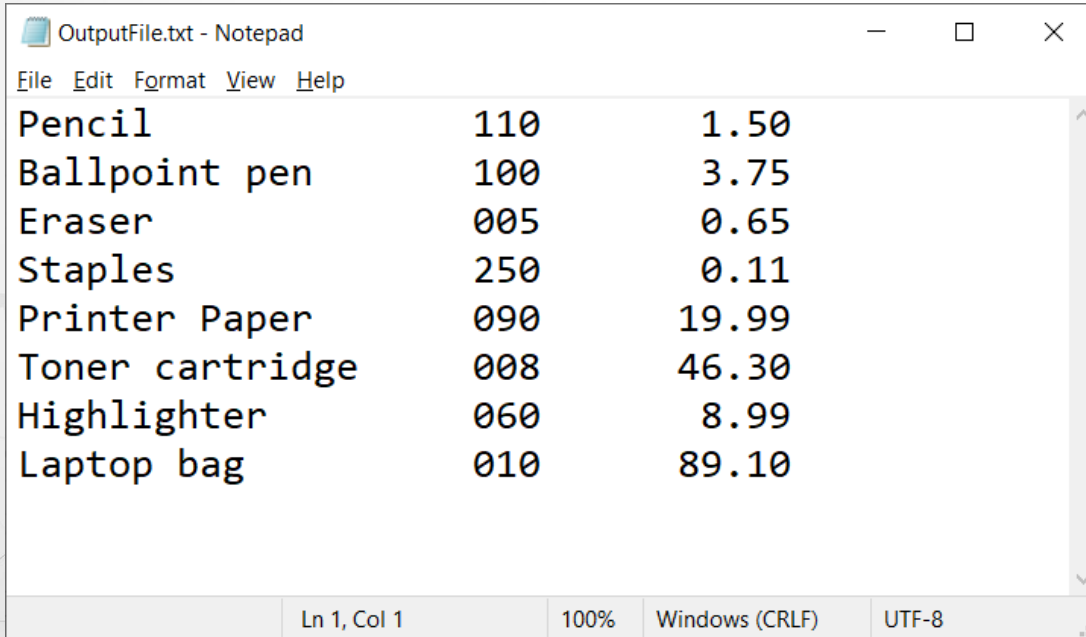


- When not running in the context of a JCL, legacy programs need the help of a FileConfig to locate external files.
- A FileConfig is an XML file that links logical file names to the actual file names on the disk.
- A FileConfig can contain many attributes that depend on the file type.
 - The logical and physical name is sufficient for this demo
- This demo uses FileConfig to write to a file
- The main program reads a table in the database and writes this information to an output file.

File Access via FileConfig - Demo



- This demo uses FileConfig to write to a file.
- The main program reads a table in the database and writes this information to an output file.
- Press Start RainCode Debugger/F5/ Ctrl-F5 to start the demo.



Pencil	110	1.50
Ballpoint pen	100	3.75
Eraser	005	0.65
Staples	250	0.11
Printer Paper	090	19.99
Toner cartridge	008	46.30
Highlighter	060	8.99
Laptop bag	010	89.10

The output file is defined as a solution item (OutputFile.txt)

National Character Output on Console

National Character Output on Console



- National character strings are declared as PIC N(n) USAGE NATIONAL.
- Each character is represented by two or four bytes.
- Numeric edited variables can also be declared with USAGE NATIONAL.
- Supported operations are:
 - computations,
 - output to the console,
 - viewing values in the debugger,
 - reading and writing to a database.

National Characters Output on Console



- To compile programs with the embedded national characters, **MOVE N'世界 ! ' TO REC**, the compilation command line must contain the following option:
StringSourceFileEncoding = utf-8
- National characters are encoded in UTF-16 big endian format.

Note: If the console cannot display Unicode characters, change the language to Japanese in [configuration/time and language/language/administrative language settings/change system locale] to make the Japanese characters visible. This requires rebooting your machine.

National Character Output on Console – Demo



- This demo shows how national characters can be written to files
- This demo reads English and Japanese descriptions from a table file
- Step 1: Open the solution from the folder Compiler – National Character Output on Console.
- Step 2: Place a breakpoint at GOBACK in COB001.cob and click on Start Raincode Debugger to execute the project.

```
在庫レベル
-----
鉛筆
ボールペン
消しゴム
ステープル
プリンター用紙
トナーカートリッジ
蛍光ペン
ノートパソコン用のバッグ

何かキーを押すと続行します ...
```

Output

User defined function to implement a coding guideline

User defined function to implement a coding guideline



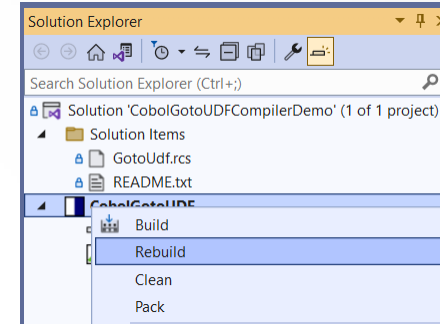
- Using the Raincode scripting language, create a user-defined function to implement a coding guideline.
- This UDF adds extra validation to the compiler to throw an error when the "GO TO" statement is encountered in the COBOL module.

User defined function – How to demonstrate in Normal flow



- Step1: Right-click on "CobolGotoUDF" from the solution explorer and select the "Rebuild" option.

```
CobolGotoUDF.rcproj  GOTOCBL.cob  README.txt
IDENTIFICATION DIVISION.
PROGRAM-ID.    GOTOCBL.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
DATA DIVISION.
WORKING-STORAGE SECTION.
    01 WS-X PIC 9(01).
LINKAGE SECTION.
PROCEDURE DIVISION.
PARA-0.
    DISPLAY 'IN PARA-0'
    MOVE 1 TO WS-X
    GO TO PARA-1 PARA-2 DEPENDING ON WS-X
    DISPLAY 'IN PARA-0 AGAIN'
    STOP RUN.
PARA-1.
    DISPLAY 'IN PARA-1'.
    GOBACK.
PARA-2.
    DISPLAY 'IN PARA-2'.
    GOBACK.
```



- Step 2: Once the build is successful. Open GOTOCBL.cob from the solution explorer and set a breakpoint at the GOBACK statement in PARA-1.

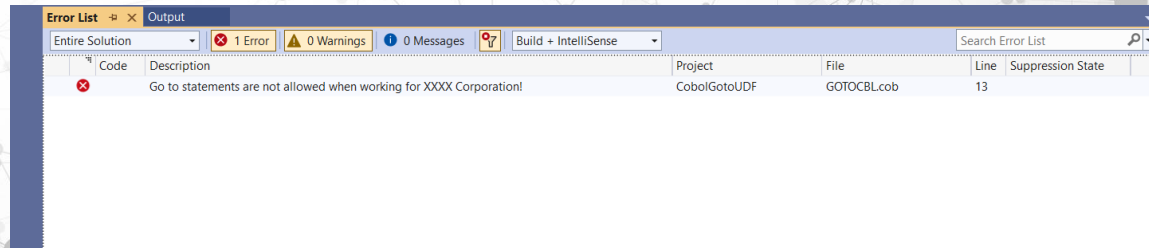
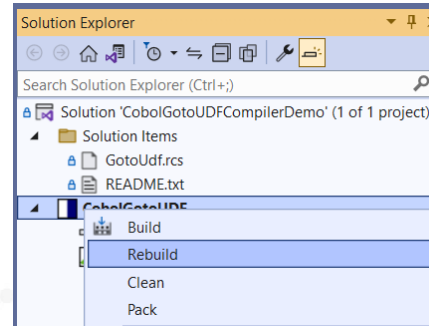
- Step 3: Start the Raincode Debugger.
- Step 4: The execution will print "IN PARA-0" and "IN PARA-1" in consecutive lines. Close the application window to exit the debugger.

```
[2022-07-01 09:05:10.473] [1] [WARNING] [RainCode.Core.Comm
object to change at any time.
IN PARA-0
IN PARA-1
-
```

User defined function – How to demonstrate in Error flow



- Step 1: Click on "CobolGotoUDF" from the solution explorer.
- Step 2: From the displayed xml lines, uncomment i.e remove tags `<!--` and `-->` from the line "`<UserDefinedParameters>:ScriptPaths="$(SolutionDir)" :UserDefinedValidationProc="GotoUdf.Validate"</UserDefinedParameters>`" and save the changes by pressing Ctrl +S.
- Step 3: Right click on "CobolGotoUDF" from the solution explorer and chose "Rebuild" option.
- Step 4: The compilation fails, and you see the error message "Goto statements are not allowed when working for XXXX Corporation!" in the Output tab.



Timeshift



- Timeshift is a feature that enables you to run programs at any particular time. It accomplishes this by manipulating the current date and time built into the Raincode runtime.
- With some additional programming, this mechanism can be further expanded to include the current date and time in embedded SQL.

Timeshift – How to demonstrate



- Step 1: Execute the script `Stored Procedures.sql` and `Stored procedures\_TimeConversion\_Raincode.sql` in the Bankdemo database.
- Step 2: Start the Raincode debugger

```
GO
CREATE OR ALTER FUNCTION [dbo].[CURRENTTIMESTAMP2] ()
RETURNS datetime2
AS
BEGIN
    DECLARE @datetimeResult datetime2
    DECLARE @starting_date datetime2 = CURRENT_TIMESTAMP
    DECLARE @ticks bigint = CONVERT(BIGINT, SESSION_CONTEXT('N'RAINCORE_DATE'))
    IF (@ticks IS NULL)
        RETURN @starting_date
    IF (@ticks < 0)
        BEGIN
            -- Hours
            SET @datetimeResult = DATETIME2(HOUR, CEILING(@ticks / 3600000000.0), @starting_date)
            SET @ticks = @ticks % 3600000000
            -- Seconds
            SET @datetimeResult = DATETIME2(SECOND, CEILING(@ticks / 1000000.0), @datetimeResult)
            SET @ticks = @ticks % 1000000
        END
    RETURN @datetimeResult
END
```

100 %
Messages
Commands completed successfully.
Completion time: 2024-11-29T09:24:16.0596879+00:00

Expected Output:

```
COBOL TS: 2032041818082831+000
SQL    TS: 2032-04-18 18:08:28.383333
-
```

RadaR



- **RadaR** (Raincode Database Runner) allows the execution of the code compiled by the Raincode legacy compilers locally on the server that hosts the database. This minimizes latency and maximizes communication throughput between the program and the database.
- This demo compares the traditional client/server approach with **RadaR**



Prerequisite:

Raincode Stack installation is to be done on the database server. For more details on installing Raincode Stack, refer to the readme available in the folder Benchmark – VsamSql with RadaR under solution items.

How to run JCL without RadaR:

Open the solution from the Benchmark–VsamSql folder with RadaR and press the start Raincode Debugger on the main menu; it will execute the JCL without RadaR.

(JCL runs a PGM=FSET as a second step)

Output – JCL without RadaR



- Once the demo is executed successfully, the output file (SYSLOG.JCL002.txt) will be available in the folder SYSOUT at C:\ProgramData\Raincode\Batch.

Output:

```
SYSLOG.JCL002.txt - Notepad
File Edit Format View Help
SYSLOG -- raincode360-vm

Date: Friday, May 3, 2024
Version: Raincode JCL v4.2.323.0
User: raincode360-vm\Martin

Accounting Information:
Programmer Name:
Job Name: JCL002
Job Number: 0000000097
Job Class:
Restart Step: *
DSNAME:
JCL: C:\Raincode360\Repositories\Demos\Benchmark - VsamSql with RadaR\RadarDsn\JCL002.jcl

Job Step Stats:
-----
Step Name      Proc Step      Return Code      Sequence      Bytes Read      Bytes Written      Step Duration
STEP001         1                0                1                0                0                00:00:00.7180031
STEP002         2                0                2                0                0                00:00:17.0244880

Job Overall Stats:
-----
Steps: 2
Job Start Time: 12:50:22
Job End Time: 12:50:41
Job Duration: 00:00:18.5412101
Result Code: NORMAL - CODE(0) RC(0)
```

The output without RadaR shows that both steps were executed successfully, and the second step took 17 seconds.

Output - JCL with RadaR



How to run JCL with RadaR:

To make the JCL run on the database server instead of the client, open the PowerShell from Tools->Command Line -> Developer PowerShell and run **FSET.ps1**.

Expected Output:

```
SYSLOGJCL002.txt - Notepad
File Edit Format View Help
$SYSLOG -- raincode360-vm

Date: Friday, May 3, 2024
Version: Raincode JCL v4.2.323.0
User: raincode360-vm\Martin
Accounting Information:
Programmer Name:
Job Name: JCL002
Job Number: 0000000098
Job Class:
Restart Step: *
DSNAME:
JCL: C:\Raincode360\Repositories\Demos\Benchmark - VsamSql with RadaR\Radarsn\JCL002.jcl

Job Step Stats:
-----
Step Name      Proc Step      Return Code      Sequence      Bytes Read      Bytes Written      Step Duration
STEP001                0                1                0                0                00:00:00.7698388
STEP002                0                2                0                0                00:00:02.6058503

Job Overall Stats:
-----
Steps: 2
Job Start Time: 13:02:41
Job End Time: 13:02:45
Job Duration: 00:00:04.1941841
Result Code: NORMAL - CODE(0) RC(0)
```

The output with RadaR shows that both steps were executed successfully, and the second step took 2.6 seconds.

RadaR - Comparison of outputs

● JCL



17 SECONDS

● JCL
+
● RadaR



2.6 SECONDS

PERFORMANCE BOOST = 6.5



Raincode Console

Raincode Console



- Raincode Console is a tool that allows you to manage IMS, QIX and Vsam Sql systems in one place.

- Double-click on the shortcut (Raincode console) placed on the desktop, and it will launch the tool in the web browser



Welcome to Raincode Console

Connection Name	Type	
Development	IMSql	→ CONNECT
Production	IMSql	→ CONNECT
QA	IMSql	→ CONNECT
Development	QIX	→ CONNECT
Production	QIX	→ CONNECT
QA	QIX	→ CONNECT
Development	VsamSql	→ CONNECT
Production	VsamSql	→ CONNECT
QA	VsamSql	→ CONNECT

1 - 9 of 9 items

- For more details, refer to the [Raincode Console](#) documentation.



- IMSql panel allows you to manage an IMS system. It allows you to view and manage IMS/DB and IMS/TM

 ● IMSql Configuration ▾ CheckPoints Terminals Hosts Diagrams Catalog ▾ Service Broker ▾ Monitoring ▾ [CHANGE CONNECTION](#)

Welcome to IMSql.


Systems


Regions


Programs


Mappings


Terminals


Hosts


Diagrams


DBDs


PSBs


Services

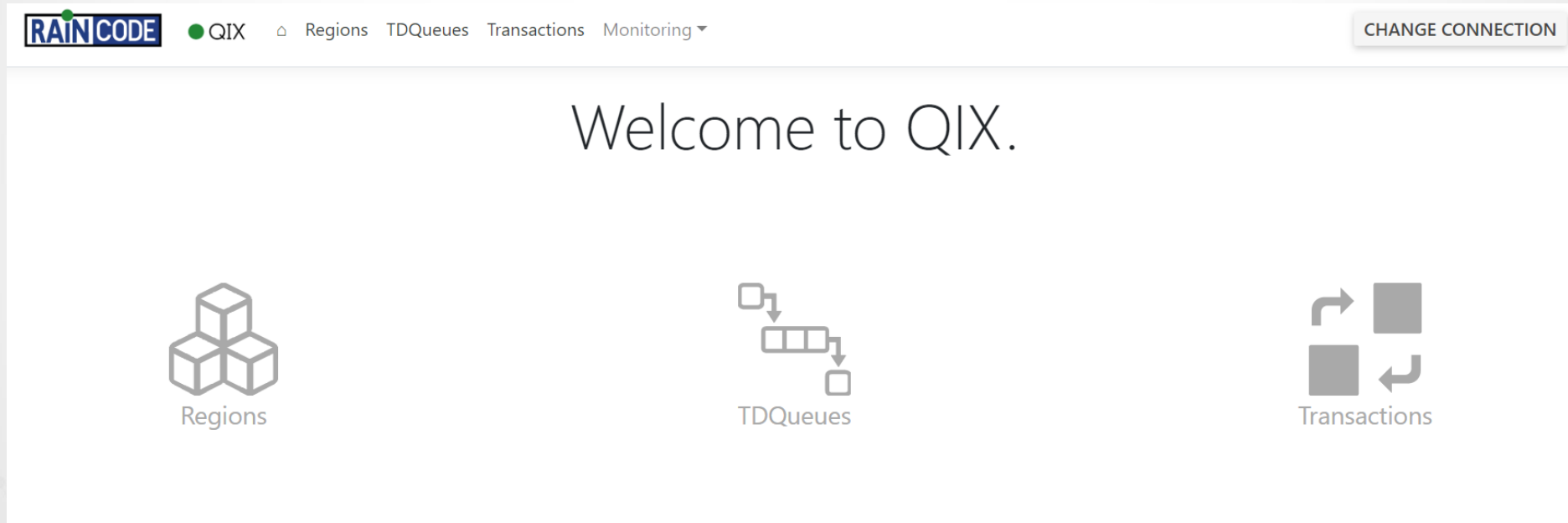

Queues


Conversations

© 2019-2024 Raincode srl - IMSql - [Privacy](#)



- QIX panel allows you to manage a QIX system such as configuring a region, transactions, etc.



Raincode Console - Vsam Sql



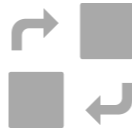
VSAM Sql panel allows you to manage a VSAM system, such as viewing the dataset and table mappings and deleting the datasets and their data.



● VsamSql △ Mappings

CHANGE CONNECTION

Welcome to VsamSql.



Mappings

Raincode Batch: JCL Interpreter



Architecture

Scaffolding of input parsers
(support for Autoedit, for instance).

Windows file system locking primitives for
synchronization

Catalog management as a reusable component

Comprehensive DF-SORT utility

Parser generator for additional utilities

Scheduler

Externalized



Focus on scale and flexibility

Support for multi-year migration
when big bang is not an option

```
000100 //IEFBRI41 JOB 'AK99', AVINASH, NOTIFY=&SYSUID
000200 //*****
000300 //*THIS UTILITY USED TO ALLOCATE OR DEALLOCATE A DATASET
000400 //*CREATES BOTH PS AND PDS FILES ALSO DELETES SAME
000410 //*****
000420 //MYPROC PROC ----->MYPROC IS PROCEDURE NAME THEN PROC THIS IS SYNTAX
000500 //DELETE EXEC PGM=IEFBRI4
000600 //SYSOUT DD SYSOUT=*
000700 //SYSPRINT DD SYSOUT=*
000800 //SYSUT1 DD DSN=AK999K.AVINASH.PS4,
000900 // DISP=(NEW,CATLG,DELETE),
000910 // DCB=(LRECL=80,BLKSIZE=8000,RECFM=FB),
000920 // SPACE=(TRK,(2,3),RLSE)
000930 // PEND ----->END OF PROC.
000940 *****
000950 //STEP2 EXEC MYPROC ----->HERE I AM CALLING MY INSTREAM PROC IN SAME JCL
001000 //SYSIN DD DUMMY
001100 //
```



JCL Contains:

- IKJEFT01
- IDCAMS
- SORT
- COBOL Program:
 - Launched by IKJEFT01
 - Database access
 - Uses CICS LINK to access a PL/I program
- Legacy calling assembler
- C# launcher
- JCL examples:
 - DSNTIAUL
 - IEBGENER
 - DSNUTLIB load and unload



Demos:

- [Legacy Calling Assembler](#)
- [Legacy with Database Access –EBCDIC Version](#)
- [SORT and IKJEFT01](#)
- [DSNTIAUL](#)
- [DSNUTILB](#)
- [Batch – File Types](#)
- [Generation Data Group](#)
- [Partitioned Dataset](#)
- [Catalog Explorer](#)
- [Automatic Restart](#)
- [Syncsort Translator](#)
- [Files with National Character](#)
- [FBA with EBCDIC](#)
- [zTrieve](#)
- [Legacy with MultiDatabase Access](#)
- [Unload MultiDatabase](#)

Legacy Calling Assembler

Legacy Calling Assembler

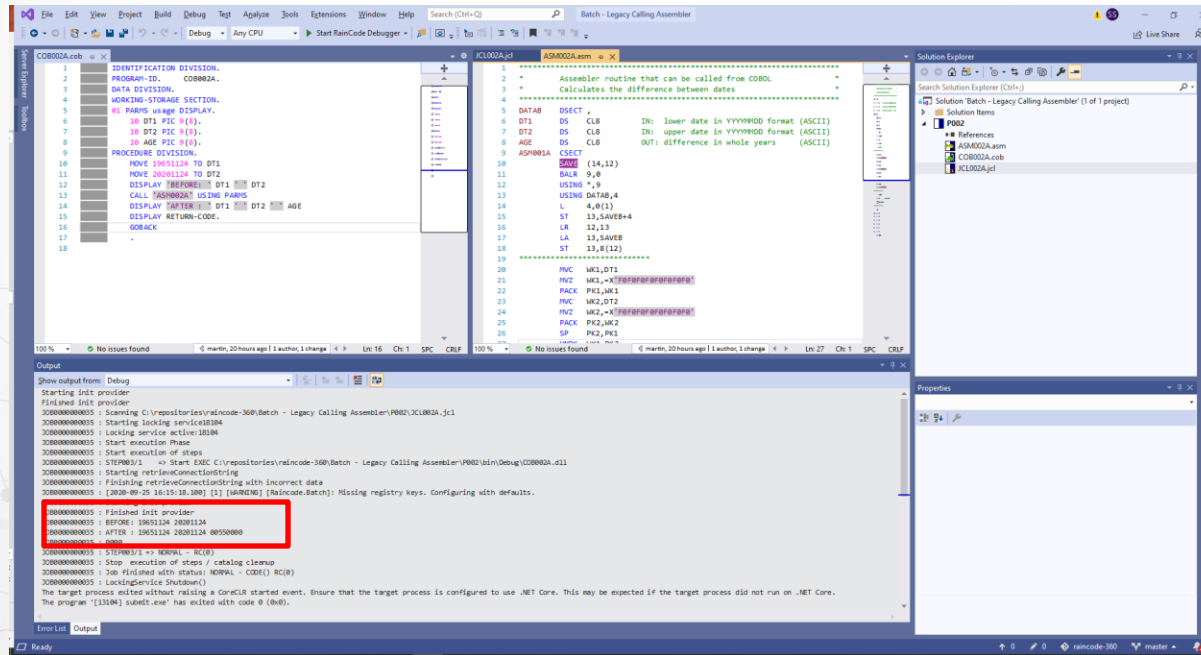


- This demo is an example of a JCL executing a COBOL program with no database access.
- The COBOL program illustrates how to call a subroutine written in Assembler.
- The assembler routine calculates the difference in years between the two dates that are passed as parameters by the COBOL main program.

Legacy Calling Assembler – How to demonstrate



- Step 1: Open the solution from the folder Batch - Legacy Calling Assembler
- Step 2: Start Raincode Debugger, and after successful debugging, it will calculate the difference between two dates.



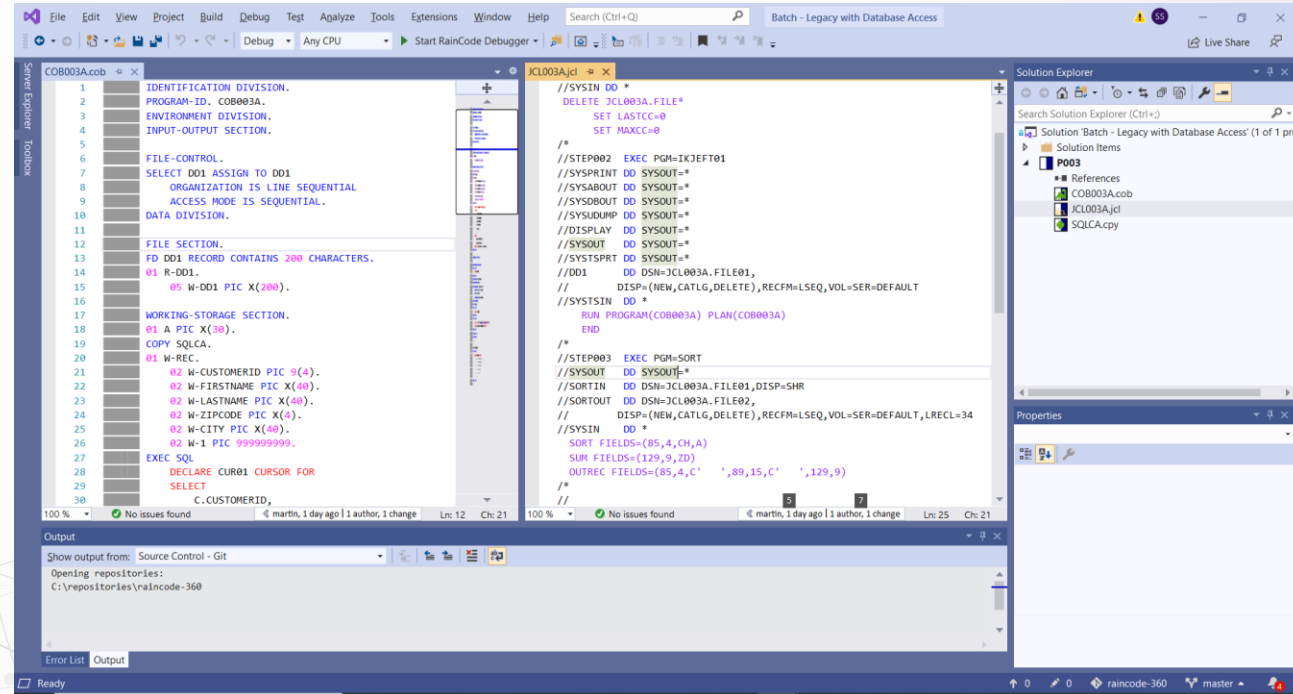
SORT and IKJEFT01

SORT and IKJEFT01



- Connect a COBOL program to the BANKDEMO database via IKJEFT01
- Example: sort then aggregate the number of existing customers per zip code.

- Step 1: Open the solution from the folder Batch – Legacy with Database Access
- Step 2: Press Start Raincode Debugger to compile and run the project



SORT and IKJEFT01 – Where to See Output Files and Job Queue



- Output and job queue created will be available at C:\ProgramData\Raincode\Batch
- Navigate to the DefaultVolume folder to the see output files
- Navigate to the SYSOUT folder to see the Job Queue
- Output of the job that just ran, together with the previous run's output is available here.
- The highest number represents the most recent run.

This PC > Windows (C:) > programdata > Raincode > Batch > SYSOUT >

Name	Date modified	Type	Size
JOB000000001.JCL003A	6/4/2020 10:42 AM	File folder	
JOB000000002.JCL003A	6/4/2020 10:53 AM	File folder	
JOB000000003.JCL004A	6/4/2020 2:04 PM	File folder	
JOB000000004.JCL004A	6/4/2020 2:12 PM	File folder	
JOB000000005.JCL004A	6/4/2020 2:14 PM	File folder	
JOB000000006.JCL003A	6/8/2020 8:35 AM	File folder	
JOB000000007.JCL004A	6/8/2020 8:38 AM	File folder	
JOB000000008.JOB001	6/9/2020 9:52 AM	File folder	
JOB000000009.JOB001	6/9/2020 9:53 AM	File folder	
JOB000000010.JOB001	6/9/2020 9:53 AM	File folder	
JOB000000011.JOB001	6/9/2020 9:53 AM	File folder	
JOB000000012.JOB001	6/9/2020 11:31 AM	File folder	
JOB000000013.JOB001	6/9/2020 11:33 AM	File folder	
JOB000000014.JOB001	6/9/2020 11:33 AM	File folder	
JOB000000015.JOB001	6/9/2020 1:47 PM	File folder	
JOB000000016.JCL003A	6/9/2020 2:57 PM	File folder	
JOB000000017.JOB001	6/10/2020 9:42 AM	File folder	
Sysout.lock	6/10/2020 9:42 AM	LOCK File	1 KB

Legacy with Database Access – EBCDIC Version

Legacy with Database Access – EBCDIC version



- This demo produces file output in EBCDIC
- This demo is a modified version of the demo 'Batch - Legacy with Database Access'
- Differences:
 - The Encoding is EBCDIC rather than ASCII
 - The files are FB rather than LSEQ
- The sort utility is installed in ASCII mode, so the character literal C' ' (three spaces) in the ASCII version was replaced by X'404040' to produce readable EBCDIC.

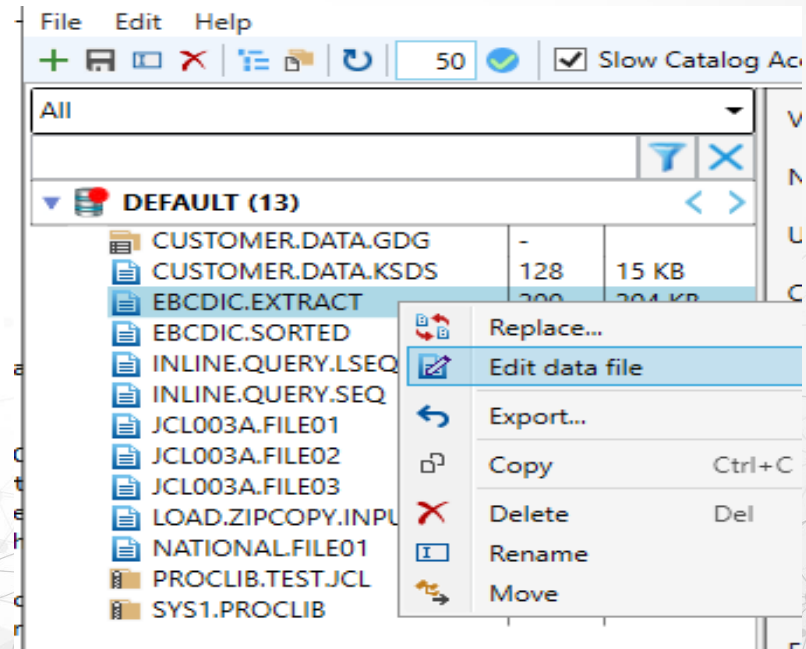
Legacy with Database Access – EBCDIC version –How to demonstrate



- Step 1: Open the solution from the folder Batch - Legacy with Database Access - EBCDIC Version
- Step 2: Press Start Raincode Debugger to compile and run the project

You can examine the output file (EBCDIC.EXTRACT and EBCDIC.SORTED) using the Catalog Explorer

- Step 3: Open the Catalog Explorer by double click on the Catalog Explorer shortcut placed on the desktop
- Step 4: Right-Click the output file and select Edit data file from the context menu.



The output file will open in the Record Editor

Legacy with Database Access – EBCDIC version



- Step 5: Click on the File details icon and select Code Page as 37 from the drop-down menu of Encoding

- Step 6: Press Reload File to load the file in EBCDIC.

EBCDIC.EXTRACT - :DEFAULT:EBCDIC.EXTRACT - Record Editor

File Tools Record Help

Page size: 50 ☒ Hide hex values

2 Go

<Record#> 0x *? <Enter value to search records>

Record Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14
51	0605Arline		Prince						1000BRUSSEL				00000001	
52	0617Julio		Jones						1000BRUSSEL				00000001	
53	0629Harland		Jimenez						1000BRUSSEL				00000001	
54	0641Susanne		Werner						1000BRUSSEL				00000001	
55	0653Wayne		Kaufman						1000BRUSSEL				00000001	
56	0665Kristopher		Jarvis						1000BRUSSEL				00000001	
57	0677Sherri		Dunlap						1000BRUSSEL				00000001	
58	0689Russ		Shepard						1000BRUSSEL				00000001	
59	0701Beatriz		Sloan						1000BRUSSEL				00000001	
60	0713Katy		Ryan						1000BRUSSEL				00000001	
61	0725Dane		Pugh						1000BRUSSEL				00000001	
62	0737Robert		Ray						1000BRUSSEL				00000001	
63	0749Ethan		Galvan						1000BRUSSEL				00000001	
64	0761Ivory		Banks						1000BRUSSEL				00000001	
65	0773Anibal		Vasquez						1000BRUSSEL				00000001	
66	0785Cyril		Zamora						1000BRUSSEL				00000001	
67	0797Brigitte		Khan						1000BRUSSEL				00000001	
68	0809Edgar		Palmer						1000BRUSSEL				00000001	
69	0821Willia		Kennedy						1000BRUSSEL				00000001	
70	0833Sergio		Peck						1000BRUSSEL				00000001	

Ready Code Page: 37 Modify file details: Succeeded

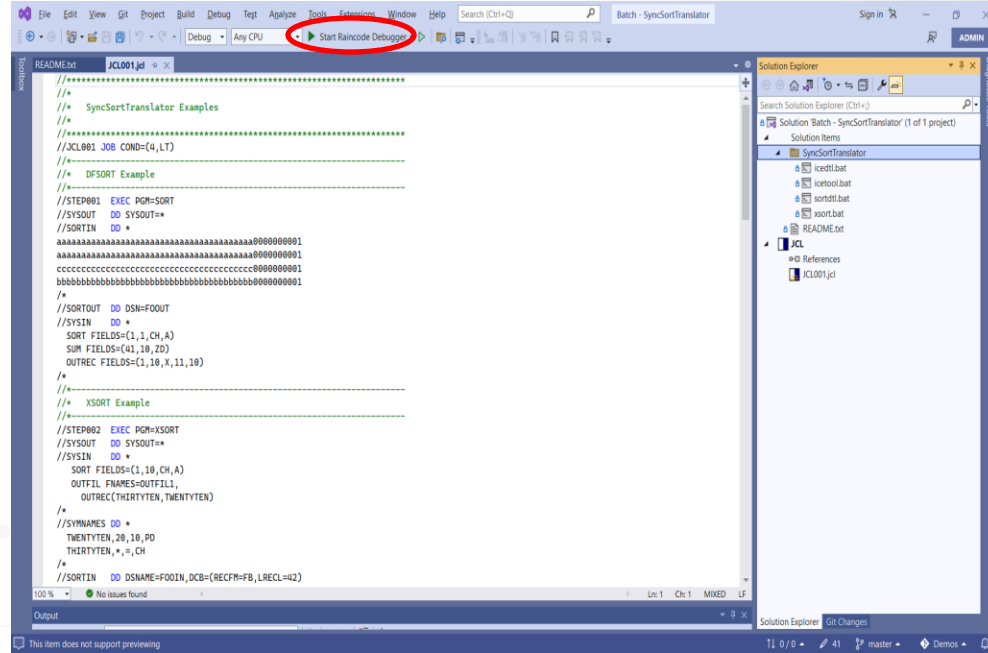
Syncsort Translator

Syncsort Translator



- The Raincode Syncsort Translator allows for SORT and ICETOOL steps in your JCL to be executed by Syncsort.
- Once installed, the Syncsort Translator can be invoked by invoking XSORT/SORTDTL to translate the SORT syntax, or by invoking ICETRANS/ICEDTL to translate the ICETOOL syntax. No further changes to the JCL are needed.
- Syncsort is a third-party sort utility that is not installed as part of the demo.
- The demo is a single JCL with each step illustrating a different sort method.

Step	Program	Input Syntax	Sort Type	Target Syntax
1	SORT	DFSORT	basic	DFSORT (no translation)
2	XSORT	DFSORT	basic	Syncsort
3	SORTDTL	same as 2	basic	Syncsort DTL
4	SORTDTL	DFSORT	join	Syncsort DTL
5	ICETOOL	ICETOOL	splice	Syncsort
6	ICEDTL	same as 5	splice	Syncsort DTL



Syncsort Translator – How to demonstrate



- Step 4: Open the job output SYSOUT.seq, and you can find the location for the translated output file

≡ SYSOUT.seq X

C: > ProgramData > Raincode > Batch > SYSOUT > JOB0000000067-JCL001 > STEP002 > ≡ SYSOUT.seq

```
1 "Starting translator"
2 Starting SyncsortTranslator : -FromJCL=true -WorkDir=workspace_directory -OutputDir=C:\Users\RAINCO~1\AppData\Local\Temp\2\RC_SSXLate
3 defaultRecordLength is Different of 0 This will not permit detection of non specified LREC VALUE. Value is Ignored
4 Starting init provider
5 Finished init provider
6 "Translation finished"
7 "Starting Syncsort"
8 "SyncSort Finished"
9 "Step Done"
10
```

- Translated Output

JCL001-STEP002_2.dme.001.txt - Notepad

File Edit Format View Help

```
/FIELDS Src_0 1 10 CHARACTER
/INFILE C:\ProgramData\Raincode\Batch\DefaultVolume\F00IN.seq MAINFRAMEFIXED 42
/KEYS Src_0 ASCENDING
/COLLATINGSEQUENCE DEFAULT EBCDIC
/OUTFILE C:\ProgramData\Raincode\Batch\DefaultVolume\F00OUT.seq MAINFRAMEFIXED 42 OVERWRITE
/END
```

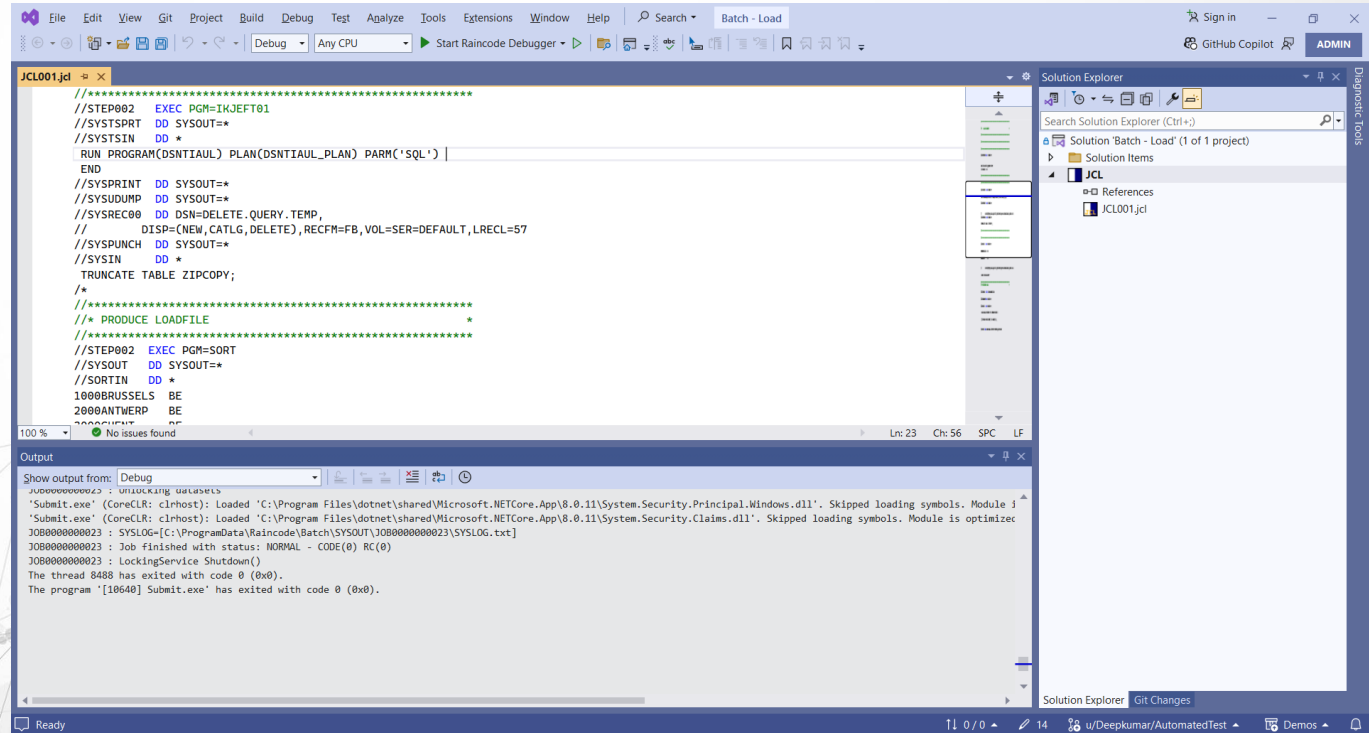
DSNTIAUL

- Output file created will be available at:
C:\ProgramData\Raincode\Batch\DefaultV
olume\Inline



DSNUTILB

- Step 2: Press Start Raincode Debugger to compile and run the project



Loading Data with DSNUTILB



- After the successful execution, the content will be copied in the database table.

The screenshot displays the Microsoft SQL Server Enterprise Manager interface. The left pane shows the Object Explorer with the 'BankDemo' database selected. The right pane shows the SQL Query window with the following query:

```
/****** Script for SelectTopNRows comma
SELECT TOP (1000) [ZipCode]
      , [City]
      , [Country]
FROM [BankDemo].[dbo].[ZipCopy]
```

The Results pane shows the execution results of the query, displaying 3 rows of data:

	ZipCode	City	Country
1	1000	BRUSSELS	BE
2	2000	ANTWERP	BE
3	3000	GHEENT	BE

The status bar at the bottom indicates the query was executed successfully, returning 3 rows.

Unloading Data with DSNUTILB



- Illustrate the use of DSNUTILB for unloading data in a JCL context. This demo copies the contents of a database table into a sequential file with the DSNUTILB utility.
- Step 1: Open the solution from the folder Batch - Unload
- Step 2: Press Start Raincode Debugger to compile and run the project

```

//**
//*****
//JCL001 JOB CLASS=A,MSGCLASS=C
//*****
//** CLEAN UP FILES FROM PREVIOUS RUN
//*****
//STEP001 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE UNLOAD.ZIP.-|
SET MAXCC = 0
/*
//*****
//** SELECTIVE UNLOAD OF ZIPCODE TABLE
//*****
//STEP002 EXEC PGM=DSNUTILB
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//UTPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSIN DD *
UNLOAD TABLESPACE PRIMARY FROM TABLE DBO.ZIPCODE LIMIT 25
WHEN (ZIPCODE > 2000 AND ZIPCODE < 3000)
//SYSREC DD DSN=UNLOAD.ZIP.SEQ,
// DISP=(NEW,CATLG,DELETE),LRECL=68
//*****

```

Output

```

'Submit.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\8.0.11\System.Security.Principal.Windows.dll'. Skipped loading symbols. Module is optimizer
[2024-11-18 11:10:11.211] [1] [INFO] [RaincodeBatch]: 308000000027 : SYSLOG[C:\ProgramData\Raincode\Batch\SYSOUT\308000000027\SYSLOG.txt]
[2024-11-18 11:10:11.252] [1] [INFO] [RaincodeBatch]: 308000000027 : Job finished with status: NORMAL - CODE(0) RC(0)
[2024-11-18 11:10:11.259] [1] [INFO] [RaincodeBatch]: 308000000027 : LockingService Shutdown()
[2024-11-18 11:10:11.370] [1] [TRACE] [RaincodeBatch]: 308000000027 : Stopping LockingServer.Binding
[2024-11-18 11:10:11.383] [1] [TRACE] [RaincodeBatch]: 308000000027 : Stopping LockingServer.Listener
The thread 5204 has exited with code 0 (0x0).
[2024-11-18 11:10:11.406] [1] [TRACE] [RaincodeBatch]: 308000000027 : Stopped LockingServer.Listener
The program '[10608] Submit.exe' has exited with code 0 (0x0).

```

Raincode Batch – File types

Indexed Sequential File – Introduction



- Demo of Indexed Sequential File is available in the folder Batch - Indexed File
- This demo demonstrates two aspects of the indexed files:
 - JCL001.jcl: This demo creates two indexed sequential files created at **C:\Program Data\Raincode\Batch\DefaultVolume** and can also be viewed in the [Catalog Explorer](#).
 - JCL002.jcl: reads two indexed files (one sequentially and one using the record key) to create a report (a sequential file).

Indexed Sequential File – How to Demonstrate – JCL001.jcl



- Step 1: Open the solution from the folder Batch – Indexed File
- Step 2: Click on Start Raincode Debugger to execute the project
- Step 3: To view the result, click on the Catalog Explorer shortcut placed on the desktop
- Step 4: To view the file content, right-click on the file name and select “Edit data file”

File Edit Help						
+ [Icons] 50 [Checkmark] [Dropdown] Slow Catalog Access						
All						
▼ [Icon] DEFAULT (7 not refreshed) < >						
[Icon] ACCOUNT.DATA.ACC	File	FB	36	07-41-2022	78 KB	
[Icon] CUSTOMER.DATA.KSDS	File	FB	128	07-41-2022	15 KB	
[Icon] OUTPUT.DATA.REPORT	File	FB	161	07-44-2022	493 KB	
[Icon] OUTPUT.DATA.REPORTVS	File	FB	161	07-20-2022	493 KB	
[Icon] SYS1.PROCLIB	PDS	LSEQ	80	05-00-2022		

CUSTOMER.DATA.KSDS and ACCOUNT.DATA.ACC are the created indexed files

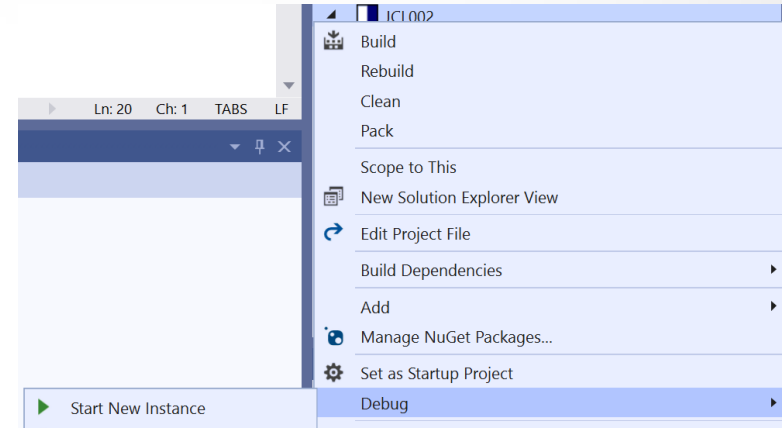
Indexed Sequential File – How to Demonstrate – JCL002.jcl



- Step 1: Open the solution from the folder Batch – Indexed File
Note: JCL001.jcl should have been executed before

- Step 2: Right click on the "JCL002" project and select "Debug - Start a new instance".

- Step 3: To view the result, click on the Catalog Explorer shortcut placed on the desktop



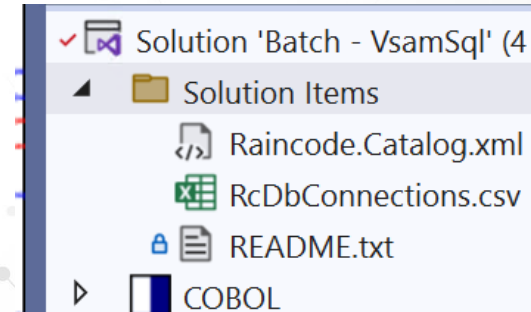
File Edit Help						
+ [Icons] 50 [Checkmark] [X] Slow Catalog Access						
All						
▼ [Icon] DEFAULT (7, not refreshed) < >						
[Icon]	ACCOUNT.DATA.ACC	File	FB	36	07-41-2022	78 KB
[Icon]	CUSTOMER.DATA.KSDS	File	FB	128	07-41-2022	15 KB
[Icon]	OUTPUT.DATA.REPORT	File	FB	161	07-44-2022	493 KB
[Icon]	OUTPUT.DATA.REPORT.VS	File	FB	161	07-20-2022	493 KB
[Icon]	SYS1.PROCLIB	PDS	LSEQ	80	05-00-2022	

OUTPUT.DATA.REPORT is the created indexed file

VSAMSql Indexed Sequential File – Introduction



- Demo of VSAMSql Indexed Sequential File is available in the folder Batch – VsamSql
- VSAMSql is a file driver that stores data in an SQL database instead of saving them on a disk (refer to [Catalog Configuration](#)).
- Raincode.Config.xml contains the configuration of the Catalog
 - catalogFormat="db" means that catalog is stored in the DB
 - catalogDbConnection="VSAMSQL" gives the connection to the DB. VSAMSQL is the name of the plan that can be found in the RcDbConnections.csv (refer to the [Map Plan Name to Actual Connection String](#))
 - The mappings between DSN and table names are given in the <datasetSqlMapping> element (refer to [Dataset mapping with tables](#))



VSAMSql Indexed Sequential File – Introduction



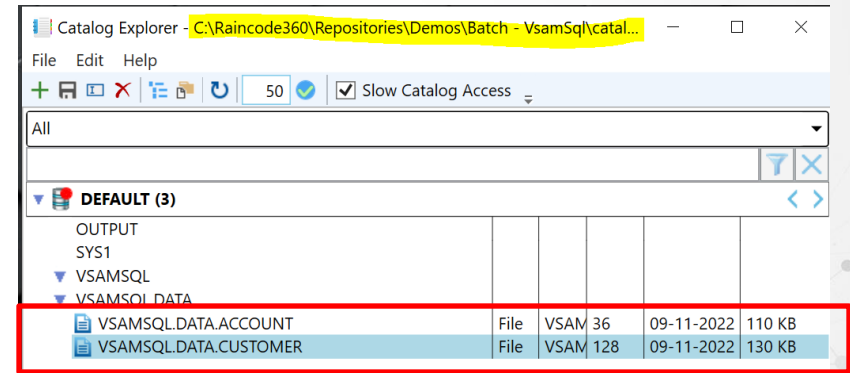
- This demo demonstrates five aspects of indexed files:
 - JCL001.jcl: This demo creates two VSAMSql files by reading data from a database
 - JCL002.jcl: Reads two VSAMSql indexed files (one sequentially and one using the record key) to create a report (a sequential file)
 - JCL003.jcl: This demo migrates a VsamSql indexed file to a flat file on disk and back
 - 'generate tables': This demo generates the creation script for tables used to store the catalog and the data
 - 'generate view': This demo generates the views and triggers the creation script for the two files

VSAMSql Indexed Sequential File – JCL001.jcl



This project is the same as the “Batch-Indexed File” demo, except that the output files are VSAMSql files instead of VSAM files (on disk).

- Step 1: Open the solution from the folder Batch – VsamSql
- Step 2: Click on Start Raincode Debugger to execute the project to execute JCL001
- Step 3: To view the result, click on the Catalog Explorer shortcut placed on the desktop
 - Because this project doesn’t use the default catalog, you should change the configuration (**File – Change configuration**) to `C:\Raincode360\Repositories\Demos\Batch - VsamSql\catalog\Raincode.Catalog.xml`



The screenshot shows the Catalog Explorer window with the following structure:

- File Edit Help
- 50 Slow Catalog Access
- All
- DEFAULT (3)
 - OUTPUT
 - SYS1
 - VSAMSQL
 - VSAMSQL DATA
 - VSAMSQL.DATA.ACCOUNT File VSAM 36 09-11-2022 110 KB
 - VSAMSQL.DATA.CUSTOMER File VSAM 128 09-11-2022 130 KB

The last two rows are highlighted with a red border.

VSAMSQL.DATA.ACCOUNT and VSAMSQL.DATA.CUSTOMER are the created VSAM indexed files

The data is stored in a database.

VSAMSql Indexed Sequential File – Raincode Vsam Sql Console



VSAMSQL.DATA.ACCOUNT and VSAMSQL.DATA.CUSTOMER are available under the mappings tab of the Raincode VSAMSql console. The Raincode VSAMSql console allows you to Truncate data or delete the files.

The screenshot shows the Raincode Vsam Sql console interface. At the top, there's a header with the Raincode logo, a green dot, 'Vsam Sql', and a 'Mappings' tab. A 'CHANGE CONNECTION' button is on the right. Below the header is a section titled 'Dataset Mappings'. It contains a table with three columns: 'Dataset Name', 'Table Name', and 'Record Count'. There are two rows of data. The first row has 'Dataset Name' as ':DEFAULT:VSAMSQL.DATA.CUSTOMER', 'Table Name' as 'REC_CUSTOMER', and 'Record Count' as '1047'. The second row has 'Dataset Name' as ':DEFAULT:VSAMSQL.DATA.ACCOUNT', 'Table Name' as 'REC_ACCOUNT', and 'Record Count' as '3141'. To the right of each row are two buttons: 'TRUNCATE' and 'DELETE'.

Dataset Name	Table Name	Record Count	
:DEFAULT:VSAMSQL.DATA.CUSTOMER	REC_CUSTOMER	1047	TRUNCATE DELETE
:DEFAULT:VSAMSQL.DATA.ACCOUNT	REC_ACCOUNT	3141	TRUNCATE DELETE

Click on one of the file name to display the data

Dataset - :DEFAULT:VSAMSQL.DATA.CUSTOMER

[EXPORT TO EXCEL](#) [TRUNCATE](#)

The screenshot shows the data for the selected dataset ':DEFAULT:VSAMSQL.DATA.CUSTOMER'. It displays a table with five columns: 'W_CUSTOMERID', 'W_FIRSTNAME', 'W_LASTNAME', 'W_ZIPCODE', and 'W_CITY'. The table contains 16 rows of data. At the bottom, there's a pagination bar showing '1 - 100 of 1047 items' and 'Items per page'.

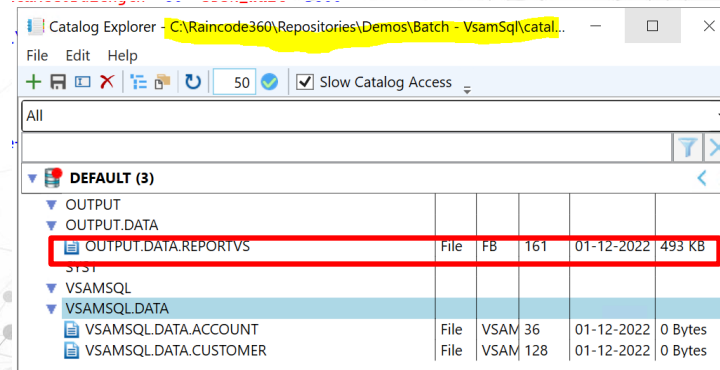
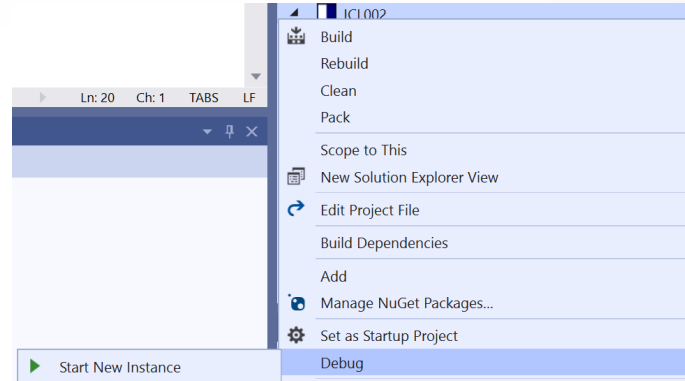
W_CUSTOMERID	W_FIRSTNAME	W_LASTNAME	W_ZIPCODE	W_CITY
5	Jean	Ward	1000	BRUSSEL
17	Pete	York	1000	BRUSSEL
29	Tracey	George	1000	BRUSSEL
41	Alexis	White	1000	BRUSSEL
53	Britt	Gordon	1000	BRUSSEL
65	Ron	Rowland	1000	BRUSSEL
77	Roosevelt	Grimes	1000	BRUSSEL
89	Tammy	Chung	1000	BRUSSEL
101	Emma	Ritter	1000	BRUSSEL
113	Aurelia	Wheeler	1000	BRUSSEL
125	Bianca	Key	1000	BRUSSEL
137	Cindy	Haney	1000	BRUSSEL
149	Cristina	Shah	1000	BRUSSEL
161	Wilma	Cummins	1000	BRUSSEL

VSAMSql Indexed Sequential File – JCL002.jcl



JCL001.jcl should have been executed before

- Step 1: Right-click on the "JCL002" project and select "Debug - Start a new instance".
- Step 2: To view the result, click on the Catalog Explorer shortcut placed on the desktop



OUTPUT.DATA.REPORTVS is the created file

VSAMSql Indexed Sequential File – JCL003.jcl



JCL001.jcl should have been executed before

- This project migrates the VSAMSql indexed file
VSAMSQL.DATA.ACCOUNT
to the flat file on disk OUTPUT.DATA.ACCOUNT
and back to VSAMSQL.DATA3.ACCOUNT
- Step 1: Right-click on the "JCL003" project and select "Debug - Start a new instance"
- Step 2: To view the result, click on the Catalog Explorer shortcut placed on the desktop ([change configuration as in JCL001](#))

Catalog Explorer - C:\Raincode360\Repositories\Demos\Batch - VsamSql\catalog\Rainco...

File Edit Help

+ [Icons] 50 [Checkmark] [Dropdown] Slow Catalog Access

All

▼ DEFAULT (3)						
▼ OUTPUT						
▼ OUTPUT.DATA						
OUTPUT.DATA.ACCOUNT	File	FB	80	09-11-2022	0 Bytes	
OUTPUT.DATA.REPORTVS	File	Unkn	-	09-11-2022	0 Bytes	
SYS1						
▼ VSAMSQL						
▼ VSAMSQL.DATA						
VSAMSQL.DATA.ACCOUNT	File	VSAM	36	09-11-2022	0 Bytes	
VSAMSQL.DATA.CUSTOMER	File	VSAM	128	09-11-2022	0 Bytes	
▼ VSAMSQL.DATA3						
VSAMSQL.DATA3.ACCOUNT	File	VSAM	36	09-11-2022	0 Bytes	

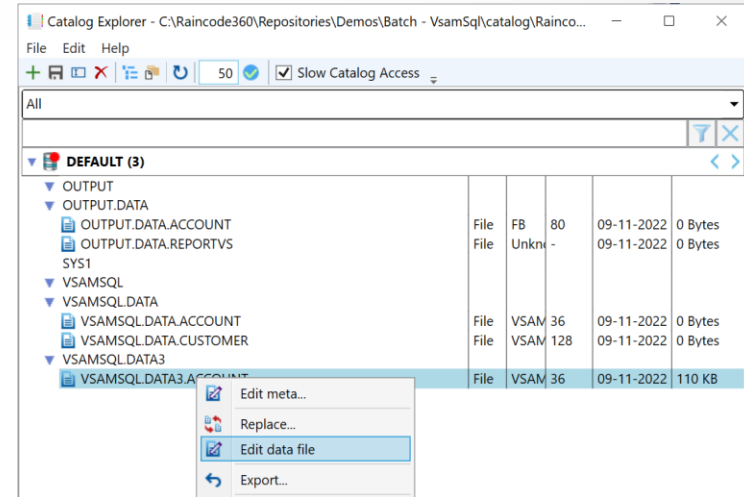
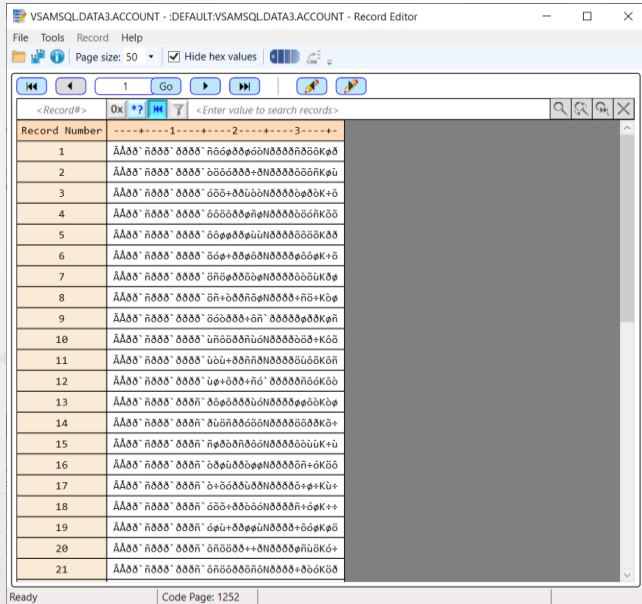
OUTPUT.DATA.ACCOUNT is a flat file

VSAMSQL.DATA3.ACCOUNT is a VSAM indexed files

VSAMSql Indexed Sequential File – JCL003.jcl – Contd..



- Step 4: To view the file content, right-click on the file name in the Catalog Explorer and select “Edit data file”.
- The file will get open in the Record Editor.



VSAMSql Indexed Sequential File – generate tables



- Execute 'generate_tables.bat'
- This step reads the catalog definition (Catalog\Raincode.Catalog.xml) and generate the tables creation script (table_creation.sql)
 - CATALOG_META: is the table to store the catalog information
 - VSAM_META: is the table to store the DSN for which the data are in the database
 - REC_ACCOUNT and REC_CUSTOMER: the tables to store the data
- The tables are already created. You didn't need to execute the Sql script

VSAMSql Indexed Sequential File – generate views

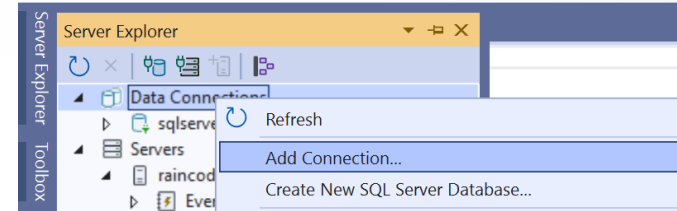
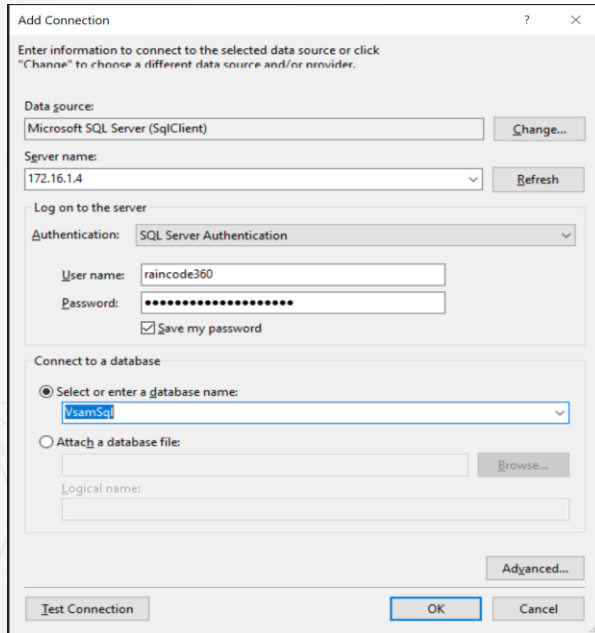


- Execute 'generate_views.bat'
- For each copybook (COBOL\ACCOUNT.cpy, COBOL\CUSTOMER.cpy) this script:
 - Use the COBOL compiler (cobrc) to convert the .cpy into an .xml file containing the declaration of the variables
 - The CopybookViewGenerator use the .xml file and connects to the DB to generate the views and the triggers creation scripts
- The views and triggers are already created. You didn't need to execute the SQL script.
- For each copybook
 - A view is created that contains one column for each copybook's variable with a picture clause and all the columns of the table
 - A "trigger instead of update" to update the data using the view
 - A "trigger instead of insert" to insert data using the view
 - A "trigger instead of delete" to delete data using the view

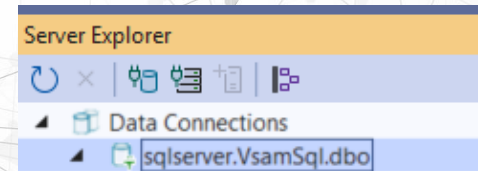
VSAMSql Indexed Sequential File – How to connect the database



- Step 1: Right-click on the *Data Connections* in the Server Explorer of the Visual Studio and select *Add Connection...*



- Step 2: Select Microsoft SQL Server and Add Connection dialog box will appear. Fill the dialog box as follows:
 - Server Name – the value of the environment variable `DB_SERVER`.
 - Authentication- *SQL Server Authentication*
 - Username: *raincode360*
 - Password: the value of the environment variable `DB_PASSWORD`
 - Connect to a database – Choose VsamSql from the list.



- Step 3: Click OK, and the database will appear under *Data connections* in the Server Explorer



- Right-click on the newly created data connection and select *New Query*
- The database contains two kinds of information
 - The catalog
 - The data themselves
- The catalog meta-information are stored in the CATALOG_META table.
Execute the following query to see the contains of the catalog

```
select * from CATALOG_META
```

ID	TYPE	PARENT_NAME	LEVEL	CATALOG_KEY	FULL_NAME	CONFIG
1	1015	P		:DEFAULT:SYS1	:DEFAULT:SYS1.PROCLIB	<dataSet version="4" name="SYS1.PROCLIB" vo...
2	1025	D		:#SYSOUT:JOB00000000011-JCL001.STEP002_3	:#SYSOUT:JOB00000000011-JCL001.STEP002_3.SYSOUT	<dataSet version="4" name="JOB00000000011-JCL...
3	1029	D		:DEFAULT:OUTPUT.DATA	:DEFAULT:OUTPUT.DATA.ACCOUNT	<dataSet version="4" name="OUTPUT.DATA.ACCO...
4	1031	D		:DEFAULT:VSAMSQL.DATA	:DEFAULT:VSAMSQL.DATA.CUSTOMER	<dataSet version="4" name="VSAMSQL.DATA.CUS...
5	1032	D		:DEFAULT:VSAMSQL.DATA	:DEFAULT:VSAMSQL.DATA.ACCOUNT	<dataSet version="4" name="VSAMSQL.DATA.ACC...

the CONFIG column contains the meta information about the DSN

VSAMSql Indexed Sequential File – How to connect the database



- The table VSAM_META contains the list of files whom data are stored in the DB.

Execute the following query to see the data

```
select * from VSAM_META
```

It contains two lines (one for each file) and each file is associated with one table

	ID	FILE_NAME	TABLE_NAME
1	2	SYSDA:DEFAULT:VSAMSQL.DATA.CUSTOMER	REC_CUSTOMER
2	3	SYSDA:DEFAULT:VSAMSQL.DATA.ACCOUNT	REC_ACCOUNT

- The data in the tables are difficult to read because they are in the binary format (VARBINARY)

```
select top(10) * from REC_ACCOUNT
```

	RID	FILE	data	KEY
1	1	3	0xC2C5F0F060F1F0F0F060F0F0F060F1F4F3F8F0F0F8F3...	0xC2C5F0F060F1F0F0F060F0F0F060F1F4F3F8
2	2	3	0xC2C5F0F060F1F0F0F060F0F0F060F2F6F4F3F0F0F0F7...	0xC2C5F0F060F1F0F0F060F0F0F060F2F6F4F3

- For each table, there is a view that shows the data in a more readable format

```
select top(10) * from REC_ACCOUNT_V
```

	RID	FILE	data	KEY	W_ACCOUNT_ID	W_CUSTOMERID_ID	W_AMOUNT
1	1	3	0xC2C5F0F060F1F0F0F060F0F0F060F1F4F3F8F0F0F8F3...	0xC2C5F0F060F1F0F0F060F0F0F060F1F4F3F8	BE00-1000-0000-1438	832	+00001064.80
2	2	3	0xC2C5F0F060F1F0F0F060F0F0F060F2F6F4F3F0F0F0F7...	0xC2C5F0F060F1F0F0F060F0F0F060F2F6F4F3	BE00-1000-0000-2643	70	+00004541.89

VSAMSql Indexed Sequential File – How to connect the database



- Triggers have been defined to modify (update, insert, delete) the data using the view
- To update the 'W_AMOUT' of account id 'BE00-1000-0000-1438'
`update REC_CUSTOMER_V set W_FIRSTNAME = 'Jean' where W_CUSTOMERID = 5;`
- Check the data have been update in the DB
`select * from REC_CUSTOMER_V where W_CUSTOMERID = 5;`

	RID	FILE	data	KEY	W_CUSTOMERID	W_FIRSTNAME	W_LASTNAME	W_ZIPCODE	W_CITY
1	15237	3008	0xF0F0F0F5D1858195404040404040404040404040...	0xF0F0F0F5	5	Jean	Ward	1000	BRUSSEL

- The record can also be used to verify the contains of the dataset (VSAMSQL.DATA.ACCOUNT)

VSAMSQL.DATA.CUSTOMER - DEFAULT-VSAMSQL.DATA.CUSTOMER - Record Editor

File Tools Record Help

Page size: 50 Hide hex values

Record Number

1	0001Kirk	Greer	2000ANTWERPEN
2	0002Owen	Le	9000GENT
3	0003Jospeh	Beard	6000CHARLEROI
4	0004Dale	Huff	4000Glain
5	0005Jean	Ward	1000BRUSSEL
6	0006Tuan	Hicks	5000Beez
7	0007Carlton	Spence	3000LEUVEN

Relative File – Introduction



- Demo of Relative File is available in the folder Batch - Relative File
- This demo creates a Relative File created at **C:\Program Data\Raincode\Batch\DefaultVolume** and can also be viewed in the [Catalog Explorer](#).

Relative File – How to Demonstrate



- Step 1: Open the solution from the folder Batch – Relative File
- Step 2: Click on Start Raincode Debugger to execute the project
- Step 3: To view the result, click on the Catalog Explorer shortcut placed on the desktop

CUSTOMER.DATA.RRDS is the created relative file

Catalog Explorer - C:\ProgramData\Raincode\Batch\Raincode.Catalog.xml

File Edit Help

+ [Icons] 50 [Slow Catalog Access]

All

▼ DEFAULT (10)					
▼ ACCOUNT					
▼ ACCOUNT.DATA					
ACCOUNT.DATA.ACC	File	VSAM	36	30-06-2022	0 Bytes
▼ CUSTOMER					
▼ CUSTOMER.DATA					
CUSTOMER.DATA.GDG	GDG	FB	-	30-06-2022	
CUSTOMER.DATA.KSDS	File	VSAM	128	30-06-2022	0 Bytes
CUSTOMER.DATA.RRDS	File	RR FB	128	30-06-2022	134 KB
▼ EBCDIC					
EBCDIC.EXTRACT	File	FB	200	30-06-2022	0 Bytes
EBCDIC.SORTED	File	FB	34	30-06-2022	0 Bytes
▼ INLINE					
▼ INLINE.QUERY					
INLINE.QUERY.LSEQ	File	LSEQ	57	30-06-2022	0 Bytes
INLINE.QUERY.SEQ	File	FB	57	30-06-2022	0 Bytes
▼ JCL003A					
JCL003A.FILE01	File	LSEQ	200	30-06-2022	0 Bytes
JCL003A.FILE02	File	LSEQ	34	30-06-2022	0 Bytes
JCL003A.FILE03	File	FB	34	30-06-2022	0 Bytes

Variable Length File – Introduction



- Demo of Variable Length File is available in the folder Batch – Variable Length File
- This demo creates a Variable Length File created at **C:\Program Data\Raincode\Batch\DefaultVolume** and can also be viewed in the [Catalog Explorer](#).

Variable Length File – How to Demonstrate



- Step 1: Open the solution from the folder Batch – Variable Length File

- Step 2: Click on Start Raincode Debugger to execute the project

- Step 3: To view the result, click, on the Catalog Explorer shortcut placed on the desktop

CUSTOMER.DATA.VB is the created variable length file

Catalog Explorer - C:\ProgramData\Raincode\Batch\Raincode.Catalog.xml

File Edit Help

+ [Icons] 50 [Checkmark] [Slow Catalog Access]

All

▼	DEFAULT (10)					
▼	ACCOUNT					
	ACCOUNT.DATA					
▼	CUSTOMER					
▼	CUSTOMER.DATA					
▶	CUSTOMER.DATA.GDG	GDG	FB	-	30-06-2022	
	CUSTOMER.DATA.KSDS	File	VSAM	128	30-06-2022	0 Bytes
	CUSTOMER.DATA.RRDS	File	RR FB	128	30-06-2022	0 Bytes
	CUSTOMER.DATA.VB	File	VB	132	30-06-2022	23 KB
▼	EBCDIC					
	EBCDIC.EXTRACT	File	FB	200	30-06-2022	0 Bytes
	EBCDIC.SORTED	File	FB	34	30-06-2022	0 Bytes
▼	INLINE					
	INLINE.QUERY					
▼	JCL003A					
	JCL003A.FILE01	File	LSEQ	200	30-06-2022	0 Bytes

Generation Data Group

```

***** Top of Data *****
Menu Options View Utilities Compilers Help
-----
DSLIST - Data Sets on volume *                               Row 1 of 4
Command ==>  Scroll ==> CSR_

Command - Enter "/" to select action                        Dsorg  Recfm  Lrec1  Blksz
-----
      RAINCODE.PROJECT1.GDGROUP1.G0001V00                PS    FB          80  27920
      RAINCODE.PROJECT1.GDGROUP1.G0002V00                PS    FB          80  27920

```

Generation Data Group – Introduction



- A Generation Data Group, also known as GDG, is versioning of data sets that are successive generations of historically related data
- This Batch demo creates multiple generations of a GDG
- Created GDG and its version will be available at C:\Program Data\Raincode\DefaultVolume.

Generation Data Group – How to demonstrate



- Step 1: Open the solution from the folder Batch - GDG Processing

- Step 2: Click on Start Raincode Debugger to execute the solution

- The JCL starts by deleting the output from the previous run (if any)

```
//JCL001 JOB CLASS=A,MSGCLASS=C
//***** DELETE GDG DEFINITION AND PREVIOUSLY CREATED FILES *****/
//*****
//STEP001 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE (CUSTOMER.DATA.GDG) GDG FORCE
SET MAXCC = 0
/*
//*****
//* DEFINE GDG WITH 3 GENERATIONS
//*****
//STEP002 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DEFINE GDG(NAME(CUSTOMER.DATA.GDG) -
LIMIT(3) -
NOEMPTY -
SCRATCH)
/*
//*****
//* CREATE GENERATION 1
//*****
//STEP003 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSUT1 DD *
GENERATION 1
/*
//SYSUT2 DD DSN=CUSTOMER.DATA.GDG(+1),
// DISP=(NEW,CATLG,DELETE)
//*****
//* CREATE GENERATION 2
//*****
//STEP004 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
```

- Created GDG and its version can also be seen using [Catalog Explorer](#)

Partitioned Dataset

Partitioned data set - Introduction



- A Partitioned Data Set, also known as **PDS**, is a Data Set that contains directory and members.
- Illustrated features:
 - Creates a PDS (pdslib) via JCL
 - Usage of IEBFR14 to create a file
 - Usage of IEBFR14 with the PATH parameter

Partitioned Dataset– How to demonstrate



- Step 1: Open the solution from the folder Batch – Partitioned Dataset.
- Step 2: Click on Start Raincode Debugger to execute P001 project.
 - It will run JCL001, which creates the pdslib PROCLIB.TEST.JCL with a single member named MYPROC
 - MYPROC is a JCL procedure that can be invoked from other JCLs with parameters.
- Step 3: Right-click on P002 and select Debug->Start New instance to execute P002 project.
 - It will run JCL002, which has the statement “LIB001 JCLLIB ORDER=PROCLIB.TEST.JCL”, that indicates where the procedure can be found.

Default location for JCL procedures is SYS1.PROCLIB

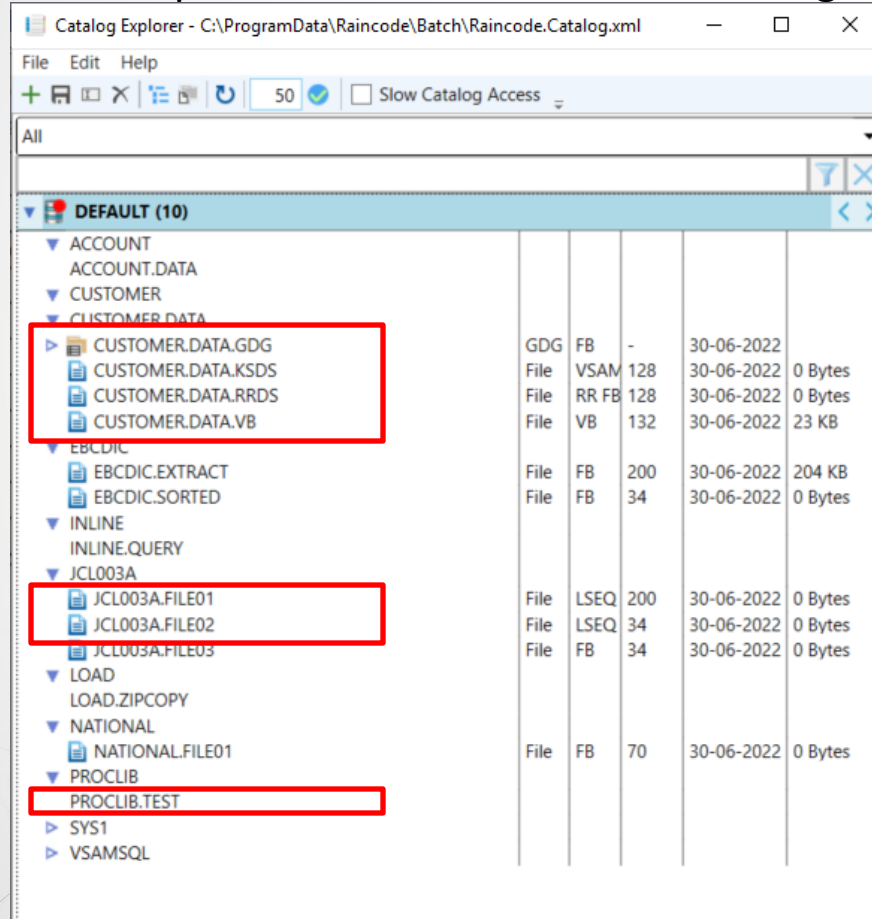
Created PDS can also be seen using [Catalog Explorer](#)

Catalog Explorer

Catalog Explorer



- A Graphical User Interface for accessing the (virtual) mainframe file system.



The Raincode Catalog Explorer allows you to manage mainframe files, PDS (Partitioned Data Set) and GDG (Generation Data Group), and more.

When Catalog Explorer starts, it populates all the data in the DSN view available at
C:\ProgramData\Raincode\Batch\DefaultVolume

CUSTOMER.DATA.GDG and its generation were created when [GDGProcessing.sln](#) was compiled and executed.

JCL003A.FILE01 and JCL003A.FILE02 are output files that were created when [Batch - Legacy with Database Access.sln](#) was compiled and executed.

PROCLIB.TEST is created when [Batch - Partitioned Dataset.sln](#) was compiled and executed.

Catalog Explorer – Features



- Files
 - Adding, replacing, exporting, deleting, renaming
 - Editing ([Record Editor](#))
- Partitioned data sets (PDS)
 - Adding a new PDS, deleting a PDS
 - Adding members to a PDS
 - Deleting members from a PDS
- Generation data group (GDG)
 - Adding a new GDG, Exporting, Deleting
 - Adding generation to an existing GDG
 - GDG (+1), GDG (-1)
- Import/Export

Catalog Explorer – How to create datasets



- Step 1: Launch the Catalog Explorer through e.g. the desktop shortcut
- Step 2: From the toolbar, select File → New or press + , and add the datasets (File, PDS, or GDG).
- For this demonstration, we will use GDG as an example
- Add the Name of Dataset
- Select Type of Dataset from the drop-down menu
- Press Ok

The screenshot shows a 'New' dialog box with the following fields and options:

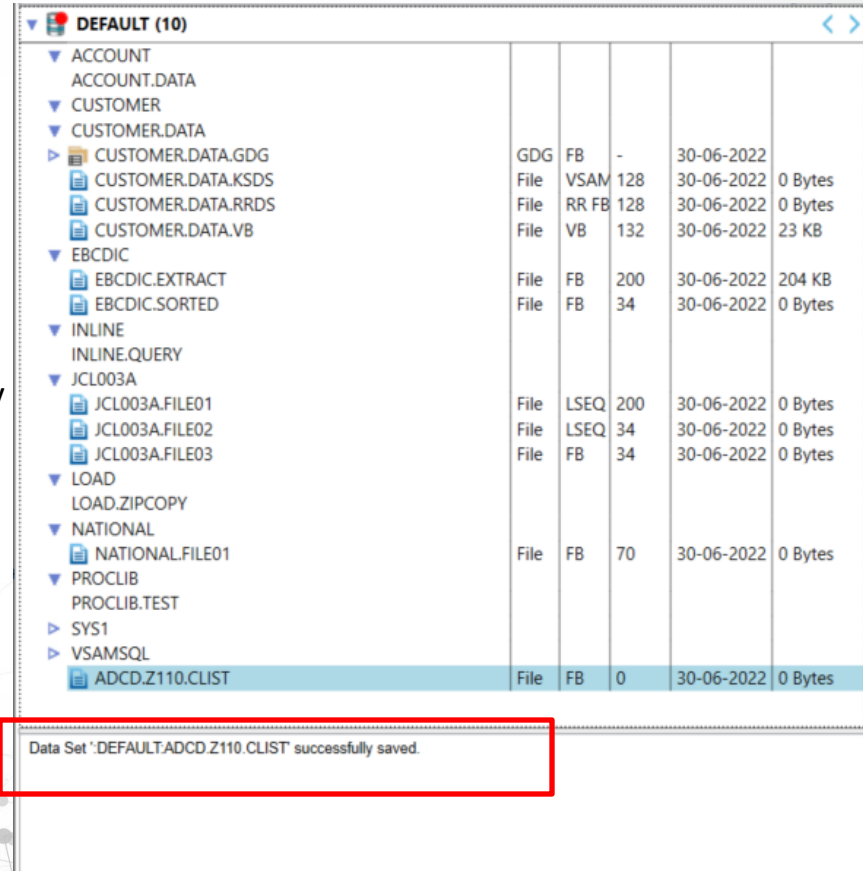
- Volume: DEFAULT
- Name: ADCD.Z110.CLIST
- Unit: (empty)
- Type: GDG (selected from a dropdown menu that also lists File, PDS, and GDG)
- Buttons: OK, Cancel

Catalog Explorer – How to save created datasets



- Step 3: Press Save. The dataset will get created in the DefaultVolume folder and will be visible in the Catalog Explorer.

If you select Edit → Activity Log → Info, then Activity Log shows all the actions taken by the user, such as creating datasets.



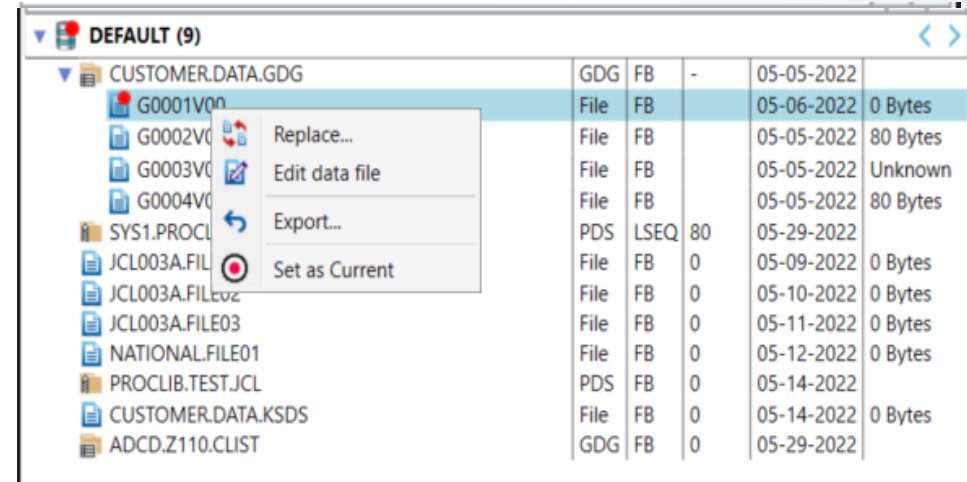
DEFAULT (10)					
ACCOUNT					
ACCOUNT.DATA					
CUSTOMER					
CUSTOMER.DATA					
CUSTOMER.DATA.GDG	GDG	FB	-	30-06-2022	
CUSTOMER.DATA.KSDS	File	VSAM	128	30-06-2022	0 Bytes
CUSTOMER.DATA.RRDS	File	RR FB	128	30-06-2022	0 Bytes
CUSTOMER.DATA.VB	File	VB	132	30-06-2022	23 KB
EBCDIC					
EBCDIC.EXTRACT	File	FB	200	30-06-2022	204 KB
EBCDIC.SORTED	File	FB	34	30-06-2022	0 Bytes
INLINE					
INLINE.QUERY					
JCL003A					
JCL003A.FILE01	File	LSEQ	200	30-06-2022	0 Bytes
JCL003A.FILE02	File	LSEQ	34	30-06-2022	0 Bytes
JCL003A.FILE03	File	FB	34	30-06-2022	0 Bytes
LOAD					
LOAD.ZIPCOPY					
NATIONAL					
NATIONAL.FILE01	File	FB	70	30-06-2022	0 Bytes
PROCLIB					
PROCLIB.TEST					
SYS1					
VSAMSQL					
ADCD.Z110.CLIST	File	FB	0	30-06-2022	0 Bytes

Data Set 'DEFAULT.ADCD.Z110.CLIST' successfully saved.

Catalog Explorer – How to add generation in the existing GDG



- Right click on the existing GDG, context menu will appear, select Add Generation
- It will add generation
- The latest added generation is always set as current by default
- It can be changed by right click on the selected generation and selecting 'set as current' from the context menu
- Previously G0004V00 was set as current, and now it's G0001V00 in CUSTOMER.DATA.GDG dataset



DEFAULT (9)					
CUSTOMER.DATA.GDG	GDG	FB	-	05-05-2022	
G0001V00	File	FB		05-06-2022	0 Bytes
G0002V00	File	FB		05-05-2022	80 Bytes
G0003V00	File	FB		05-05-2022	Unknown
G0004V00	File	FB		05-05-2022	80 Bytes
SYS1.PROCLIB	PDS	LSEQ	80	05-29-2022	
JCL003A.FILE01	File	FB	0	05-09-2022	0 Bytes
JCL003A.FILE02	File	FB	0	05-10-2022	0 Bytes
JCL003A.FILE03	File	FB	0	05-11-2022	0 Bytes
NATIONAL.FILE01	File	FB	0	05-12-2022	0 Bytes
PROCLIB.TEST.JCL	PDS	FB	0	05-14-2022	
CUSTOMER.DATA.KSDS	File	FB	0	05-14-2022	0 Bytes
ADCD.Z110.CLIST	GDG	FB	0	05-29-2022	

Record Editor

Record Editor – Introduction

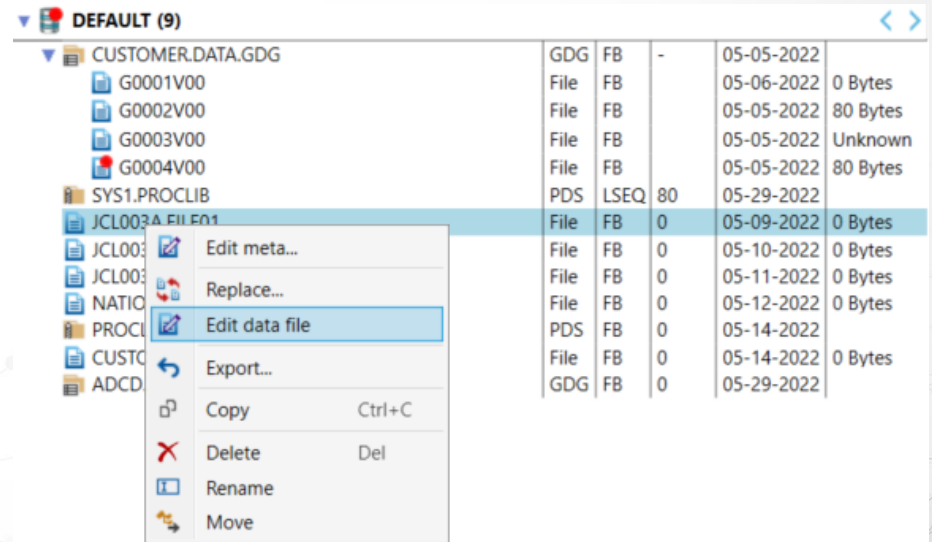


- Record Editor is a GUI-based tool that allows users to view and edit file contents and view and modify certain metadata.
- Record Editor deals with record-based files. This tool aims to let the user work with data files as a sequence of records to provide Mainframe-like functionality.
- Record Editor provides the following features:
 - Display data file contents as records
 - Edit records in binary format (hexadecimal)
 - Display records in terms of fields
 - Edit records field-wise according to declaration description
 - Search records as per encoding
 - Search within fields
 - Display certain metadata
 - Modify record length and the encoding

Record Editor – How to launch



- Raincode Editor can be launched in two ways:
 - By double-clicking on the “Record Editor” shortcut placed at the desktop
 - From the Catalog Explorer:
 - Select the file and right-click, the context menu will appear
 - Select Edit data file to launch Record Editor



Record Editor – How to load data file



- Step 1: Select Tool->Open from the toolbar, a dialog box will appear.
- Step 2: Select the appropriate metadata file. Record Editor will load the corresponding data file.

Record Number	Record ID	Record Data	Record Metadata
1	000000000	abcdefghijklmnopqrstuvwxyz	ABCDEFGHIJKLMNOPQRSTUVWXYZ `-[\\';',./~!@#%&*()_+{} :"<>?
2	000000001	abcdefghijklmnopqrstuvwxyz	ABCDEFGHIJKLMNOPQRSTUVWXYZ `-[\\';',./~!@#%&*()_+{} :"<>?
3	000000002	abcdefghijklmnopqrstuvwxyz	ABCDEFGHIJKLMNOPQRSTUVWXYZ `-[\\';',./~!@#%&*()_+{} :"<>?
4	000000003	abcdefghijklmnopqrstuvwxyz	ABCDEFGHIJKLMNOPQRSTUVWXYZ `-[\\';',./~!@#%&*()_+{} :"<>?
5	000000004	abcdefghijklmnopqrstuvwxyz	ABCDEFGHIJKLMNOPQRSTUVWXYZ `-[\\';',./~!@#%&*()_+{} :"<>?
6	000000005	abcdefghijklmnopqrstuvwxyz	ABCDEFGHIJKLMNOPQRSTUVWXYZ `-[\\';',./~!@#%&*()_+{} :"<>?
7	000000006	abcdefghijklmnopqrstuvwxyz	ABCDEFGHIJKLMNOPQRSTUVWXYZ `-[\\';',./~!@#%&*()_+{} :"<>?
8	000000007	abcdefghijklmnopqrstuvwxyz	ABCDEFGHIJKLMNOPQRSTUVWXYZ `-[\\';',./~!@#%&*()_+{} :"<>?
9	000000008	abcdefghijklmnopqrstuvwxyz	ABCDEFGHIJKLMNOPQRSTUVWXYZ `-[\\';',./~!@#%&*()_+{} :"<>?
10	000000009	abcdefghijklmnopqrstuvwxyz	ABCDEFGHIJKLMNOPQRSTUVWXYZ `-[\\';',./~!@#%&*()_+{} :"<>?
11	000000010	abcdefghijklmnopqrstuvwxyz	ABCDEFGHIJKLMNOPQRSTUVWXYZ `-[\\';',./~!@#%&*()_+{} :"<>?
12	000000011	abcdefghijklmnopqrstuvwxyz	ABCDEFGHIJKLMNOPQRSTUVWXYZ `-[\\';',./~!@#%&*()_+{} :"<>?
13	000000012	abcdefghijklmnopqrstuvwxyz	ABCDEFGHIJKLMNOPQRSTUVWXYZ `-[\\';',./~!@#%&*()_+{} :"<>?
14	000000013	abcdefghijklmnopqrstuvwxyz	ABCDEFGHIJKLMNOPQRSTUVWXYZ `-[\\';',./~!@#%&*()_+{} :"<>?
15	000000014	abcdefghijklmnopqrstuvwxyz	ABCDEFGHIJKLMNOPQRSTUVWXYZ `-[\\';',./~!@#%&*()_+{} :"<>?
16	000000015	abcdefghijklmnopqrstuvwxyz	ABCDEFGHIJKLMNOPQRSTUVWXYZ `-[\\';',./~!@#%&*()_+{} :"<>?
17	000000016	abcdefghijklmnopqrstuvwxyz	ABCDEFGHIJKLMNOPQRSTUVWXYZ `-[\\';',./~!@#%&*()_+{} :"<>?
18	000000017	abcdefghijklmnopqrstuvwxyz	ABCDEFGHIJKLMNOPQRSTUVWXYZ `-[\\';',./~!@#%&*()_+{} :"<>?
19	000000018	abcdefghijklmnopqrstuvwxyz	ABCDEFGHIJKLMNOPQRSTUVWXYZ `-[\\';',./~!@#%&*()_+{} :"<>?
20	000000019	abcdefghijklmnopqrstuvwxyz	ABCDEFGHIJKLMNOPQRSTUVWXYZ `-[\\';',./~!@#%&*()_+{} :"<>?
21	000000020	abcdefghijklmnopqrstuvwxyz	ABCDEFGHIJKLMNOPQRSTUVWXYZ `-[\\';',./~!@#%&*()_+{} :"<>?

Refer to the [Record Editor](#) documentation for more details.

Automatic Restart

Automatic Restart – Concepts



- Mass update programs must perform periodic commits for various reasons.
 - To make sure that the transaction log of the database is not flooded.
 - To make sure that locks are periodically released.
 - To limit the amount of rollback in case of an error.
- Intermediate commits make it more difficult to restart a job after an abend.
- Automatic Restart optionally performs certain actions behind the scenes.
 - Taking periodic commits.
 - Saving the state of the program with each commit.
 - Repositioning cursors and file pointers at restart.

Automatic Restart – Configuration



- Automatic Restart is configured by means of XML stored in a configuration file.

```
<?xml version="1.0" encoding="utf-8"?>
<Configuration>
  <ProcessingOptions>
    <ProcessingOption AutomaticRestart="true"
      AutomaticCheckpoints="true"
      AutomaticCursorPositioning="true"
      PaceCheckpoints="false"
      TriggerCount="5" |
      RestoreWorkingStorage="PGMSTART"
      ProgName="*"
      JobName="*"
      StepName="*" />
    <Cursor Package="COB001" Name="CUR01" />
  </ProcessingOption>
</ProcessingOptions>
<CursorRepositioningOptions>
  <CursorRepositioningOption CursorPackage="COB001" CursorName="CUR01" RepositioningMethod="COLUMN">
    <Columns>
      <Column Name="AccountNumber" />
    </Columns>
  </CursorRepositioningOption>
</CursorRepositioningOptions>
</Configuration>
```

- Meaning of the parameters in this particular example:
 - The program COB001 is controlled by Automatic Restart.
 - Commits are taken every five updates
 - Working storage is restored at the restart
 - The main cursor CUR01 is repositioned at the restart

Automatic Restart – Demos



- There are three Demos, each accomplishing the same task with a different configuration.
 - Demo 1: Cursor based processing and automatic repositioning
 - Demo 2: Cursor based processing and Manual Cursor repositioning
 - Demo 3: File based processing
- In each demo, the main program is started and purposefully abended by forcing an error. After that, the main program is restarted, at which point it will run until successful completion.
- Each demo produces its own set of output files or 'snapshots'.
 - Snapshot 1: before the execution of the main program
 - Snapshot 2: after the main program has abended
 - Snapshot 3: after the restart and successful end of the main program

Automatic Restart – Script



- Open the appropriate demo in Visual Studio.
- Start the main script by typing 'run' at the Developer Command Prompt.
- The script starts by running the JCL002, which executes the program COB001.
- COB001 updates a total of 20 accounts in the database.
- Automatic Restart is configured to perform an automatic checkpoint after every 5 updates.
- An abend is forced just before the 4th commit (TRMBEFORCKP=3).
- JCL002 is submitted with the necessary restart parameters (-Restart and -RestartJobId).
- The restarted job runs until completion.

```
C:\WINDOWS\system32\cmd.exe - push

DEND : AUTOMATIC RESTART
- Cursor based processing
- Automatic checkpointing
- Automatic cursor repositioning

=====
JCL001: Initial setup
Starting init provider
Finished init provider
JCL001 JCL : Scanning C:\repositories\raincode-360\Batch - Restart - Cursor with automatic checkpointing and automatic repositioning\Batch
JCL001 JCL : Starting locking service18412
JCL001 JCL : Locking service active:18412
JCL001 JCL : Start execution Phase
JCL001 JCL : DataSetLock StepID:3 locking Add ARC01.OUTPUT : Exclusive
JCL001 JCL : Exclusive locking ARC01.OUTPUT StepID:3
JCL001 JCL : Start execution of steps
JCL001 JCL : STEP001/1 -> Start EXEC C:\Program Files\Raincode\Batch\IDCAMS.exe
JCL001 JCL : Starting retrieveConnectionString
JCL001 JCL : Finishing retrieveConnectionString with correct data
JCL001 JCL : Error deleting ARC01.*: entry not found
JCL001 JCL : Deleting Dataset SYSIN : C:\ProgramData\Raincode\Batch\DefaultVolume\SYSTEMP\JOB0000000003\STEP1\N637369611719980380.seq
JCL001 JCL : STEP001/1 -> NORMAL - RC(0)
JCL001 JCL : STEP002/2 -> Start EXEC C:\repositories\raincode-360\Batch - Restart - Cursor with automatic checkpointing and automatic repositioning\Batch\bin\Debug\DBSETUP.dll
JCL001 JCL : Starting retrieveConnectionString
JCL001 JCL : Finishing retrieveConnectionString with correct data
JCL001 JCL : STEP002/2 -> NORMAL - RC(0)
JCL001 JCL : STEP003/3 -> Start EXEC C:\Program Files\Raincode\Batch\IEFBR14.exe
JCL001 JCL : Starting retrieveConnectionString
JCL001 JCL : Finishing retrieveConnectionString with correct data
JCL001 JCL : Cataloging Dataset OUTPUT : C:\ProgramData\Raincode\Batch\DefaultVolume\ARC01\OUTPUTFILE.seq
JCL001 JCL : STEP003/3 -> NORMAL - RC(0)
JCL001 JCL : Stop execution of steps / catalog cleanup
JCL001 JCL : Deleting temporary files
JCL001 JCL : Job finished with status: NORMAL - CODE(0) RC(0)
JCL001 JCL : LockingService Shutdown()
```

Automatic Restart – Output (1)



- Snapshot 1 is taken before COB001 is started.
 - DBDUMP reflects the contents of the database at the time of the snapshot.
 - OUTFILE is a sequential file produced by COB001 and is still empty at this point.

ARC0(x).SNAP01.DBDUMP

Accountnumber	Amount
ARC01-01	0,00
ARC01-02	0,00
ARC01-03	0,00
ARC01-04	0,00
ARC01-05	0,00
ARC01-06	0,00
ARC01-07	0,00
ARC01-08	0,00
ARC01-09	0,00
ARC01-10	0,00
ARC01-11	0,00
ARC01-12	0,00
ARC01-13	0,00
ARC01-14	0,00
ARC01-15	0,00
ARC01-16	0,00
ARC01-17	0,00
ARC01-18	0,00
ARC01-19	0,00
ARC01-20	0,00

ARC0(x).SNAP01.OUTFILE

ARC01.SNAP01.OUTFILE is still empty at this point

Ln 1, Col 1

Automatic Restart – Output (2)



- SNAP02: after abend, another snapshot is taken.

ARC0(x).SNAP02.DBDUMP lists the state of the database at this point.

DBDUMP.seq - Notepad

File Edit Format View Help

CONTENTS OF DBO.AMOUNT

Accountnumber	Amount
ARC01-01	0,10
ARC01-02	0,10
ARC01-03	0,10
ARC01-04	0,10
ARC01-05	0,10
ARC01-06	0,10
ARC01-07	0,10
ARC01-08	0,10
ARC01-09	0,10
ARC01-10	0,10
ARC01-11	0,10
ARC01-12	0,10
ARC01-13	0,10
ARC01-14	0,10
ARC01-15	0,10
ARC01-16	0,00
ARC01-17	0,00
ARC01-18	0,00
ARC01-19	0,00
ARC01-20	0,00

Ln 25, Col 1

Rollback

abend

ARC0(x).SNAP02.OUTFILE is truncated after the 15th account

OUTFILE.seq - Notepad

File Edit Format View Help

COB001 UPDATES

Accountnumber	Amnt Before	Amnt After
ARC01-01	0,00	0,10
ARC01-02	0,00	0,10
ARC01-03	0,00	0,10
ARC01-04	0,00	0,10
ARC01-05	0,00	0,10
ARC01-06	0,00	0,10
ARC01-07	0,00	0,10
ARC01-08	0,00	0,10
ARC01-09	0,00	0,10
ARC01-10	0,00	0,10
ARC01-11	0,00	0,10
ARC01-12	0,00	0,10
ARC01-13	0,00	0,10
ARC01-14	0,00	0,10
ARC01-15	0,00	0,10

Ln 12, Col 80

The lines containing accounts 16 through 20 were actually written by COB001 but are removed by the TRMBEFORCKP after the abend

Automatic Restart – Output (3)



- SNAP03: Creates a new snapshot (after the successful ending).

ARC0(x).SNAP03.DBDUMP

DBDUMP.seq - Notepad

File Edit Format View Help

CONTENTS OF DBO.AMOUNT

Accountnumber	Amount
ARC01-01	0,10
ARC01-02	0,10
ARC01-03	0,10
ARC01-04	0,10
ARC01-05	0,10
ARC01-06	0,10
ARC01-07	0,10
ARC01-08	0,10
ARC01-09	0,10
ARC01-10	0,10
ARC01-11	0,10
ARC01-12	0,10
ARC01-13	0,10
ARC01-14	0,10
ARC01-15	0,10
ARC01-16	0,10
ARC01-17	0,10
ARC01-18	0,10
ARC01-19	0,10
ARC01-20	0,10

← Final Commit

ARC0(x).SNAP03.OUTFILE

OUTFILE.seq - Notepad

File Edit Format View Help

COB001 UPDATES

Accountnumber	Amnt Before	Amnt After
ARC01-01	0,00	0,10
ARC01-02	0,00	0,10
ARC01-03	0,00	0,10
ARC01-04	0,00	0,10
ARC01-05	0,00	0,10
ARC01-06	0,00	0,10
ARC01-07	0,00	0,10
ARC01-08	0,00	0,10
ARC01-09	0,00	0,10
ARC01-10	0,00	0,10
ARC01-11	0,00	0,10
ARC01-12	0,00	0,10
ARC01-13	0,00	0,10
ARC01-14	0,00	0,10
ARC01-15	0,00	0,10
ARC01-16	0,00	0,10
ARC01-17	0,00	0,10
ARC01-18	0,00	0,10
ARC01-19	0,00	0,10
ARC01-20	0,00	0,10

Files with National Characters

Files with National Characters



- National character strings are declared as PIC N(n) USAGE NATIONAL.
- Each character is represented by two or four bytes.
- Numeric edited variables can also be declared with the USAGE NATIONAL.
- Supported operations are:
 - computations,
 - output to the console,
 - viewing values in the debugger,
 - reading and writing to a database.

Files with National Characters



- To compile programs with the embedded national characters, **MOVE N'世界 ! ' TO REC** , the compilation command line must contain the following option:
StringSourceFileEncoding = utf-8
- National characters are encoded in UTF-16 big endian format.

Note: If the console cannot display Unicode characters, change the language to Japanese in [configuration/time and language/language/administrative language settings/change system locale] to make the Japanese characters visible. This requires rebooting of your machine.

Files with National Characters – Demo

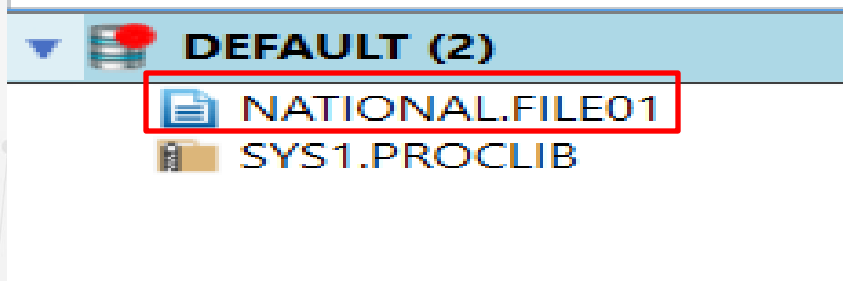


- This demo shows how national characters can be written to files.
- Demonstrates how the Record Editor can be used to view these files.
- This demo reads English and Japanese descriptions from a table file.

Files with National Characters – How to demonstrate



- Step 1: Open the solution from the folder Batch – Files with National Character.
- Step 2: Click on Start Raincode Debugger to execute the project.
- Step 3: To view the output file contents, double click the Catalog Explorer shortcut placed on the desktop.

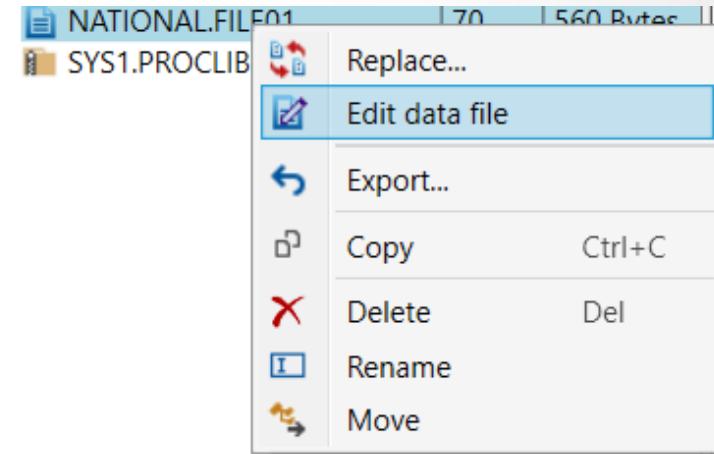


NATIONAL.FILE01 is the output file.
It contains both character fields and
national fields.

Files with National Characters – How to demonstrate



- Step 4: Right-click on the output file in the Catalog Explorer and select Edit data file to open it in the Record Editor.
- You can also directly open the File in the Record Editor by double-clicking the Record Editor shortcut placed on the desktop and navigating to the desired location. By default, all the output files are created at C:\ProgramData\Raincode\Batch\DefaultVolume.

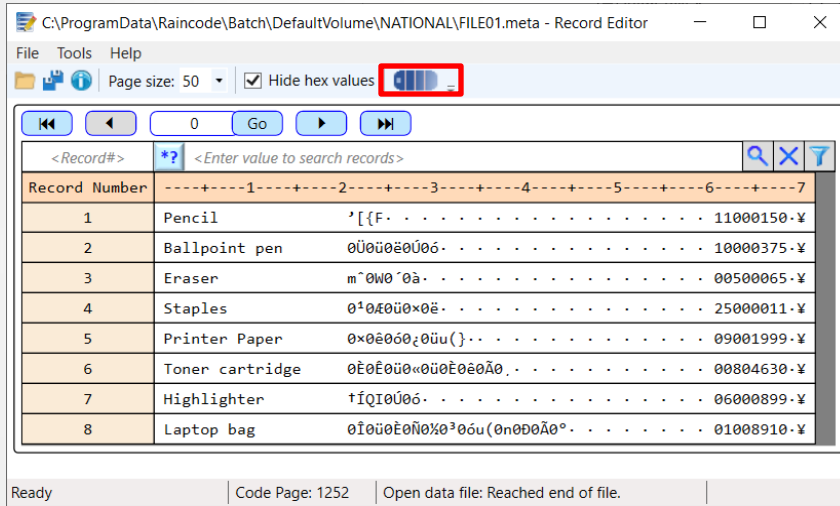


Files with National Characters – How to demonstrate

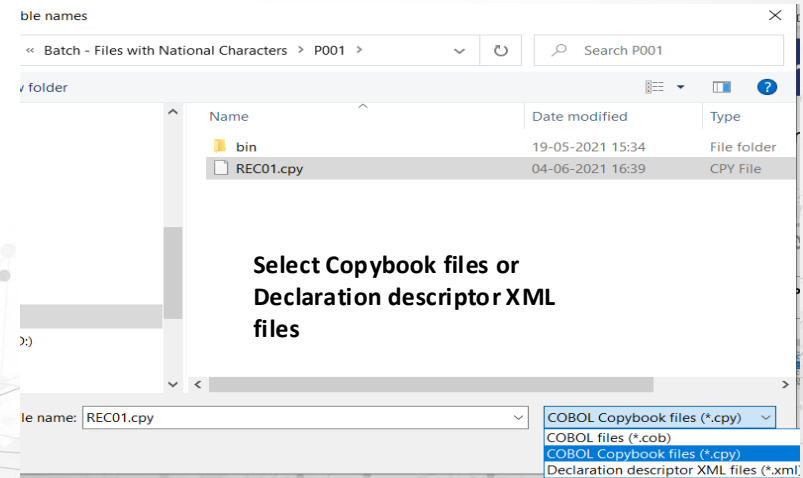


- Step 5: Switch to field view using the declaration descriptor or the copybook to make Japanese characters visible.

1.



2.



Run **gendd.bat** in the solution directory if you want to update *DeclarationDescriptors.xml*

Files with National Characters – How to demonstrate



3.

Select variable name

Details

COBOL Copybook file: C:\repositories\raincode-360\Batch - Files with Na

Change...

Variable name: REC01

REC01

OK

4.

C:\ProgramData\Raincode\Batch\DefaultVolume\NATIONAL\FILE01.meta - Record Editor

File Tools Help

Page size: 50 Hide hex values Variable name: REC01

<< < 0 Go > >>

<Record#> *? <Enter value to search records>

Record Number	REC01-NM-ENG	REC01-NM-JPN	REC01-QTY	REC01-PRICE	REC01-CURRENCY
1	Pencil	鉛筆	110	001.50	¥
2	Ballpoint pen	ボールペン	100	003.75	¥
3	Eraser	消しゴム	005	000.65	¥
4	Staples	ステープル	250	000.11	¥
5	Printer Paper	プリンター用紙	090	019.99	¥
6	Toner cartridge	トナーカートリッジ	008	046.30	¥
7	Highlighter	蛍光ペン	060	008.99	¥
8	Laptop bag	ノートパソコン用の	010	089.10	¥

Ready Code Page: 1252

FBA with EBCDIC



- This demo produces a dataset of type FBA that is encoded in the EBCDIC
- FBA is a print format, normally 133 bytes in length with 132 printable characters, where the first character is a carriage control, which determines the action of the print head after the line is printed.
 - The valid carriage control characters are:
 - 1: new page
 - 0: two new lines
 - +: no new line
 - " ": new line
- In this demo the character 0 is produced with the WRITE AFTER 2 statement in the COBOL.
- In this demo the data and the carriage control character are encoded in the EBCDIC.
- The following parameter in the COBOL build parameters of the project will ensure that the output of the program is encoded in EBCDIC:
 - :StringRuntimeEncoding=IBM037



- In this demo, however, we must also ensure that the **0** at the beginning of each line is also encoded in EBCDIC. We set the defaultCodePage in the catalog to EBCDIC, which is not the case in the default configuration file that is shared by most of the demos.
- For this reason, we provide a private catalog configuration file in the solution folder with the following line:
 - `<catalogConfiguration dbConnectionDataProvider="RcDbConnections.csv" defaultCodePage="IBM037" ...`
- Because our program uses a database connection, we also provide a private copy of the connection file **RcDbConnections.csv** that can be pointed to by the catalog configuration.

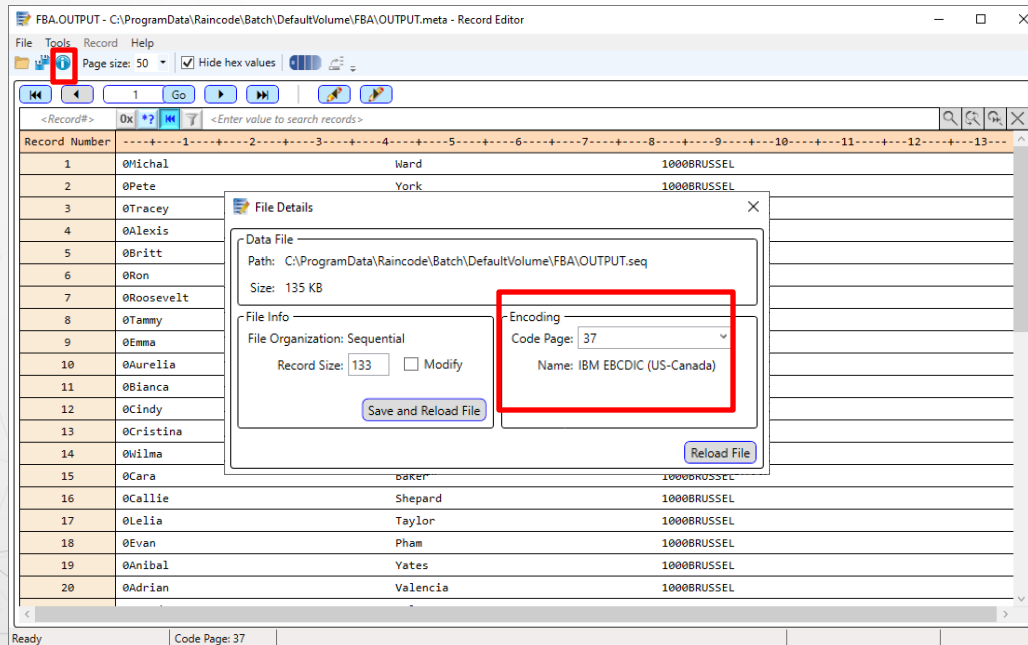


- The submit utility will look for these files in the bin folder, the current folder from where it is executed. To make sure that it finds these files, we will copy them from the solution folder to the bin folder with a post-build event:
 - `copy /Y "$(SolutionDir)\Raincode.Catalog.xml" . && copy /Y "$(SolutionDir)\RcDbConnections.csv"`
- The **&&** part is needed so that both files can be copied with a single command.
- Finally, we need to set the **Catalog Configuration** field of the common JCL properties of our project as follows so that it points to the private catalog configuration file in the bin folder:
 - Raincode.Catalog.xml

FBA with EBCDIC – How to demonstrate and see output



- Step 1: Open the solution from the folder Batch – FBA with EBCDIC.
- Step 2: Start the Raincode Debugger
- Step 3: Navigate to the folder C:\ProgramData\Raincode\Batch\DefaultVolume\FBA and double-click on the **OUTPUT. meta**.
- Step 4: The dataset **FBA.OUTPUT** will open in the **Record Editor**



Click on  to see that the code page is 37 (EBCDIC)

zTrieve- Compiled and Interpreted



- zTrieve reproduces the exact functionality of Easytrieve®.
- zTrieve offers two modes: the **intermediate** and the **compiled** mode.
- The Easytrieve® program is read from the JCL on the fly in the intermediate or interpreted mode.
- On the other hand, the compiled mode will generate standalone DLLs. Further, this DLL can be executed in two ways: via the command line or as part of a JCL.

zTrieve- Compiled



- Open the solution from the folder *C:\repositories\raincode-360\Batch - zTrieve Compiled*
- Start Raincode Debugger to execute the program. It will compile the program, create a DLL, and execute the DLL as part of a JCL.

```
//JCL001 JOB CLASS=A,MSGCLASS=C
//STEP001 EXEC PGM=ZT001
//SYSOUT DD SYSOUT=*
//PERSNL DD *
Bonita Pepper          123
Kathie Rosabella       001
Lynsey Sidney          888
Lotus Lorrin           789
Ami Loren              336
Sherie Lynwood         911
Jake Karen             529
Izabelle Chanelle      630
Evvie Walker           111
Lilac Mia              345
/*
//
```

Output

Job00000000000000000000 : Deleting Dataset PERSNL : C:\ProgramData\raincode\Batch\DefaultVolume\SYS\TEMP\JOB00000000000000000000\STEP1\W638568914231984254.seq

Job00000000000000000000 : [STEP001] => [NORMAL] - RC[0]/[0] HighestRC[0]

Job00000000000000000000 : Stop execution of steps / catalog cleanup

Job00000000000000000000 : Deleting temporary files

Job00000000000000000000 : Spooling :#SYSOUT:JOB00000000000000000000-JCL001.STEP001.SYSOUT to print class C

Job00000000000000000000 : Spooling Done

Job00000000000000000000 : Unlocking datasets

'Submit.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.32\System.Security.Principal.Windows.dll'. Skipped loading symbols. Module is optimized.

'Submit.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.32\System.Security.Claims.dll'. Skipped loading symbols. Module is optimized.

Job00000000000000000000 : SYSLOG=[C:\ProgramData\raincode\Batch\SYSOUT\JOB00000000000000000000-JCL001\SYSLOG.JCL001.txt]

Job00000000000000000000 : Job finished with status: NORMAL - CODE(0) RC(0)

Job00000000000000000000 : LockingService Shutdown()

'Submit.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.32\System.Threading.Overlapped.dll'. Skipped loading symbols. Module is optimized.

The thread '[Thread Destroyed]' (3940) has exited with code 0 (0x0).

The program '[6296] Submit.exe' has exited with code 0 (0x0).

zTrieve- Interpreted



- Open the solution from the folder *C:\repositories\raincode-360\Batch - zTrieve Interpreted*
- Start Raincode Debugger to execute the program.
- In this example, PGM=EZTPA00 interprets the program in SYSIN

The screenshot displays the Raincode Debugger interface. The main window shows the JCL001.jcl file with the following content:

```
JCL001.jcl
//JCL001 JOB CLASS=A,MSGCLASS=C
//STEP001 EXEC PGM=EZTPA00
//SYSOUT DD SYSOUT=*
//PERSNL DD *
Bonita Pepper          123
Kathie Rosabella       001
Lynsey Sidney          888
Lotus Lorrin           789
Ami Loren              336
Sherie Lynwood         911
Jake Karen             529
Izabelle Chanelle      630
Evvie Walker           111
Lilac Mia              345
/*
//SYSIN DD *
  PARM DEBUG(FLOW)
  *
FILE PERSNL FB(80 800)
NAME 1 25 A
```

The Output window at the bottom shows the following log:

```
100 %  No issues found
Show output from: Debug
JOB00000000000000000000 : Deleting Dataset PERSNL : C:\ProgramData\Raincode\Batch\DefaultVolume\SYS\TEMP\JOB00000000000000000000\STEP1\W638568993480989341.seq
JOB00000000000000000000 : Deleting Dataset SYSIN : C:\ProgramData\Raincode\Batch\DefaultVolume\SYS\TEMP\JOB00000000000000000000\STEP1\W638568993481140666.seq
JOB00000000000000000000 : [STEP001/I] => [NORMAL] - RC[0]/[0] HighestRC[0]
JOB00000000000000000000 : Stop execution of steps / catalog cleanup
JOB00000000000000000000 : Deleting temporary files
JOB00000000000000000000 : Spooling :#SYSOUT:JOB00000000000000000000-JCL001.STEP001.SYSOUT to print class C
JOB00000000000000000000 : Spooling Done
JOB00000000000000000000 : Unlocking datasets
'Submit.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.32\System.Security.Principal.Windows.dll'. Skipped loading symbols. Module is
'Submit.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.32\System.Security.Claims.dll'. Skipped loading symbols. Module is optimized
JOB00000000000000000000 : SYSLOG[C:\ProgramData\Raincode\Batch\SYSOUT\JOB00000000000000000000-JCL001\SYSLOG-JCL001.txt]
JOB00000000000000000000 : Job finished with status: NORMAL - CODE(0) RC(0)
JOB00000000000000000000 : LockingService Shutdown()
'Submit.exe' (CoreCLR: clrhost): Loaded 'C:\Program Files\dotnet\shared\Microsoft.NETCore.App\6.0.32\System.Threading.Overlapped.dll'. Skipped loading symbols. Module is opti
The thread '[Thread Destroyed] (8404) has exited with code 0 (0x0).
The program '[1152] Submit.exe' has exited with code 0 (0x0).
```

Legacy with MultiDatabase Access



- This demo showcases connecting COBOL programs to the BANKDEMO and SecondaryDBDemo databases within a single SQL Server instance via IKJEFT01 in a single JCL workflow. It reads data from multiple databases and writes the output to a file.
- The SQL Server connection relies on the following elements:
 - Plan name
 - Defined in the **Raincode.catalog.xml**:

```
<programDBConnectionDefinitions>
```

```
<dbConnection JobId=".*" >DEFAULTPLAN</dbConnection >
```

```
</programDBConnectionDefinitions>
```

- Connection String

- Defined in the **RcDbConnections.csv**

```
*,SqlServer,Data Source=<datasource>;Uid=<user>;Pwd=<password>;Initial Catalog=BankDemo;
```

How to run



- Open the solution from the folder Batch – Legacy with Multi Database Access
- Start the Raincode Debugger
- Output files are located at C:\ProgramData\Raincode\Batch\DefaultVolume\MULTIDB

Fetches from the BANKDEMO.

MULTIDB.FILE01 - C:\ProgramData\Raincode\Batch\DefaultVolume\MULTIDB\FILE0...

File Tools Record Help

Page size: 50 Hide hex values Variable name: W-REC

1 Go

<Record#> 0x *? <Enter value to search records>

Record Number	W-CUSTOMERID	W-FIRSTNAME	W-LASTNAME	W-ZIPCODE	W-CITY	W-SSN
1	0001	Kirk	Greer	2000	ANTWERP	200000
2	0002	Owen	Le	9000	GENT	900000
3	0003	Jospeh	Beard	6000	CHARLEI	600000
4	0004	Dale	Huff	4000	Glain	400000
5	0005	Michal	Ward	1000	BRUSSEI	100000
6	0006	Tuan	Hicks	5000	Beez	500000
7	0007	Carlton	Spence	3000	LEUVEN	300000
8	0008	Trinidad	Forbes	7000	MONS	700000

Fetches from the SecondaryDB Demo

Unload MultiDatabase



- This demo showcases how to use the DSNTIAUL utility to copy data from multiple tables across different databases into a sequential file.
- The SQL server connection relies on the following elements:

- Plan name

- Defined in the **Raincode.catalog.xml**:

```
<programDBConnectionDefinitions>
```

```
<dbConnection JobId=".*" >DEFAULTPLAN</dbConnection >
```

```
</programDBConnectionDefinitions>
```

- Connection String

- Defined in the **RcDbConnections.csv**

```
*,SqlServer,Data Source=<datasource>;Uid=<user>;Pwd=<password>;Initial Catalog=BankDemo;
```

How to run



- Open the solution from the folder Batch – Unload with MultiDatabase
- Start the Raincode Debugger
- Output files are located at C:\ProgramData\Raincode\Batch\DefaultVolume\JCLUNLD\MULTIDB

JCLUNLD.MULTIDB.UNLOAD - C:\ProgramData\Raincode\Batch\DefaultVolume\JCLU...

File Tools Record Help

Page size: 50 Hide hex values Variable name: W-REC

Record Number	W-CUSTOMERID	W-FIRSTNAME	W-LASTNAME	W-ZIPCODE	W-CITY	W-SSN
1	0001	..Greer....	..00002000			
2	0002	..Le.....	..00009000			
3	0003	..Beard....	..00006000			
4	0004	..Huff.....	..00004000			
5	0005	..Ward.....	..00001000			

W-CUSTOMERID and W-SSN are fetched from the *SecondaryDBDemo*

W-LASTNAME and W-ZIPCODE are fetched from the *BANKDEMO*

zBridge

zBridge-Introduction



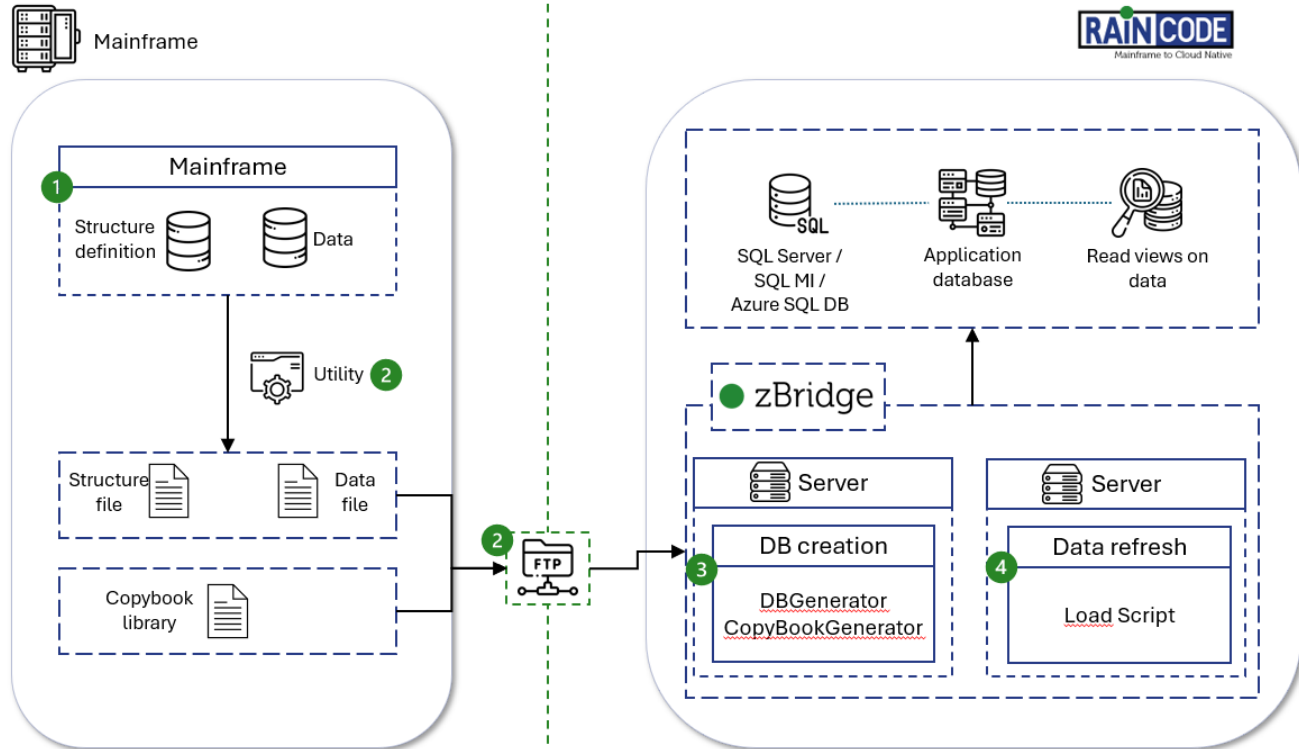
- zBridge is an innovative solution from Raincode that exposes mainframe data as SQLServer views.
- It transforms complex, non-relational datasets into easily accessible relational formats and eliminates the need for deep data restructuring or long, laborious cleanups.
- This, in turn, allows organizations to gain value from their data without fully migrating or disrupting existing systems.

zBridge-Introduction



Using zBridge is a four-step process:

1. Non-Relational Data Extraction
2. Data Transfer
3. Automated DB Creation
4. Data loading





To run this demo, follow these steps:

- Run the script ``generateTable.ps1`` to produce a table creation script named ``table_creation.sql`` based on the catalog configuration (C:\Raincode360\Repositories\Demos\zBridge\zBridgeFile\catalog\Raincode.Catalog.xml). There is no need to execute the ``table_creation.sql`` script since the table creation was already taken care of during this Raincode360 VM provisioning.
- Next, run the script ``generate view.ps1`` to generate the SQL view creation script named ``ACCOUNT_view.sql`` based on the copybook. There is no need to execute ``ACCOUNT_view.sql`` script since the view creation was already taken care of during this Raincode360 VM provisioning.



- Finally, run the script `file.LoadData.ps1` to load the data into the `[VsamSql].[REC\_ACCOUNT]` table. The view `[VsamSql].[REC\_ACCOUNT\_V]` can be used to query the data in a more user-friendly format.

```
select top(5) [RID], [W_ACCOUNT_ID], [W_CUSTOMERID_ID], [W_AMOUNT]
from [REC_ACCOUNT_V]
```

100 %

Results Messages

	RID	W_ACCOUNT_ID	W_CUSTOMERID_ID	W_AMOUNT
1	1	BE00-1000-0000-1438	832	+00001064.80
2	2	BE00-1000-0000-2643	70	+00004541.89
3	3	BE00-1000-0000-3557	922	+00002802.74
4	4	BE00-1000-0000-4464	818	+00002631.55
5	5	BE00-1000-0000-4488	899	+00005565.00



- Demos:
 - [Raincode QIX](#)
 - [How to debug a CICS application](#)
 - [Running a QIX program from a command line](#)
 - [Redefining LINK](#)
 - [Redefining SEND and RECEIVE](#)
 - [Legacy with database access](#)
 - [Asynchronous Processing with START/ RETRIEVE](#)
 - [VSAM – C# calling legacy with VSAM access using CICS LINK](#)
 - [Create REST API from Legacy Program](#)
 - [Create REST API from Legacy Program –Rainbox Generated](#)
 - [Online Internal Reader](#)
 - [CWA](#)



- Raincode QIX is a CICS emulator for the .NET and the Azure Platforms
- It is the ideal companion for the Raincode Legacy Compilers
- In addition to your existing COBOL and PL/I applications, QIX also allows you to develop new transactions in C#
- Provides a robust, distributed, and scalable architecture
- Support for dynamically added processing servers



- Usage of the **Raincode QIX Emulator**
- Six CICS transactions written in COBOL
- Six BMS maps
- One transaction written in PL/I (screenless)
- Programming Features:
 - Obtaining and passing a COMMAREA
 - Embedded SQL
 - SQL Server database
- CICS LINK, XCTL, SEND and RECEIVE
- Application Features:
 - Validation, error messages, highlighting of errors
 - Cursor-sensitive fields
 - Navigation via PF-Keys
 - List/selection screens with filtering

Raincode QIX – Starting the Raincode Bank Demo



- To start the demo, launch the “Start CICS demo” on the desktop.

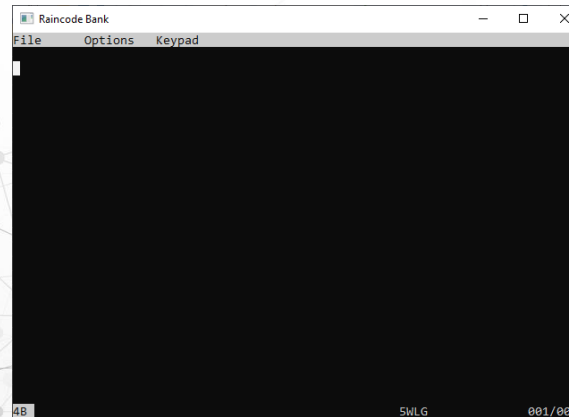
Three additional windows will open up (with a short delay)

- Terminal Server (TS)
- Processing Server (PS)

```
TS
[2020-05-31 11:54:02.617] [1] [TRACE] [RainCode.Core.CommandLine]: -TCTUASIZE=512
=> TCTUASIZE = 512
[2020-05-31 11:54:02.617] [1] [TRACE] [RainCode.Core.CommandLine]: -BindAddress=lo
calhost => BindAddress = localhost
[2020-05-31 11:54:02.625] [1] [TRACE] [RainCode.QIX.Runners]: Running server in A
ppDomain RainCode.QIX.TerminalServer.TerminalServerRunner_48902609-dbf7-44d7-8bb8
-49cd563ce338
[2020-05-31 11:54:02.649] [1] [WARNING] [QIX]: Creating log file at C:\Users\shipra\
AppData\Roaming\RainCode QIXMS\Terminal server_NA_BANKDEMO_31052020_115402.log
[2020-05-31 11:54:02.669] [1] [INFO] [RainCode.Core.Plugin]: Plugin Implementatio
n: RainCode.QIX.Plugins.ADPlugin.QixAccountSecurityManager, ADPlugin, Version=4.0
.261.0, Culture=neutral, PublicKeyToken=null.TryBuild from rccorlib, Version=1.0
.5.0, Culture=neutral, PublicKeyToken=44ab4e15cfffad0e3 providing an implementati
on of plugin: TerminalServer.SecurityManagerPlugin
[2020-05-31 11:54:02.670] [1] [INFO] [RainCode.Core.Plugin]: Plugin Implementatio
n: RainCode.QIX.Plugins.ADPlugin.QixmsAccountSecurityManager, ADPlugin, Version=4
.0.261.0, Culture=neutral, PublicKeyToken=null.TryBuild from rccorlib, Version=1
.0.5.0, Culture=neutral, PublicKeyToken=44ab4e15cfffad0e3 providing an implementati
on of plugin: TerminalServer.QixmsSecurityManagerPlugin
[2020-05-31 11:54:02.671] [1] [INFO] [QIX]: File C:\Program Files\Raincode\RainCo
de QIXMS\plugins\Terminal server\ADPlugin.dll loaded as plugin
[2020-05-31 11:54:06.803] [1] [INFO] [QIX]: Could not discover region BANKDEMO ma
nager after 1 attempt(s). Retrying...
```

```
PS
[2020-05-31 11:54:12.378] [1] [TRACE] [RainCode.Core.CommandLine]: -BindAddress=loca
lhost => BindAddress = localhost
[2020-05-31 11:54:12.389] [1] [TRACE] [RainCode.QIX.Runners]: Creating new AppDomain
from AppDomain ProcessingServerRunner.exe
[2020-05-31 11:54:12.635] [1] [TRACE] [RainCode.Core.CommandLine]: -BindAddress=loca
lhost => BindAddress = localhost
[2020-05-31 11:54:12.645] [1] [TRACE] [RainCode.QIX.Runners]: Running server in AppD
omain RainCode.QIX.ProcessingServer.ProcessingServerRunner_1ce1fc0b-c185-413f-a38f-2
e5d800ede0e
[2020-05-31 11:54:12.674] [1] [WARNING] [QIX]: Creating log file at C:\Users\shipra\
AppData\Roaming\RainCode QIXMS\Processing server_NA_BANKDEMO_31052020_115412.log
[2020-05-31 11:54:12.694] [1] [INFO] [RainCode.Core.Plugin]: Plugin Implementatio
n: RainCode.QIXMS.Plugins.AltDestinationPlugin.AltDestinationHandler, AltDestinationPlu
gin, Version=4.0.261.0, Culture=neutral, PublicKeyToken=null.TryBuild from rccorlib
, Version=1.0.5.0, Culture=neutral, PublicKeyToken=44ab4e15cfffad0e3 providing an impl
ementation of plugin: ProcessingServer.QixmsAltDestinationPlugin
[2020-05-31 11:54:12.695] [1] [INFO] [QIX]: File C:\Program Files\Raincode\RainCode
QIXMS\plugins\Processing server\AltDestinationPlugin.dll loaded as plugin
[2020-05-31 11:54:13.149] [1] [INFO] [RainCodeLegacyRuntime]: Registering module RC_
DEBUGBREAK, from RainCodeLegacyRuntime.Module.BuiltinModules.DebugBreak, RainCodeL
egacyRuntime, Version=4.0.261.0, Culture=neutral, PublicKeyToken=b351c3853ee4a149,
static memory size 0
[2020-05-31 11:54:13.168] [1] [INFO] [RainCodeLegacyRuntime]: Registering module RC_
CHECKHEAP, from RainCodeLegacyRuntime.Module.BuiltinModules.CheckHeap, RainCodeLegac
yRuntime, Version=4.0.261.0, Culture=neutral, PublicKeyToken=b351c3853ee4a149, stati
```

- Raincode Bank (in 3270 terminal emulator, connects to TS)



TS/PS are minimized by default. Together, they implement the Raincode QIX emulator.

Raincode QIX – Raincode Bank Demo (3270 terminal)



- The Raincode Bank window is a 3270 terminal emulator session that allows you to interact with the QIX.
- Type C004 and press Enter to start the application

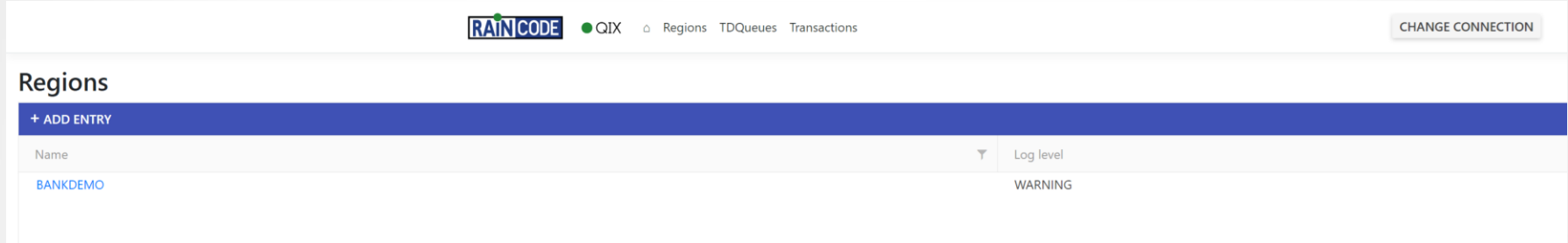
CustNo	Lastname	FirstName	ZIP	City	Birthdate
000001	Greer	Kirk	2000	ANTWERPEN	1958-05-17
000002	Le	Owen	9000	GENT	1958-07-08
000003	Beard	Jospeh	6000	CHARLEROI	1962-09-07
000004	Huff	Dale	4000	Glain	1982-06-16
000005	Ward	Michal	1000	BRUSSEL	1964-10-20
000006	Hicks	Tuan	5000	Beez	1975-11-10
000007	Spence	Carlton	3000	LEUVEN	1967-01-27
000008	Forbes	Trinidad	7000	MONS	1995-08-28
000009	Hess	Cody	3500	HASSELT	1961-10-02
000010	Bates	Shirley	8000	BRUGGE	1997-07-21
000011	Stone	Jeromy	6700	ARLON	1985-07-05
000012	Price	Jewell	1300	Limal	1977-06-21
000013	Fry	Carlo	2000	ANTWERPEN	1970-09-13
000014	Haas	Joshua	9000	GENT	1974-12-19
000015	Spears	Royal	6000	CHARLEROI	1966-04-13
000016	Gaines	Bernard	4000	Glain	1981-01-09
000017	York	Pete	1000	BRUSSEL	1952-08-05
000018	Wolf	Neal	5000	Beez	1992-01-29

- Use the filter keys above the list to search for names and/or zip codes
- Position the cursor and press Enter to select a beneficiary account from a list.
- A transaction between two accounts can be created.

You can also see the regions and transaction details in the Raincode Console.

Raincode Bank Demo (Raincode Console)

BANKDEMO is the region which appears in the Raincode Console - QIX Panel when you execute the [Raincode Bank Demo](#).

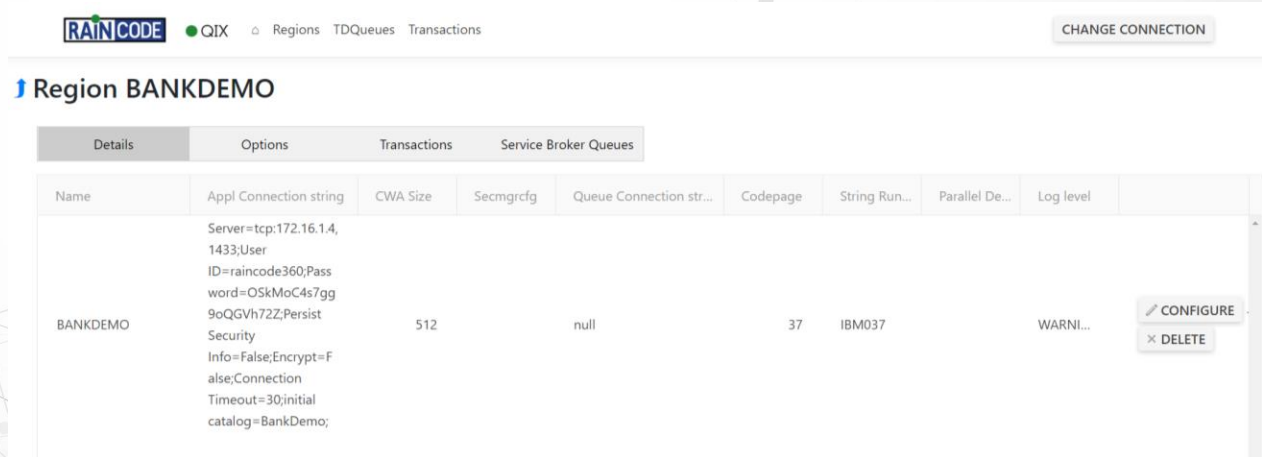


The screenshot shows the Raincode Console interface. At the top, there's a navigation bar with the Raincode logo, a green dot next to 'QIX', and tabs for 'Regions', 'TDQueues', and 'Transactions'. A 'CHANGE CONNECTION' button is on the right. Below the navigation bar, the 'Regions' section is active, showing a list of regions. A blue bar at the top of the list says '+ ADD ENTRY'. The list has two columns: 'Name' and 'Log level'. One region, 'BANKDEMO', is listed with a 'WARNING' log level.

Name	Log level
BANKDEMO	WARNING

Once you click the region name, it will provide you with more detailed information about that region like:

- Details
- Options
- Transactions
- Service Broker Queues

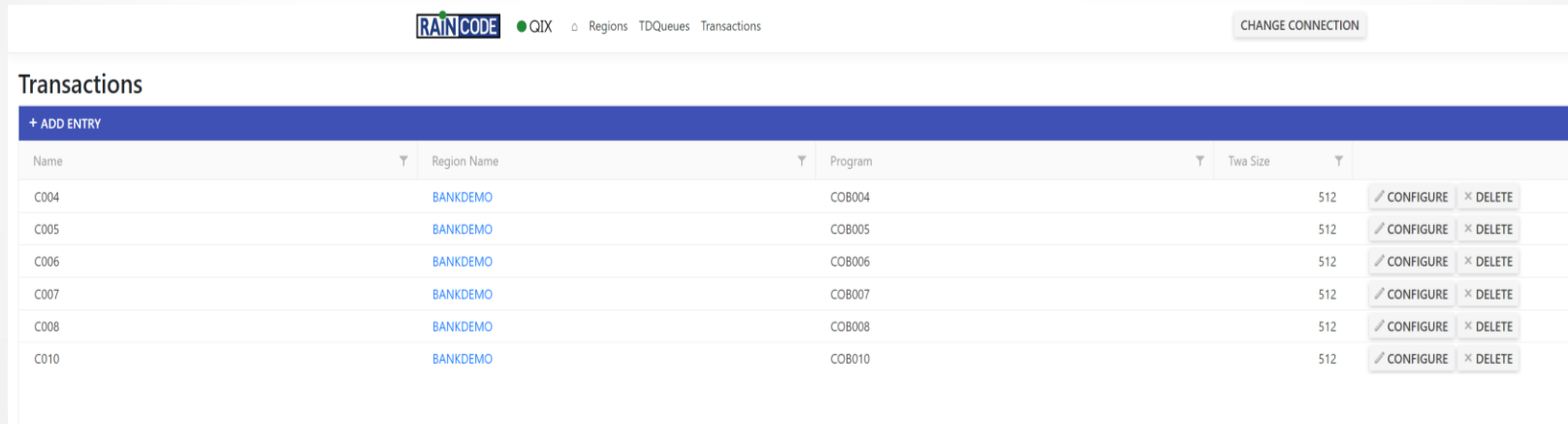


The screenshot shows the detailed view of the 'BANKDEMO' region in the Raincode Console. The navigation bar is the same as the previous screenshot. Below it, the title 'Region BANKDEMO' is displayed. There are four tabs: 'Details', 'Options', 'Transactions', and 'Service Broker Queues'. The 'Details' tab is selected, showing a table with columns: 'Name', 'Appl Connection string', 'CWA Size', 'Secmgrcfg', 'Queue Connection str...', 'Codepage', 'String Run...', 'Parallel De...', 'Log level', and an empty column. The 'BANKDEMO' row is highlighted. To the right of the table, there are 'CONFIGURE' and 'DELETE' buttons.

Name	Appl Connection string	CWA Size	Secmgrcfg	Queue Connection str...	Codepage	String Run...	Parallel De...	Log level	
BANKDEMO	Server=tcp:172.16.1.4,1433;User ID=raincode360;Password=OSkMoC4s7gg9oQGVh72Z;Persist Security Info=False;Encrypt=False;Connection Timeout=30;initial catalog=BankDemo;	512		null	37	IBM037		WARNI...	

Raincode Bank Demo (Raincode Console)

- The below transactions are configured while creating the Raincode Bank demo. Through the Raincode console, you can add a new transaction or configure the existing transaction.



The screenshot shows the Raincode console interface. At the top, there is a navigation bar with the Raincode logo, a green dot indicating QIX status, and links for Regions, TDQueues, and Transactions. A 'CHANGE CONNECTION' button is on the right. Below the navigation bar, the 'Transactions' section is active, showing a table of configured transactions. The table has columns for Name, Region Name, Program, Twa Size, and actions (CONFIGURE and DELETE). There are 6 transactions listed, all with Region Name 'BANKDEMO' and Program 'COB004' through 'COB010'. Each transaction has a 'Twa Size' of 512. The background of the slide features a network diagram with nodes and connecting lines.

Name	Region Name	Program	Twa Size	
C004	BANKDEMO	COB004	512	CONFIGURE DELETE
C005	BANKDEMO	COB005	512	CONFIGURE DELETE
C006	BANKDEMO	COB006	512	CONFIGURE DELETE
C007	BANKDEMO	COB007	512	CONFIGURE DELETE
C008	BANKDEMO	COB008	512	CONFIGURE DELETE
C010	BANKDEMO	COB010	512	CONFIGURE DELETE

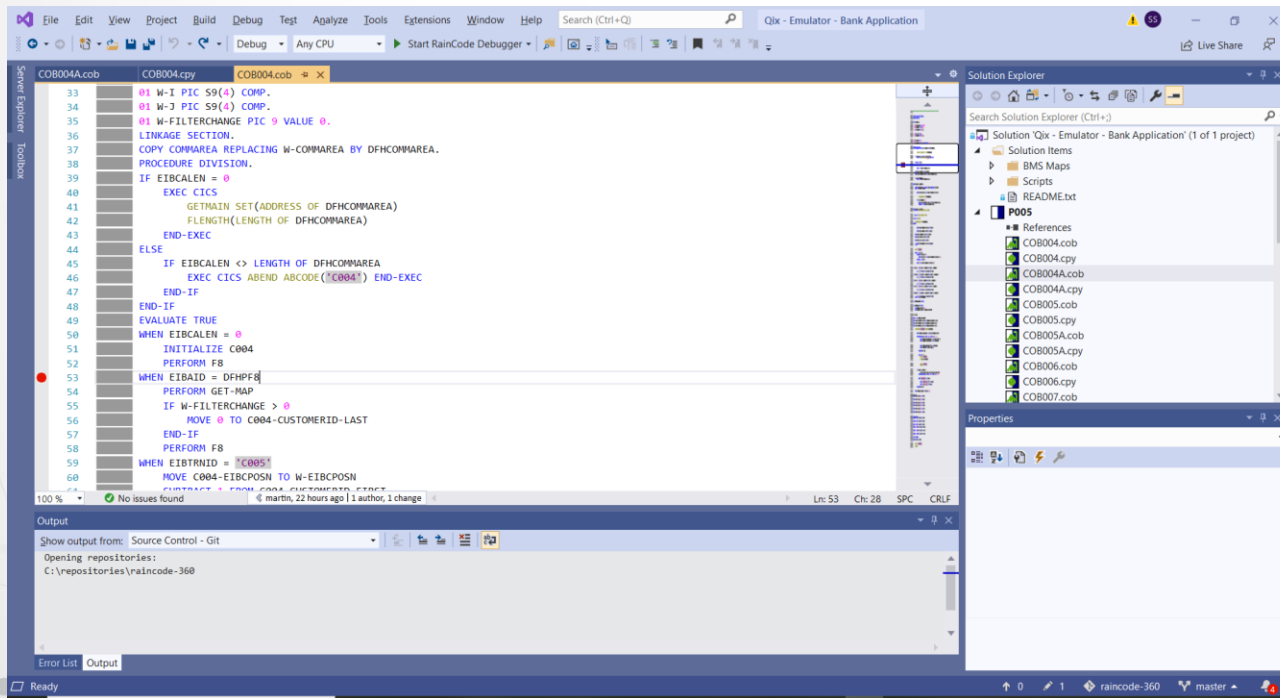
How to debug a CICS application



How to debug a CICS application



- Step 1: Open the solution from the folder Qix - Emulator - Bank Application.
- Step 2: Place a breakpoint at line 53 of program COB004.cob (`WHEN EIBAID = DFHPF8`) by clicking in the left margin. A red dot will appear.



- *Make sure that QIX is running and connected*
- *Start the application in the 3270 terminal emulator window*

How to debug a CICS application



- Step 3: In the Visual Studio, select Debug → Attach to Process (Ctrl+Alt+P)

Attach to Process

Connection type: Default

Connection target: raincode360-vm Find...

Connection type information
The default connection lets you select processes on this computer or a remote computer running the Visual Studio Remote Debugger (MSVSMON.EXE).

Attach to: Automatic: Managed (v4.6, v4.5, v4.0) code Select...

Available processes

Process	ID	Title	Type	User Name	Session
ProcessingServerRu...	11172	PS	Managed (v...	raincode360-vm\sh...	3
rdpclip.exe	6948		x64	raincode360-vm\sh...	3
RegionManagerRun...	6436	RM	Managed (v...	raincode360-vm\sh...	3
RuntimeBroker.exe	7924	Microsoft.Windows.Photos_2020.19111.2411...	x64	raincode360-vm\sh...	3
RuntimeBroker.exe	4160		x64	raincode360-vm\sh...	3
RuntimeBroker.exe	10692	Microsoft.YourPhone_1.20041.91.0_x64_8we...	x64	raincode360-vm\sh...	3
RuntimeBroker.exe	7920	Microsoft.SkypeApp_14.56.102.0_x64_kzf8q...	x64	raincode360-vm\sh...	3
RuntimeBroker.exe	5588	Microsoft.Windows.ShellExperienceHost_10...	x64	raincode360-vm\sh...	3
RuntimeBroker.exe	6064	Microsoft.MicrosoftEdge_44.17763.831.0_ne...	x64	raincode360-vm\sh...	3
RuntimeBroker.exe	5228	Microsoft.Windows.Cortana_1.11.6.17763_ne...	x64	raincode360-vm\sh...	3
PrintedSandhouf4	8412		x64	raincode360-vm\sh...	3

☐ Show processes from all users Refresh

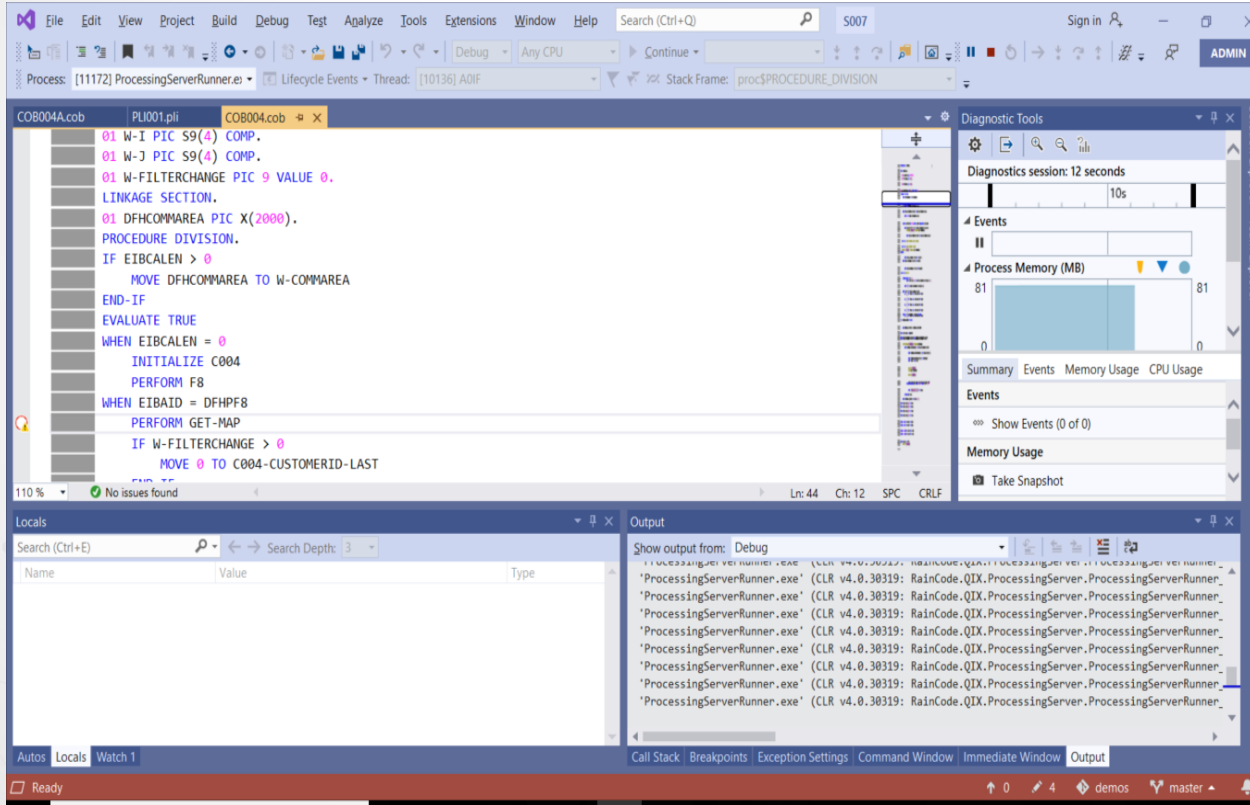
Attach Cancel

Select 'ProcessingServerRunner.exe' from the list of processes and click on the Attach button

How to debug a CICS application



- Visual Studio switches to debug mode.

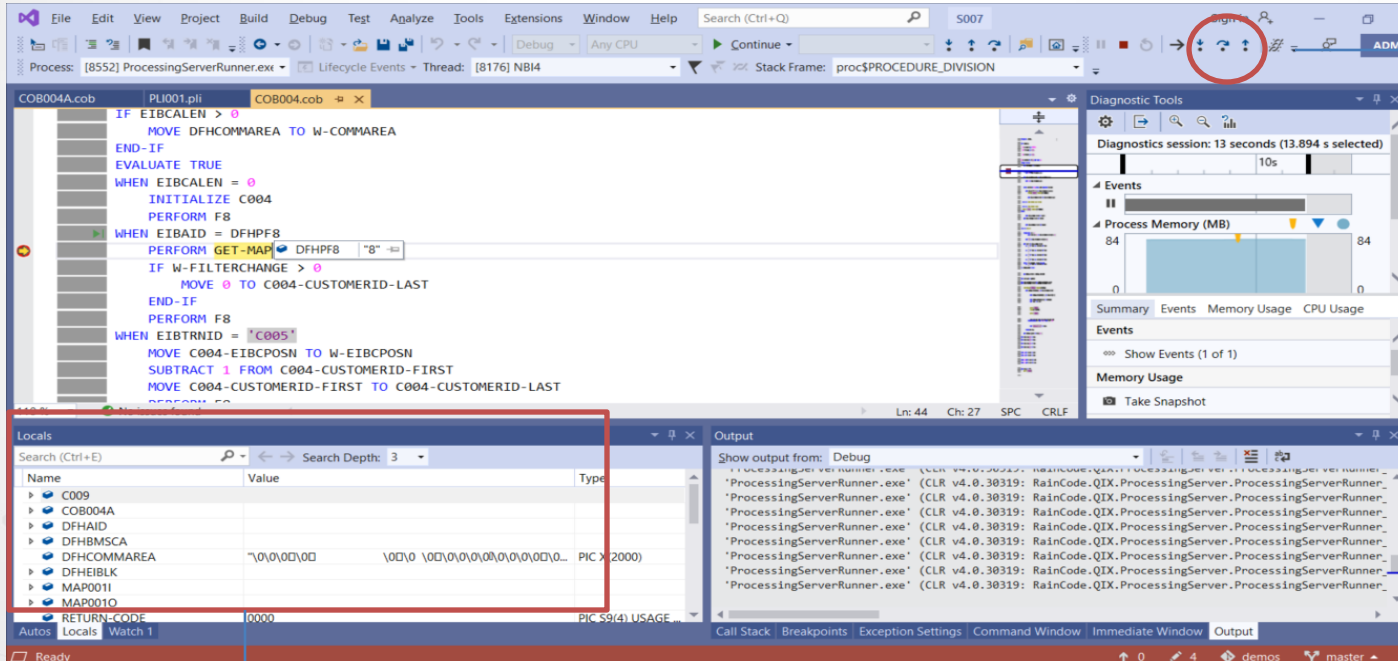


Because of the pseudo-conversational nature of the application, the program COB004.cob is not running at this point

How to debug a CICS application



- Step 4: Press **F8** on the main screen of the Raincode Bank application to scroll down. This will activate COB004.cob, and the execution will stop at the breakpoint.



Use the 'Step' buttons to step through the program

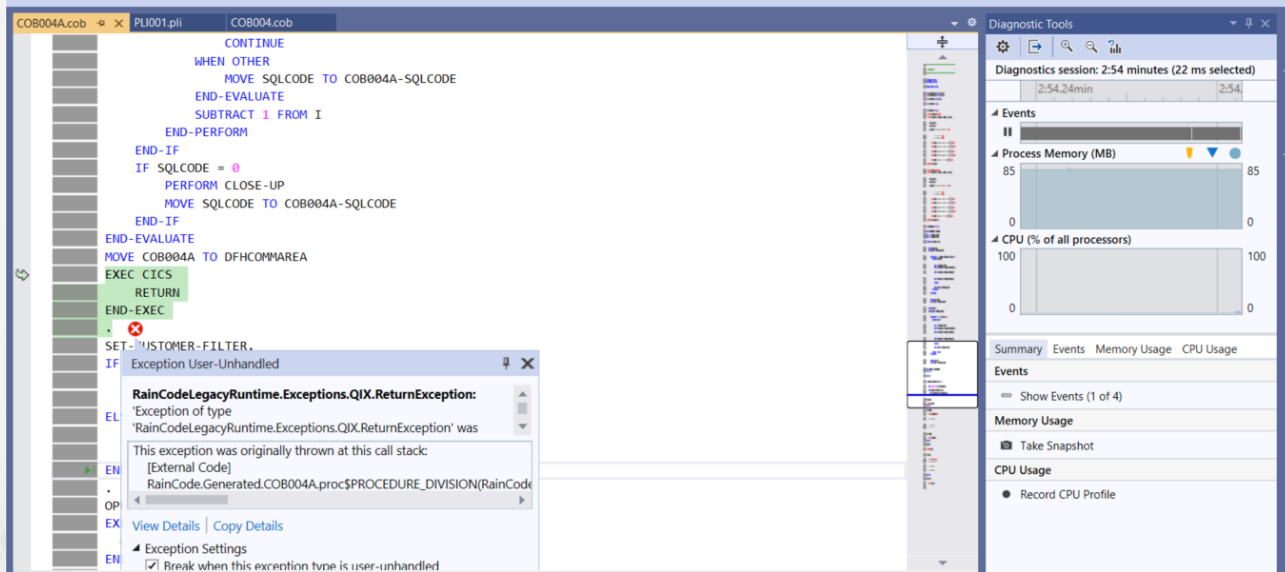
Hover over any variable in the source text of COB004.cob to see its contents.

Check the **Locals** window in Visual Studio to see a list of local variables and their corresponding values

How to debug a CICS application



- Stepping over the 'EXEC CICS RETURN' will trigger an exception. This is normal.
- Pressing “Continue” will end the program without detaching the debug session.
- Continue debugging by pressing **F8** or **Enter** in the Raincode Bank application.



Stop debugging by pressing the stop icon in the debug toolbar

Running a QIX program from a command line

Running a QIX program from a command line



- This demo runs a COBOL/QIX program from the command line without a runner
- Step 1: Open the solution from the folder QIX -Legacy – Run from the Command Line

There are two ways to RUN the program:

- Step 2a: Place a breakpoint on the last line of MAIN.Cob and start the Raincode debugger
- Step 2b: Run `run.bat` from the command line

```
00020  
01234567890123456789  
15:05:10
```

Redefining LINK

Redefining LINK – Introduction

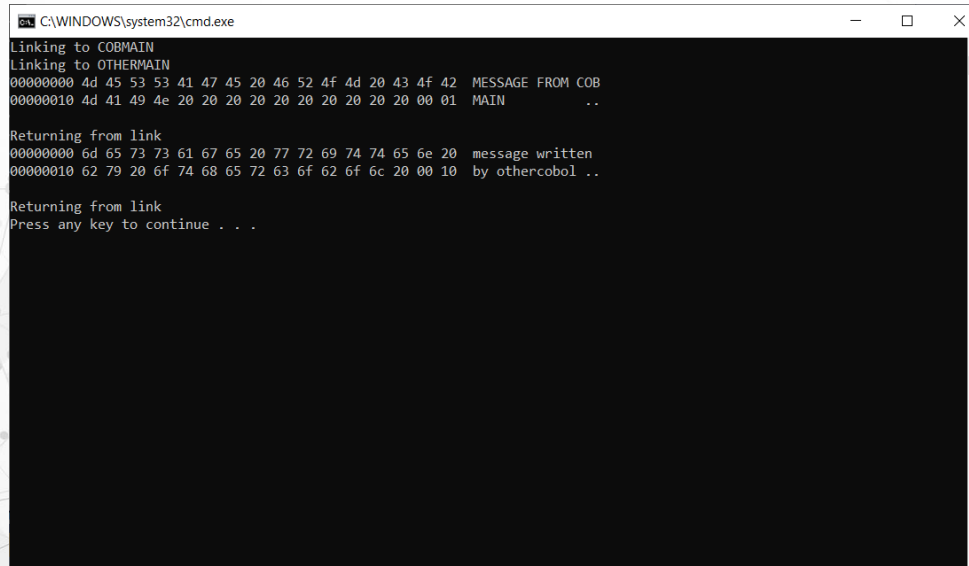


- Demonstrate how the behavior of CICS commands can be overwritten using Qix factories.
- The new link command writes a message plus a hex dump of the commarea before and after the actual link.
- The original link mechanism is invoked by the `base.Execute()` in `CustomLink.cs`.
- For more details, see [Qix-Factory](#).

Redefining LINK – How to run and see output



- Step 1: Open the solution from folder Qix - Factories - Redefining LINK
- Step 2: Select start without debugging (Ctrl + F5).
 - It will avoid automatic breakpoints and will also keep the console window open after the program ends
- You can check the messages in the console window



```
C:\WINDOWS\system32\cmd.exe
Linking to COBMAIN
Linking to OTHERMAIN
00000000 4d 45 53 53 41 47 45 20 46 52 4f 4d 20 43 4f 42 MESSAGE FROM COB
00000010 4d 41 49 4e 20 20 20 20 20 20 20 20 20 20 00 01 MAIN ..

Returning from link
00000000 6d 65 73 73 61 67 65 20 77 72 69 74 74 65 6e 20 message written
00000010 62 79 20 6f 74 68 65 72 63 6f 62 6f 6c 20 00 10 by othercobol ..

Returning from link
Press any key to continue . . .
```

Redefining SEND and RECEIVE

Redefining SEND and RECEIVE – Introduction



- Redefine the behaviour of the standard CICS commands SEND and RECEIVE by providing custom factories.
- SEND and RECEIVE are traditionally used to send and receive text messages to and from the terminal.
- The behavior of these commands is changed so that they can send and receive data structures to and from an in-memory queue.
- The main program asks the user for a city prefix and calls a subroutine (via CICS LINK) to retrieve all cities matching this prefix from the database.
- The subroutines push a variable number of results on the internal queue.
- The main routine reads the results from the queue and shows them on the screen.

Redefining SEND and RECEIVE – How to run and see output

- Step 1: Open the solution from the folder Qix - Factories - Redefining SEND and RECEIVE
- Step 2: Select start without debugging (Ctrl + F5).

It will avoid automatic breakpoints and will also keep the console window open after the program ends.

- Enter a prefix of a Belgian city ('BR' for Brussels will work just fine)

```
Please enter a city prefix: BR
BRAINE-L'ALLEUD          Length: 0015
BRAINE-LE-CHATEAU        Length: 0017
BRAINE-LE-COMTE          Length: 0015
BRAKEL                   Length: 0006
BRAS                     Length: 0004
BRASSCHAAT               Length: 0010
BRECHT                   Length: 0006
BREDENE                  Length: 0007
BREENDONK                Length: 0009
BRESSOUX                 Length: 0008
BRIELEN                  Length: 0007
BROECHEM                 Length: 0008
BRUCARGO                 Length: 0008
BRUGELETTE               Length: 0010
BRUGGE                   Length: 0006
BRUSSEGEEM               Length: 0009
BRUSSEL                  Length: 0007
BRUSSEL X                 Length: 0009
BRUYELLE                 Length: 0008
BRYE                     Length: 0004
Press any key to continue . . .
```

Legacy with the database access

Legacy with the database access – Introduction



- C# calling legacy with the database access using CICS LINK
- Demonstrates how a COBOL program can be called from a C# main program (a proxy)
- The C# program uses the CICS LINK mechanism to invoke the COBOL Program
- Demonstrates how to prepare an execution context with a database connection at the C# level
- In this demo, the legacy program fetches an item (minimum zip code) from the database, displays it on the console and exits.

Legacy with database access – How to demonstrate



- Step 1: Open the solution from the folder Qix - Legacy with Database Access - Csharp Main
- Step 2: Select Start without debugging (Ctrl + F5).

It will avoid automatic breakpoints and will keep the console window open after the program ends

A screenshot of a Windows command prompt window. The title bar reads 'C:\WINDOWS\system32\cmd.exe'. The window contains the text: 'Minimum zipcode: 1000' followed by 'Press any key to continue . . .'. The rest of the window is black, indicating it is waiting for a key press.

```
C:\WINDOWS\system32\cmd.exe
Minimum zipcode: 1000
Press any key to continue . . .
```

Asynchronous Processing with START / RETRIEVE



- This demo implements the CICS START and RECEIVE commands with Qix factories
- The START command implements a 'fire-and-forget' mechanism by starting a new asynchronous transaction: **START TRANSID('name') FROM(data)**, whereby **data** can be any part of working storage.
- If CICS cannot execute the new transaction immediately, it will be queued to start at a later point in time. The original transaction does not wait for the new transaction to be started. The return status of the command indicates that the new transaction has been successfully queued.
- The transaction receives the data with the following command: **RETRIEVE INTO(data)**, whereby **data** refers to the working storage of the transaction.



- PRACTICAL APPLICATIONS OF START AND RECEIVE
 - The START and RECEIVE commands are often used to create a listener for incoming data on a queue.
 - The listener will launch a separate task to process the data whenever data is received.

Asynchronous Processing with START / RETRIEVE- How to demonstrate



- Step 1: Open the solution from the folder Qix Factories - START and RETRIEVE
- Step 2: Select Start without debugging (Ctrl + F5).
- Expected output:

```
Microsoft Visual Studio Debug Console
ETTERBEEK      Laken      BRUSSEL

SINT-JANS-MOLENBEEK
SINT-GILLIS
Postcheque
EVERE          Neder-Over-Heembeek

OUDERGEM
UKKEL
SINT-LAMBRECHTS-WOLUWE

C:\repositories\raincode360\Qix Factories - START and RETRIEVE\Runner\bin\Debug\netcoreapp3.1\Runner.exe (process 1760)
exited with code 0.
Press any key to close this window . . .
```

- The program launches ten separate tasks to retrieve the cities associated with certain zip codes.
- The output can appear out of order depending on which task finishes first.

VSAM - C# calling legacy with VSAM access using CICS LINK



- The program demonstrates how to Write and Read VSAM files from CICS modules.
 - The legacy program COBWRITE fetches account information from the database and writes it to the VSAM file.
 - The legacy program PLIREAD reads and displays the VSAM file for a particular record on the console.

C# calling legacy with VSAM access using CICS LINK– How to demonstrate



- Step 1: Open the Powershell console via Tools->Command Line->Developer Powershell.
- Step 2: Navigate to the catalog folder using the command: `cd catalog`.
- Step 3: Execute the script *CatalogVsam.ps1* to catalog the empty VSAM file .

```
Developer PowerShell Visual Studio Community 2022 17.12.35527.113
PS C:\Raincode360\Repositories\Demos\Qix - Legacy with VSAM File Access - Csharp Main> cd catalog
PS C:\Raincode360\Repositories\Demos\Qix - Legacy with VSAM File Access - Csharp Main\catalog> .\CatalogVsam.ps1
Using Catalog manager configuration at C:\Raincode360\Repositories\Demos\Qix - Legacy with VSAM File Access - Csharp Main\catalog\Raincode.catalog.xml
JOB000000000003 : CatalogConfig Msg:configuration.SysoutFolderPath changed from: SYSOUT to: C:\Raincode360\Repositories\Demos\Qix - Legacy with VSAM File Access - Csharp Main\catalog\SYSOUT
JOB000000000003 : CatalogConfig Msg:AutoEditCalendarFolderPath changed from: AutoEditCalendars to: C:\Raincode360\Repositories\Demos\Qix - Legacy with VSAM File Access - Csharp Main\catalog\AutoEditCalendars
JOB000000000003 : Scanning CATGVSAM.jcl
JOB000000000003 : Setting RCRETURNCODEPATH = C:\Raincode360\Repositories\Demos\Qix - Legacy with VSAM File Access - Csharp Main\catalog\SYSOUT\JOB000000000003-CATGVSAM\RETURNCODE.TXT
JOB000000000003 : Starting locking service:18104
JOB000000000003 : Locking service active:18104
JOB000000000003 : Start execution Phase EnhGDG=True CM_NCT2=True ControlM=True
JOB000000000003 : MetaFileCaching=Enabled IOFileCaching=Enabled UpdLastRefDate=Disabled
JOB000000000003 : Start execution of steps
JOB000000000003 : ----- Step STEP001 -----
JOB000000000003 : STEP001/1 => Start EXEC C:\Program Files\Raincode\Crossbow\net8.0\bin\IDCAMS.rcexe
JOB000000000003 : Finishing retrieveConnectionString with no data
JOB000000000003 : (Execute) FileName: C:\Program Files\Raincode\Crossbow\net8.0\bin\IDCAMS.dll
JOB000000000003 : (Execute) Arguments:
JOB000000000003 : Start EXEC (SUBMIT internal call) C:\Program Files\Raincode\Crossbow\net8.0\bin\IDCAMS.dll
JOB000000000003 : Starting (InternalCall) IDCAMS :
JOB000000000003 : >DELETE VSAM.ACCOUNT.KSDS
JOB000000000003 : Locking VSAM.ACCOUNT.KSDS
JOB000000000003 : DataSetLock StepID:1 locking Add :DEFAULT:VSAM.ACCOUNT.KSDS : Exclusive
JOB000000000003 : StepID[1] [Exclusive] locking [:DEFAULT:VSAM.ACCOUNT.KSDS]
JOB000000000003 : Deleting dataset VSAM.ACCOUNT.KSDS
JOB000000000003 : >SET MAXCC = 0
JOB000000000003 : >DEFINE CLUSTER (NAME(VSAM.ACCOUNT.KSDS) INDEXED RECORDSIZE (0 34) KEYS (19 0) ) DATA (NAME(VSAM.ACCOUNT.KSDS.DAT)) INDEX (NAME(VSAM.ACCOUNT.KSDS.IDX))
JOB000000000003 : Allocating CLUSTER dataset VSAM.ACCOUNT.KSDS
JOB000000000003 : IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
JOB000000000003 : Deleting Dataset SYSIN : C:\Raincode360\Repositories\Demos\Qix - Legacy with VSAM File Access - Csharp Main\catalog\DefaultVolume\SYS\TEMP\JOB000000000003\STEP1\N638727009994742533.seq
JOB000000000003 : [STEP001/1] => [NORMAL] - RC[0]/[0] HighestRC[0]
JOB000000000003 : Stop execution of steps / catalog cleanup
JOB000000000003 : Deleting temporary files
JOB000000000003 : Unlocking datasets
JOB000000000003 : SYSLOG=[C:\Raincode360\Repositories\Demos\Qix - Legacy with VSAM File Access - Csharp Main\catalog\SYSOUT\JOB000000000003-CATGVSAM\SYSLOG.txt]
JOB000000000003 : Job finished with status: NORMAL - CODE(0) RC(0)
JOB000000000003 : LockingService Shutdown()
PS C:\Raincode360\Repositories\Demos\Qix - Legacy with VSAM File Access - Csharp Main\catalog>
```

C# calling legacy with VSAM access using CICS LINK– How to demonstrate



- Step 4: Rebuild the solution by selecting Build->Rebuild solution from the toolbar.
- Step 5: In the solution explorer, right-click *WriteVsam* project and select *Set as Startup Project*.
- Step 6: Run the *WriteVsam* project by clicking the **play** button on the toolbar. This process will write the VSAM file with the records fetched from the database.

```
Select Microsoft Visual Studio Debug Console
[2025-01-17 09:04:39.844] [1] [INFO] [ConfigurationSerializer]: Deserializing file ConfigV2
[2025-01-17 09:04:40.111] [1] [INFO] [ConfigurationSerializer]: Deserializing file ConfigV2 has succeeded.
[2025-01-17 09:04:40.114] [1] [INFO] [ConfigurationSerializer]: Applying default values for fields not present in XML.
[2025-01-17 09:04:40.128] [1] [WARNING] [RainCodeBatch]: Missing registry keys. Configuring with defaults.
[2025-01-17 09:04:40.362] [1] [INFO] [RainCodeLegacyRuntime]: Registering module RC_CHECKHEAP, from RainCodeLegacyRuntime.Module.BuiltinModules.CheckHeap, RainCodeLegacyRuntime, Version=5.0.46.0, Culture=neutral, PublicKeyToken=null, static memory size 0
[2025-01-17 09:04:40.409] [1] [INFO] [RainCodeLegacyRuntime]: Registering module QIXMODULE, from RainCodeLegacyRuntime.QIX.QixModule, RainCodeLegacyRuntime, Version=5.0.46.0, Culture=neutral, PublicKeyToken=null, static memory size 0
[2025-01-17 09:04:40.837] [1] [INFO] [RainCodeLegacyRuntime]: Registering module COBWRITE, from COBWRITE.COB, static memory size 296
[2025-01-17 09:04:40.911] [1] [INFO] [RainCodeLegacyRuntime.Sql]: Create Cursor for Cnd number : 1
[2025-01-17 09:04:40.999] [1] [INFO] [DotNetFHRecordBaseFile]: Using EXTFLH driver fileOrganization=2 access_mode=0 lock_mode=4 other_flags= 0x80 interlanglocking=0 status_file=8
C:\Raincode360\Repositories\Demos\Qix - Legacy with VSAM File Access - Csharp Main\WriteVsam\bin\Debug\net8.0\WriteVsam.exe (process 8464) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

Note:

Ignore the ``RainCodeLegacyRuntime.Exceptions.QIX.ReturnException`` if you encounter it during debugging. This is not a real exception, but it is expected.

C# calling legacy with VSAM access using CICS LINK– How to demonstrate



- Step 7: In the solution explorer, right-click *ReadVsam* project and select *Set as Startup Project*.
- Step 8: Run the *ReadVsam* project by clicking the **play** button on the toolbar; it will read the VSAM file using the key "BE00-1000-0000-9297" and display the file content on the console.

Note:

Before re-running *WriteVsam*, ensure to run the script *CatalogVsam.ps1*; otherwise, you will encounter a *duplicate record QIX exception*.

```
Microsoft Visual Studio Debug Console
[2025-01-17 09:08:26.361] [1] [INFO] [ConfigurationSerializer]: Deserializing file ConfigV2
[2025-01-17 09:08:26.492] [1] [INFO] [ConfigurationSerializer]: Deserializing file ConfigV2 has succeeded.
[2025-01-17 09:08:26.493] [1] [INFO] [ConfigurationSerializer]: Applying default values for fields not present in XML.
[2025-01-17 09:08:26.625] [1] [WARNING] [RaincodeBatch]: Missing registry keys. Configuring with defaults.
[2025-01-17 09:08:28.100] [1] [INFO] [RainCodeLegacyRuntime]: Registering module RC_CHECKHEAP, from RainCodeLegacyRuntime
e.Module.BuiltinModules.CheckHeap, RainCodeLegacyRuntime, Version=5.0.46.0, Culture=neutral, PublicKeyToken=null, static
memory size 0
[2025-01-17 09:08:28.101] [1] [INFO] [RainCodeLegacyRuntime]: Registering module QIXMODULE, from RainCodeLegacyRuntime.Q
IX.QixModule, RainCodeLegacyRuntime, Version=5.0.46.0, Culture=neutral, PublicKeyToken=null, static memory size 0
[2025-01-17 09:08:28.394] [1] [INFO] [RainCodeLegacyRuntime]: Registering module PLIREAD, from PLIREAD.PLI, static memor
y size 8
[2025-01-17 09:08:28.512] [1] [INFO] [DotNetFHRecordBaseFile]: Using EXTfH driver fileOrganization=2 access_mode=0 lock
mode=4 other_flags= 0x80 interlanglocking=0 status_file=8

ACCOUNT_ID: BE00-1000-0000-9297

CUSTOMERID: 00110

AMOUNT: 0000694541

C:\Raincode360\Repositories\Demos\Qix - Legacy with VSAM File Access - Csharp Main\ReadVsam\bin\Debug\net8.0\ReadVsam.ex
e (process 7532) exited with code 0 (0x0).
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

Create REST API from Legacy Program

Create REST API from Legacy Program - Introduction



- The program demonstrates how to create and invoke a REST API on top of the CICS module.
- **Prerequisite:**
 - In the solution explorer, right-click on Legacy project and select Build. This process will perform the following:
 - Compiles the PLI001 and COB004A modules using the Raincode PLI and COBOL compiler.
 - Executes the script *CopyDlls*, which transfers the generated DLLs to the LegacyDll folder of the *RCMainframeAPI* project.

Note:

This step must be repeated whenever there is a change in the PLI001 or COB004A modules.

- Open PowerShell via Tools->Command Line->Developer PowerShell and execute the *Helper.ps1* script. This script performs the following:
 - Generates the helper classes for PLI001 and COB004A modules using the Raincode PLI and COBOL compilers.
 - It also copies the generated *helperclasses* to the **HelperClass** folder of the **RCMainframeAPI** project.

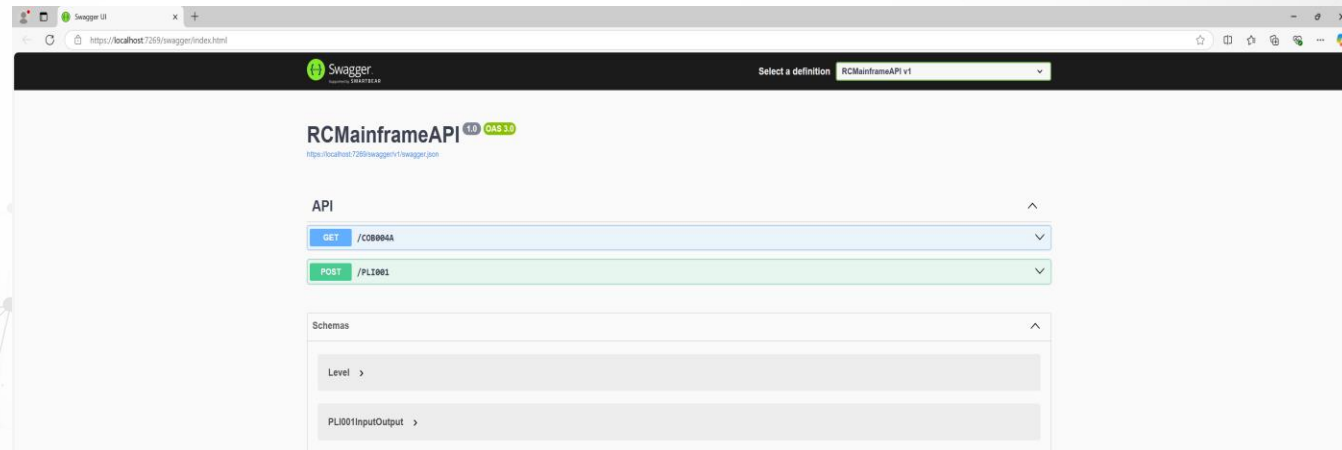
Note:

This script is to be rerun when there is a change in the copybook layout of PLI001 or COB004A modules.

Create REST API from Legacy Program – How to demonstrate



- Step 1: In the solution explorer, right-click the **RCMainframeAPI** project and select *Set as Startup Project*.
- Step 2: Execute the project by pressing *Ctrl+F5* or navigating to the debug menu and selecting *Start Without Debugging*.
- Step 3: A browser with Swagger UI for the GET/COB004A and POST/PLI001 methods will open.



Create REST API from Legacy Program – How to demonstrate



- Click on **Try it out** and enter the following inputs for GET /COB004A method
 - DIRECTION - D
 - SORT - C
 - FIRSTNAME_FILTER - DE
 - LINECOUNT - 15
- Click **Execute**.

The screenshot shows a REST client interface with the following details:

- Request:**
 - Method: GET
 - URL: `https://localhost:7269/COB004A?DIRECTION=D&SORT=C&FIRSTNAME_FILTER=DE&LINECOUNT=15`
 - Headers: `-H 'accept: */*'`
- Response:**
 - Status: 200 OK
 - Content-Type: `text/plain; charset=utf-8`
 - Body (JSON):

```
{  "COB004A_SQLCODE": 0,  "COB004A_FETCHED": 15,  "COB004A_LASTNAME": [    "Mathis",    "Deer",    "Allen",    "Montes",    "Gain",    "Dawson",    "Avery",    "Finley",    "Yu",    "Pars",    "Simon",    "Hensley",    "Park",    "Frederick",    "King"  ],  "COB004A_FIRSTNAME": [    "Demetrius",    "Deidre",    "Deanne",    "Dewitt",    "Deidre",    "Delia"  ]}
```

The response will display a list of customer names that starts with DE.

Create REST API from Legacy Program – How to demonstrate



- Click on **Try it out**, provide the request body as shown in the screenshot for the PLI001 method, and click **Execute**.
- The server response will display the result with value 201 indicating success.

The screenshot displays a REST client interface with the following sections:

- Request body:** A JSON object with the following fields: `"pl01_ACCOUNTNUMBER_FROM": "8000-1000-0000-1438", "pl01_ACCOUNTNUMBER_TO": "8000-1000-0000-2643", "pl01_AMOUNT": 10.5, "pl01_DESCRIPTIONTEXT": "RCHainframeAPI_Swagger", "pl01_SQLCODE": 0, "pl01_ERRTXT": "string"`.
- Execute:** A blue button to execute the request.
- Responses:** A section showing the response details.
 - Code:** 201 (Unauthenticated).
 - Response body:** A JSON object identical to the request body.
 - Response headers:** `content-type: text/plain; charset=utf-8`, `date: Fri, 17 Jan 2025 12:00:27 GMT`, `server: kestrel`.
- Request URL:** `https://localhost:7269/PLI001`.
- Server response:** A section with tabs for Code and Details.
- Responses table:** A table with columns Code, Description, and Links.

Code	Description	Links
200	OK	No links

Create REST API from Legacy Program –Rainbox Generated

Create REST API from Legacy Program – Rainbow Generated



- This demo shows how to generate the REST API from a CICS module with the help of **Rainbox** and invoke it using the Postman API Client.
- **Rainbox** is a tool which is used to generate REST APIs from legacy COBOL and PL/I programs.
- **Steps to generate an API:**
 - Configure rainbow.xml
 - Commarea or Channel mappings
 - Endpoint details (single or multiple)
 - File configuration settings
 - Make sure COBOL and PL/I are compiled successfully, and generated DLLs are available in the assembly path for Rainbow to access.
 - Invoke Rainbow.exe to generate the REST wrapper:
 - `"C:\Program Files\Raincode\Rainbox\rainbox.exe" "-ConfigurationFile=rainbox.xml" "-OutputDir=rest" "-AppNamespace=Rainbox.Demo" "-AssemblyPath=.\\output" "-SourcePath=." "-LogLevel=TRACE"`
 - Rainbow looks for the DB connection string (`RAINBOX_DB_CONNECTION_STRING`) from the environment variable.



- Steps:
 - Run the script **install_postman.ps1** to install Postman, which will be used to call the API (this is a one-time process)
 - Execute the **run.ps1** script, which performs the following steps:
 - Invokes the Raincode's COBOL compiler to compile the COBOL programs COBACCT and COBCUST; generating DLL files in the **output** directory
 - Invokes the Rainbow by providing the configuration file **rainbox.xml** to generate a REST API containing multiple endpoints (/COBCUST and /COBACCT)
 - The generated API project (**Box.csproj**) will be created in the **rest** directory.
 - Builds and launches the REST API, which will be running at Localhost:5000



- Trigger a series of requests from Postman as follows
- COBCUST:
 - GET request to the endpoint *http://Localhost:5000/COBCUST/Schema* to understand the schema
 - POST request to the endpoint *http://Localhost:5000/COBCUST* with the following body to read the customer details
 - POST request to the endpoint *http://Localhost:5000/COBCUST* with the following body to update the customer's first name

```
{
  "COBCUST-IN": {
    "COBCUST-ACTION": "G",
    "COBCUST-CUSTOMERID": 101,
    "COBCUST-FILTER" : "C"
  }
}
```

```
{
  "COBCUST-IN": {
    "COBCUST-ACTION": "U",
    "COBCUST-CUSTOMERID": 101,
    "COBCUST-FILTER" : "C",
    "COBCUST-FIRSTNAME": "Emma-Test"
  }
}
```

POST COBCUST (read the customer details)



POST http://localhost:5000/C

HTTP

http://localhost:5000/COBCUST

Save

</>

POST

http://localhost:5000/COBCUST

Send

Body

raw

JSON

Beautify

```
1 {
2   "COBCUST-IN": {
3     "COBCUST-ACTION": "G",
4     "COBCUST-CUSTOMERID": 101,
5     "COBCUST-FILTER": "C"
6   }
7 }
```

Body

Pretty

JSON

200 OK 40 ms 687 B

Save Response

```
12 "COBCUST-OUT": {
13   "COBCUST-OUT-RETURNCODE": 0,
14   "COBCUST-OUT-MESSAGE": "",
15   "COBCUST-OUT-FETCHED": 1,
16   "COBCUST-OUT-CUSTOMERID": [
17     101,
18     0,
19     0,
20     0,
21     0
22   ],
23   "COBCUST-OUT-LASTNAME": [
24     "Ritter",
25     "",
26     "",
27     "",
28     ""
29   ],
30 }
```



- COBACCT:

- GET request to the endpoint *http://Localhost:5000/COBACCT/Schema* to understand the schema

- POST request to the endpoint *http://Localhost:5000/COBACCT* with the following body to read the account details

```
{|
  "COBACCT-IN": {
    "IN-COBACCT-REQ-TYPE": "G",
    "IN-COBACCT-CUST-ID": 101
  }
}
```

- POST request to the endpoint *http://Localhost:5000/COBACCT* with the following body to update the amount of the account

```
{
  "COBACCT-IN": {
    "IN-COBACCT-REQ-TYPE": "U",
    "IN-COBACCT-CUST-ID": 101,
    "IN-COBACCT-AMOUNT": 5.00,
    "IN-COBACCT-ACCT-NO": "BE00-2000-0038-0073"
  }
}
```

POST COBACCT (update the amount of the account)



POST http://localhost:5000/COBACCT

HTTP http://localhost:5000/COBACCT

POST http://localhost:5000/COBACCT

Send

Body raw JSON Beautify

```
1 {
2   "COBACCT-IN": {
3     "IN-COBACCT-REQ-TYPE": "U",
4     "IN-COBACCT-CUST-ID": 101,
5     "IN-COBACCT-AMOUNT": 5.00,
6     "IN-COBACCT-ACCT-NO":
7       "BE00-2000-0038-0073"
8   }
9 }
```

Body Pretty JSON Save Response

200 OK 68 ms 502 B

Save Response

200 OK 68 ms 502 B

Save Response

4
5 "IN-COBACCT-ACCT-NO":
6 "BE00-2000-0038-0073",
7 "IN-COBACCT-CUST-ID": 101,
8 "IN-COBACCT-AMOUNT": 5.00
9 },
10 "COBACCT-OUT": {
11 "OUT-COBACCT-ERRCODE": 0,
12 "OUT-COBACCT-MSG": "ACCOUNT UPDATE IS
13 SUCCESSFUL",
14 "OUT-COBACCT-FETCHED": 0,
15 "OUT-COBACCT-ACCT-NO": [
16 "",
17 "",
18 "",
19 ""
20],
21 "OUT-COBACCT-CUST-ID": [

RAIN CODE 360
Advanced Showcase

Online Internal Reader



- This program demonstrates how a Batch can be started from an online CICS application using SPOOL commands
- It highlights the following:
 - Providing a custom QixFactory to the QIX Emulator
 - Utilization of CICS commands, such as CICS SPOOLOPEN, SPOOLWRITE, and SPOOLCLOSE
 - Utilization of BMS maps
 - Execution of a JCL from a CICS application
- The CICS writes a file in a designated folder
- A watchdog picks up the file and starts a JCL job
- The generated JCL filters the customers from the database based on birth years within a specified range (start date and end date) and writes them into the dataset (UNLOAD.CUST.SEQ)



Online Internal Reader- Steps to run

- Open Powershell via Tools->Command Line-> Developer PowerShell
- Execute the *run.ps1* script and wait for the QIX windows and the 3270 emulator to launch. The bank application will start automatically

The screenshot displays a Visual Studio Code interface with a Developer PowerShell terminal window at the top. The terminal shows the execution of a script that configures a QIX environment. The output includes various log messages, warnings, and information about the QIX configuration, such as the region name, transaction options, and the addition of QIX transactions. Below the terminal, two application windows are visible: 'Raincode Bank' and 'CustomSpool.cs'. The 'Raincode Bank' window shows a list of transactions, and the 'CustomSpool.cs' window shows a list of references.

```
PS C:\Raincode360\Repositories\Demos\Qix - Emulator - Online Intrdr> ./run
[2025-02-17 06:05:02.394] [WARNING] : 'ServiceBrokerConnectionString' not set, using 'QixConn
[2025-02-17 06:05:02.422] [WARNING] : 'ServiceBrokerConnectionString' not set, using 'Ser
[2025-02-17 06:05:02.422] [WARNING] : 'TSQueuesConnectionString' not set, using 'QixConnec
[2025-02-17 06:05:04.496] [INFO] : Found DB version 2.
[2025-02-17 06:05:04.975] [INFO] : No region named BANKDEMO found
[2025-02-17 06:05:05.017] [INFO] : Added Region BANKDEMO
[2025-02-17 06:05:05.032] [INFO] : Added QIX Region Option SYSID -> CICS
[2025-02-17 06:05:05.044] [INFO] : Added QIX Region Option APPLID -> CICSAP
[2025-02-17 06:05:05.073] [INFO] : Added QIX transaction C004 -> COB004
[2025-02-17 06:05:05.075] [INFO] : Added QIX transaction C005 -> COB005
[2025-02-17 06:05:05.076] [INFO] : Added QIX transaction C006 -> COB006
[2025-02-17 06:05:05.077] [INFO] : Added QIX transaction C007 -> COB007
[2025-02-17 06:05:07.412] [1] [TRACE] [RainCode.Core.CommandLine]: -ConfigConnectionSt
[2025-02-17 06:05:07.444] [1] [WARNING] [QIX]: Creating log file at C:\Users\rngb\AppData
[2025-02-17 06:05:07.449] [1] [TRACE] [RainCode.Core.AssemblyLoading]: Adding path 'C:
[2025-02-17 06:05:08.165] [1] [TRACE] [RaincodeLegacy.LicenseKey]: Raincode Legacy Lic
[2025-02-17 06:05:08.168] [1] [TRACE] [RaincodeLegacy.LicenseKey]: Raincode Legacy Lic
[2025-02-17 06:05:08.516] [1] [TRACE] [RaincodeLegacy.LicenseKey]: Raincode Legacy Lic
```

-
- ```

** Visual Studio 2022 Developer PowerShell v17.12.1
** Copyright (c) 2022 Microsoft Corporation
**
PS C:\Raincode360\Repositories\Demos\Qix - Emulator - Online Intrdr> .\run3270
PS C:\Raincode360\Repositories\Demos\Qix - Emulator - Online Intrdr>

Raincode Bank
File Options Keypad

Customer Report per Year

From Year.....: 1940
To Year.....: 2000

F3=Cancel F5=Confirm
4B DX4T 009/038

```

# Raincode QIX Emulator - CWA

# Raincode QIX Emulator – Common Work Area – Purpose and Concept



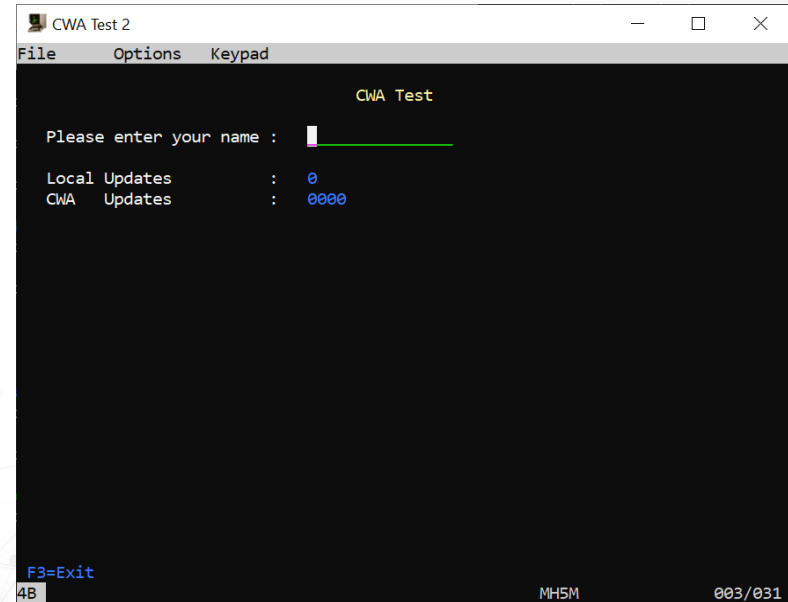
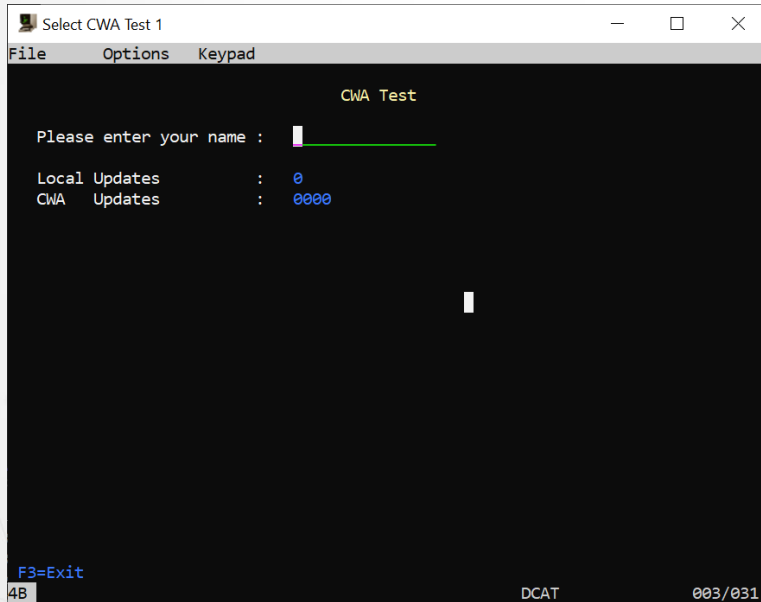
- Raincode's CICS implementation supports shared memory through the Common Work Area (CWA). CICS provides access to a CWA by returning a pointer to it when invoking the ADDRESS CWA command.
- However, in QIX, programs from the same region may run on different processing servers or even on separate machines, which makes direct memory sharing impractical.
- To solve this problem, the contents of the CWA are stored in SQLServer and are loaded into memory at the beginning of the transaction. Any updates done to it during the lifetime of the transaction are written back into the database when the transaction finishes.
- The idea is to run a single transaction that both reads and writes the CWA twice in two separate emulator windows.

*To keep things simple, we limit ourselves to a single processing server and a single terminal server running on the same machine.*

# Raincode QIX Emulator – Running the demo



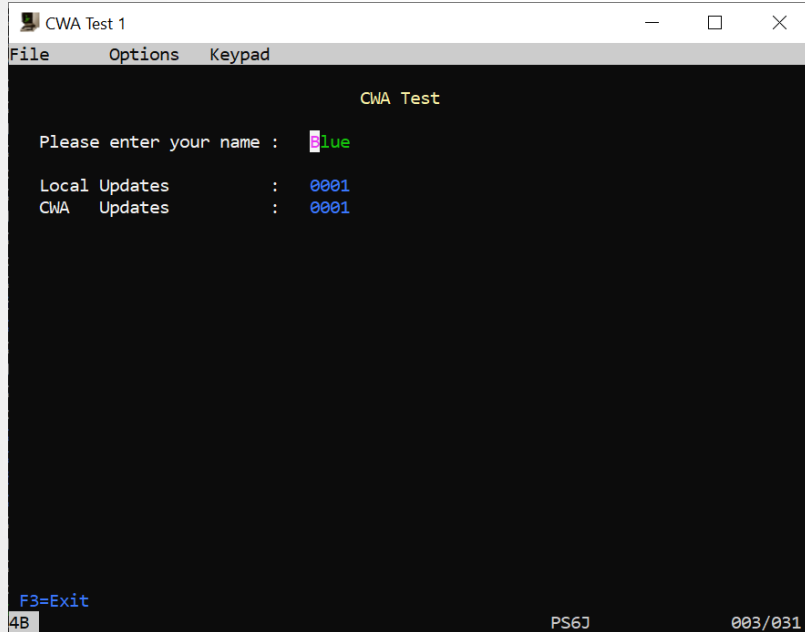
- Run the *run.ps1* PowerShell script from any directory.
- When the two terminal windows appear, type **CWA1** in each of them.



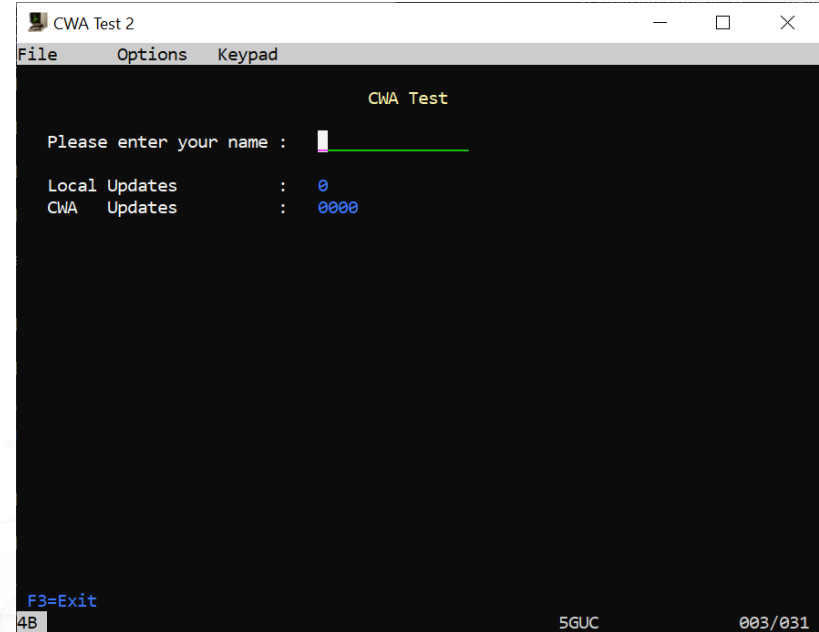
# Raincode QIX Emulator – Running the demo



- In the **CWA Test 1** window, type any sequence of characters and press Enter.



The screenshot shows a terminal window titled "CWA Test 1" with a menu bar containing "File", "Options", and "Keypad". The main display area shows the text "CWA Test" at the top. Below it, a prompt "Please enter your name :" is followed by the input "blue" in green. Further down, two lines of status information are displayed: "Local Updates : 0001" and "CWA Updates : 0001". At the bottom left, a status bar shows "F3=Exit" and "4B". At the bottom right, it shows "PS6J" and "003/031".

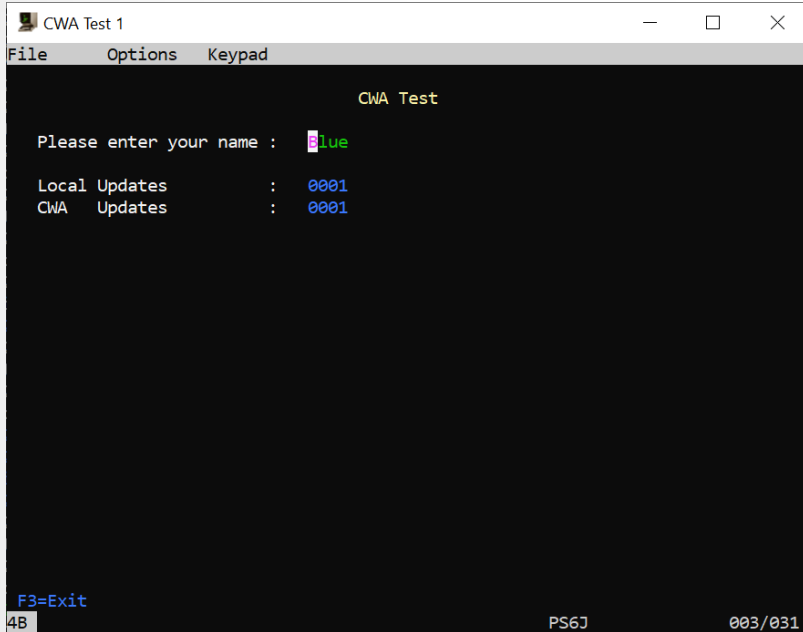


The screenshot shows a terminal window titled "CWA Test 2" with a menu bar containing "File", "Options", and "Keypad". The main display area shows the text "CWA Test" at the top. Below it, a prompt "Please enter your name :" is followed by a green cursor line. Further down, two lines of status information are displayed: "Local Updates : 0" and "CWA Updates : 0000". At the bottom left, a status bar shows "F3=Exit" and "4B". At the bottom right, it shows "5GUC" and "003/031".

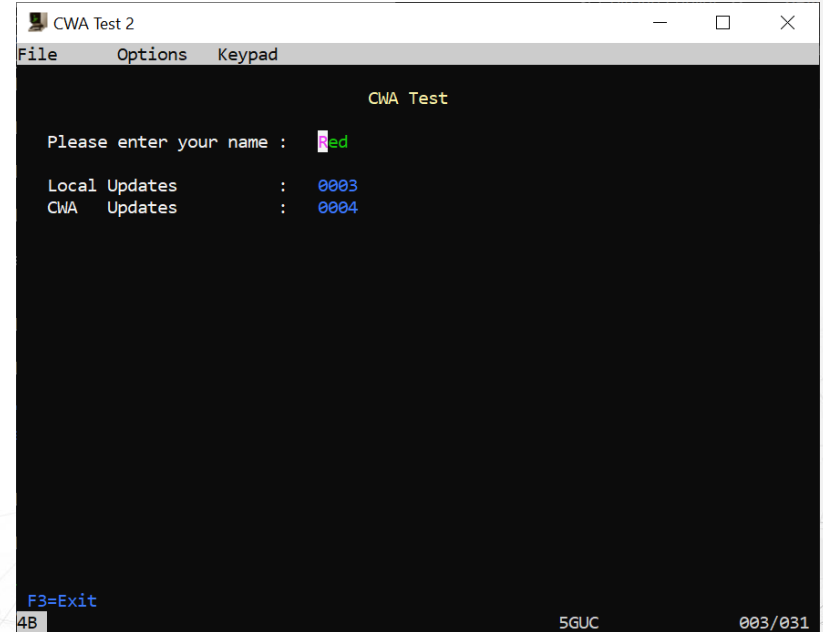
# Raincode QIX Emulator – Running the demo



- In the **CWA Test 2** window, press Enter and then type something.



A screenshot of a terminal window titled "CWA Test 1". The window has a menu bar with "File", "Options", and "Keypad". The main display area shows the text "CWA Test" at the top. Below it, it says "Please enter your name : " followed by the word "blue" in green. Further down, it shows "Local Updates : 0001" and "CWA Updates : 0001". At the bottom left, it says "F3=Exit" and "4B". At the bottom right, it says "PS6J" and "003/031".

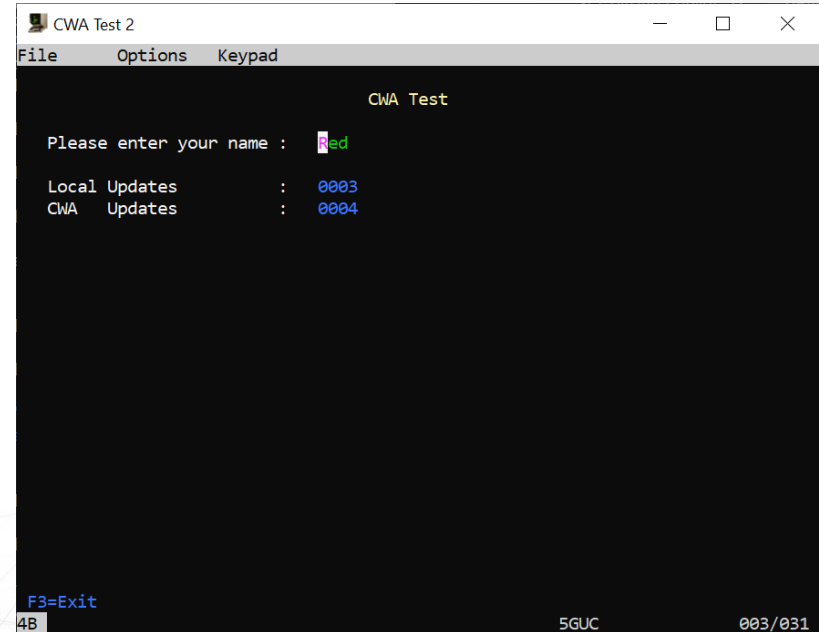
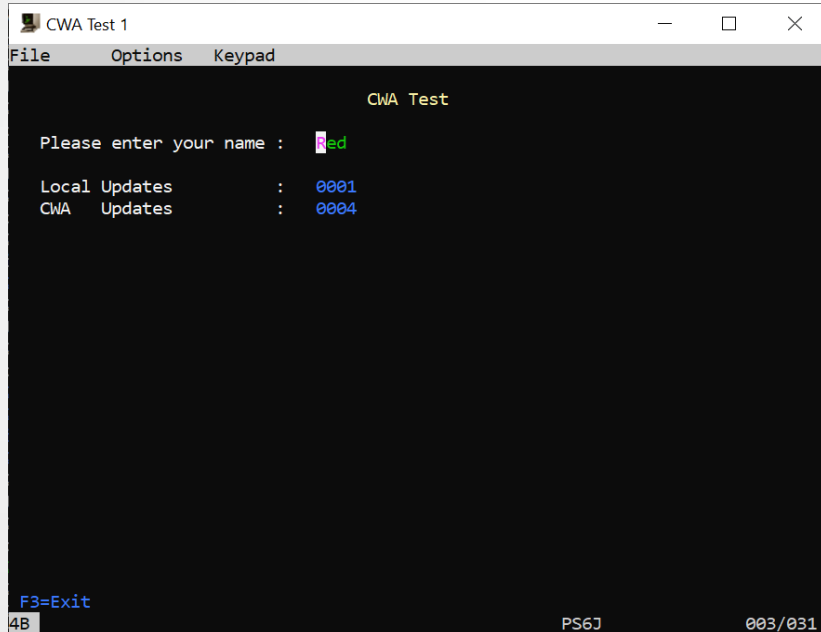


A screenshot of a terminal window titled "CWA Test 2". The window has a menu bar with "File", "Options", and "Keypad". The main display area shows the text "CWA Test" at the top. Below it, it says "Please enter your name : " followed by the word "red" in green. Further down, it shows "Local Updates : 0003" and "CWA Updates : 0004". At the bottom left, it says "F3=Exit" and "4B". At the bottom right, it says "5GUC" and "003/031".

# Raincode QIX Emulator – Running the demo



- When you return to the **CWA Test 1** window, the value entered in the **CWA Test 2** window will be displayed

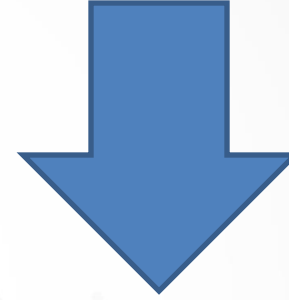


- Demos:
  - [SQL Level Caching](#)
  - [Program Caching](#)
  - [Standalone SQL Conversion](#)
  - [SQL Caching - Singleton Statements](#)

# Caching – Introduction



- Keep a copy of slow-moving data closer to the application
- Can be in-process, in-machine, or even on a separate machine
  - Provided it is closer and faster than the original data provider (database)



- A reimplementation of a (simpler) database
- Even if slow-moving, caches can be invalidated
- Deciding that something is slow-moving is application-dependent
- Changes to the application

# SQL Level Caching

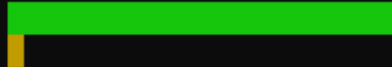


- Test the performance of SQL Level Caching
- Illustrates the use of static SQL
- Type `.\run` from the powershell command line to execute the demo
- SQL Caching Benchmark

```
Test 1: Static SQL with no caching
Test 2: Static SQL with SQL caching
```

## ELAPSED TIMES

| Test   | SQLs<br>Total | SQLs<br>Cached | Time<br>(s) |
|--------|---------------|----------------|-------------|
| Test 1 | 500           | 0              | 6.08        |
| Test 2 | 500           | 400            | 0.26        |



The use of static SQL requires that the COBOL program names are unique for the database that they connect to.

When the cache is active, the first 50 SQLs will connect to the database to fill up the cache.

Test 1 and Test 2 execute the same compiled code. Only difference is Test 2 uses the **-plugin** argument of `rclrun`.

When launching the program from the menu, it will run in the cached mode.

# Program Caching



- Test the performance of Program Caching
- Illustrates the use of static SQL
- Caching Benchmark

```
Test 1: Static SQL with no caching
Test 2: Static SQL with program caching
```

ELAPSED TIMES

| Test   | SQLs Total | SQLs Cached | Time (s) |
|--------|------------|-------------|----------|
| Test 1 | 500        | 0           | 5.90     |
| Test 2 | 500        | 400         | 0.27     |



The use of static SQL requires that the COBOL program names are unique for the database that they connect to.

When the cache is active, the first 50 SQLs will connect to the database to fill up the cache.

Test 1 and Test 2 execute the same compiled code. Only difference is Test 2 uses the **-plugin** argument of rclrun.

When launching the program from the menu, it will run in the cached mode.

# SQL Caching – Singleton Statements – Introduction



- The following pattern:

```
SELECT F1, F2, F3
FROM T
INTO :HT1, :HT2, :HT3
WHERE K1=:HK1 AND
 K2=:HK2 AND
 K3=:HK3
```

- VSAM-like use of SQL
- Probably cacheable
  - Chatty (one record at most)
  - Read (as opposed to write)
  - Simple criterion (typical of a primary key)

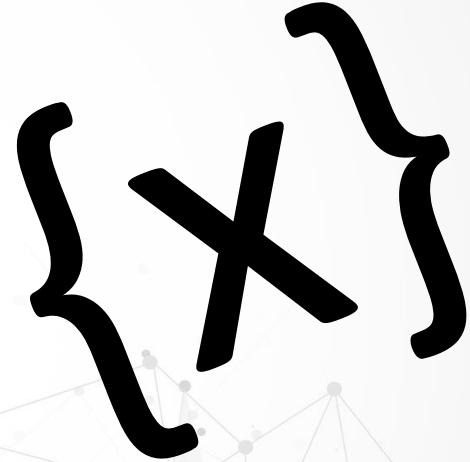
- Recognizing that a SQL statement that is cacheable is not trivial
  - SQL is way too versatile for comfort
  - Even more so if the analysis must happen at runtime
- But our compilers perform this at compile-time
- At runtime, it is merely an attribute to consult
  - Together with the table, source fields, target fields

$(K1, K2, K3) \xrightarrow{T} (F1, F2, F3)$

# How to implement a cache for singletons?



- When executing a SQL statement:
  - Check for the singleton attribute
  - Ensure that the table is cacheable (slow-moving)
  - Ensure that the Kn tuple defines a primary key
- If all these conditions are fulfilled
  - Redirect the query to the cache
- In all other cases, revert to default behaviour
  - Send the actual statement to the database
- We do not supply an actual cache implementation!
  - Many different design decisions to make



# Benefits



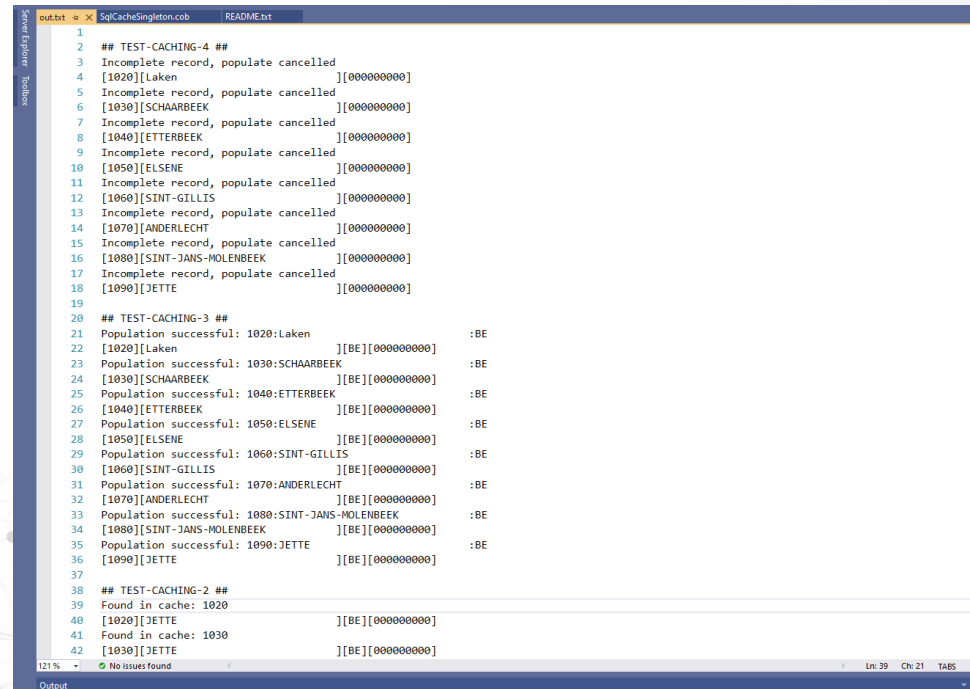
- No change to the application code
  - The version running on the mainframe remains unaltered
- Database agnostic
  - SQL Server or DB2 (through HIS)
- Scalability
  - One must visit tables, not statements
- Information present in the repository

| START_LINE | RC_END_LINE | RC_START_COL | RC_END_COL | RC_COMMAND | RC_USED | RC_SINGLETON |
|------------|-------------|--------------|------------|------------|---------|--------------|
| 13         | 38          | 20           | 11         | DECLARE    | Y       | N            |
| 13         | 109         | 20           | 11         | DECLARE    | Y       | N            |
| 16         | 33          | 20           | 11         | DECLARE    | Y       | N            |
| 6          | 117         | 20           | 11         | DECLARE    | Y       | N            |
| 17         | 76          | 20           | 11         | DECLARE    | Y       | N            |
| 792        | 797         | 15           | 44         | SELECT     | Y       | Y            |
| 907        | 912         | 14           | 49         | SELECT     | Y       | Y            |
| 954        | 958         | 14           | 50         | SELECT     | Y       | Y            |
| 1220       | 1240        | 15           | 46         | SELECT     | Y       | Y            |
| 1278       | 1284        | 15           | 45         | SELECT     | Y       | Y            |
| 1650       | 1659        | 14           | 66         | DECLARE    | Y       | N            |
| 1674       | 1674        | 15           | 20         | OPEN       | Y       | N            |

# SQL Caching – Singleton Statements



- Demonstrate SQL Caching via a plugin
- Demonstrate that statements that do not retrieve all the columns of the table should not populate the cache
- Illustrates the use of static SQL
- After the script is executed, it will write the output to a file named **out.txt**



```
1 ## TEST-CACHING-4 ##
2 Incomplete record, populate cancelled
3 [1020][Laken]][000000000]
4 Incomplete record, populate cancelled
5 [1030][SCHAARBEEK]][000000000]
6 Incomplete record, populate cancelled
7 [1040][ETTERBEEK]][000000000]
8 Incomplete record, populate cancelled
9 [1050][ELSENE]][000000000]
10 Incomplete record, populate cancelled
11 [1060][SINT-GILLIS]][000000000]
12 Incomplete record, populate cancelled
13 [1070][ANDERLECHT]][000000000]
14 Incomplete record, populate cancelled
15 [1080][SINT-JANS-MOLENBEEK]][000000000]
16 Incomplete record, populate cancelled
17 [1090][JETTE]][000000000]
18
19
20 ## TEST-CACHING-3 ##
21 Population successful: 1020:Laken :BE
22 [1020][Laken]][BE][000000000] :BE
23 Population successful: 1030:SCHAARBEEK :BE
24 [1030][SCHAARBEEK]][BE][000000000] :BE
25 Population successful: 1040:ETTERBEEK :BE
26 [1040][ETTERBEEK]][BE][000000000] :BE
27 Population successful: 1050:ELSENE :BE
28 [1050][ELSENE]][BE][000000000] :BE
29 Population successful: 1060:SINT-GILLIS :BE
30 [1060][SINT-GILLIS]][BE][000000000] :BE
31 Population successful: 1070:ANDERLECHT :BE
32 [1070][ANDERLECHT]][BE][000000000] :BE
33 Population successful: 1080:SINT-JANS-MOLENBEEK :BE
34 [1080][SINT-JANS-MOLENBEEK]][BE][000000000] :BE
35 Population successful: 1090:JETTE :BE
36 [1090][JETTE]][BE][000000000] :BE
37
38 ## TEST-CACHING-2 ##
39 Found in cache: 1020
40 [1020][JETTE]][BE][000000000]
41 Found in cache: 1030
42 [1030][JETTE]][BE][000000000]
```

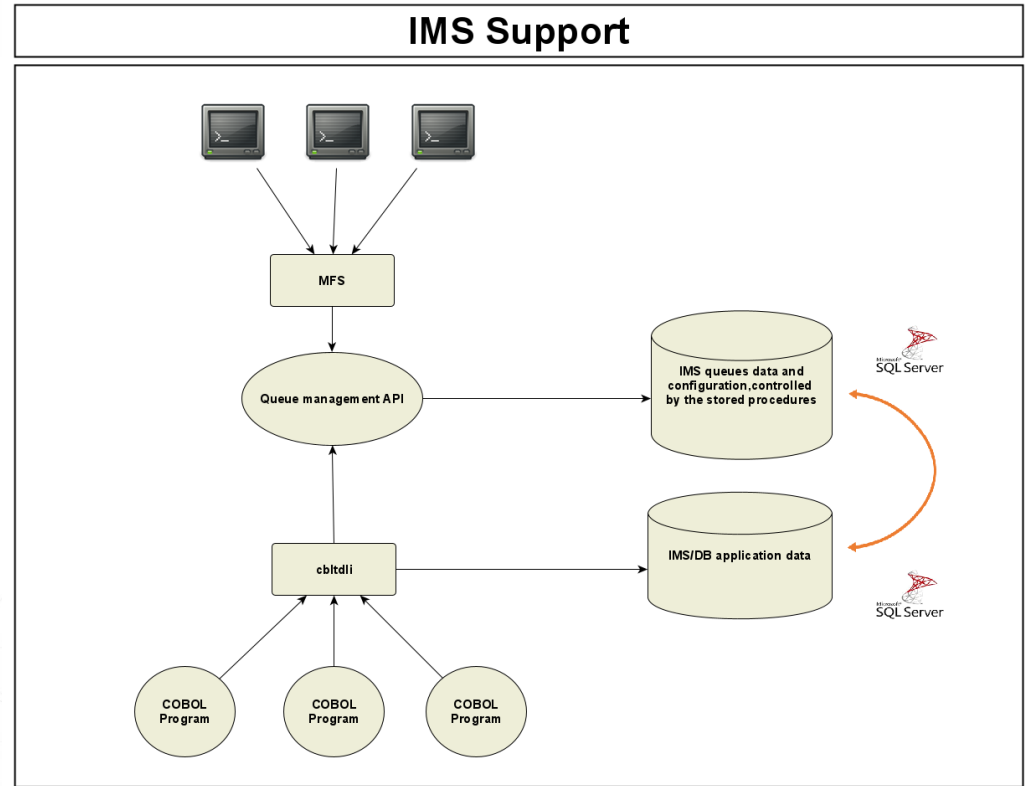


# IMSql

# IMSql – Introduction



- IMSql rehosts IMS/DB and IMS/TM systems on .NET and SQL Server. Mainframe applications compiled for .NET can interact with IMSql in the same manner as they would interact with IMS.
- IMSql is non-transformational, which means it keeps the sources (COBOL, PL/I) as-is with the IMS-specific CBLTDLI, PLITDLI, and EXEC DLI calls unaltered. This way, new sources are not introduced, which plays a crucial role in program maintainability.





## ■ IMS/DB

### ■ Data Migration

- The new DB is created using the DBD's
- A table is created for each physical segment
- Data is not transformed; IMS segments are stored as raw data

## ■ Program compilation

- Entity Framework *DbContext* is generated for each DBD
- Untransformed programs (COBOL and PL/I) are compiled as usual
- The *rcpsb* utility translates the PSB into XML files that are loaded at runtime

## ■ IMS/TM

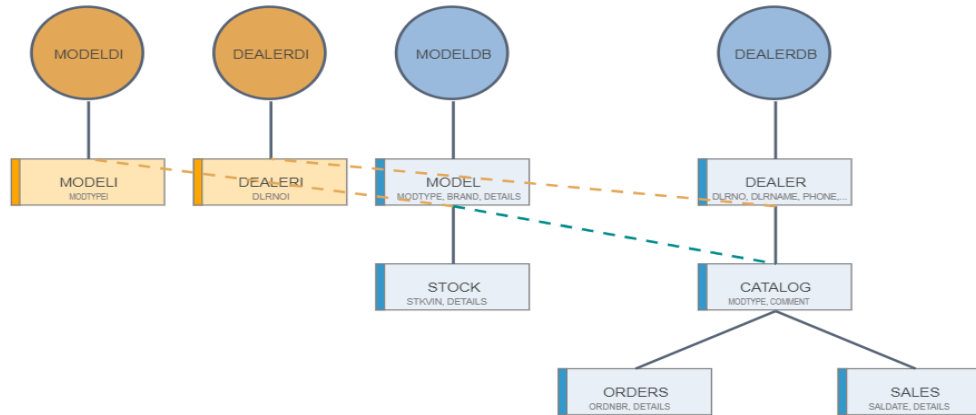
- The IMS/TM is translated using SQL Server Queues and Messages
- The *rcmfs* utility translates the MFS files into XML files that are loaded at runtime



- [Introduction](#)
- [Batch](#)
- [Online](#)
- [Checkpoint/Restart](#)

# IMSql – Demo – Introduction

- For the demo we use two DBDs (MODELDB and DEALERDB)
  - In the graph below, DBDs are presented by circle and segments by rectangle. Physical links by plain lines, logical links by dotted lines
- Each DBD has a primary index, respectively MODELDI and DEALERDI
- DEALERDB.CATALOG is a logical child of MODELDB.MODEL

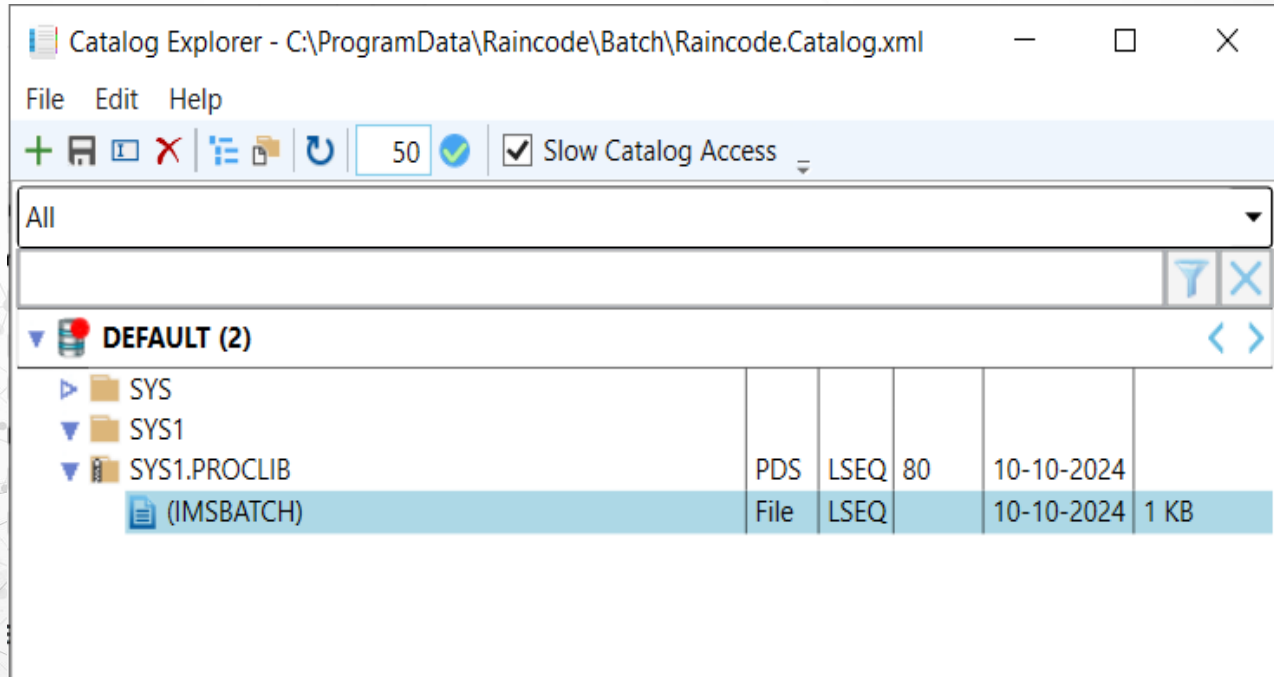


# IMSql – Demo – Batch



- Step 1: Open the solution from the folder IMSql\DealerBatch
- Step 2: Make 'ADDPROC' as the startup project by right clicking on the project and selecting 'Set as Startup Project'
- Step 3: Click on 'Start Raincode Debugger', this creates the PROC IMSBATCH in the catalog, which is one time task

## Expected Result:



The screenshot shows the 'Catalog Explorer' window for 'C:\ProgramData\Raincode\Batch\Raincode.Catalog.xml'. The interface includes a menu bar (File, Edit, Help), a toolbar with icons for file operations and a refresh button, a status bar showing '50' and 'Slow Catalog Access', and a search bar. The main pane displays a tree view of the catalog structure. Under 'DEFAULT (2)', there are folders for 'SYS' and 'SYS1'. Inside 'SYS1', there is a folder 'SYS1.PROCLIB' which contains a procedure '(IMSBATCH)'. The procedure details are shown in a table below the tree view.

| Object Name | Type | Format | Pages | Created    | Size |
|-------------|------|--------|-------|------------|------|
| (IMSBATCH)  | File | LSEQ   | 80    | 10-10-2024 | 1 KB |

# IMSql – Demo – Batch

- Step 1: Open the solution from the folder IMSql\DealerBatch
- Step 2: Make 'BESTSALE' as the startup project by right clicking on the project and selecting 'Set as Startup Project'
- Step 3: Click on 'Start Raincode Debugger', this executes BESTSALE.JCL and the program BESTSALE.cbl searches for the DEALER with the biggest number of SALES segments

- **Expected Result:**

Job must execute successfully and  
SYSOUT Should have this information

```
NBR RECORD : 00041
BEST DEALER : DEAL00030 / 00011
END TIME 2024/10/10 14:28:29:83
final PCB STATE:
PCB STATE :
DB-LEVEL : 0
DB-KFBA : 000000 ->
DB-SEGMENT :
DB-STAT : GB
DB-PROC : GIRD
```



- Step 1: Open the solution from the folder IMSql\DealerBatch
- Step 2: Make 'EXTRACT-STAT' as the startup project by right clicking on the project and selecting 'Set as Startup Project'
- Step 3: Click on 'Start Raincode Debugger', this executes EXTRACT-STAT.JCL and the program DLRSTAT.cbl counts the number of segments of each type for the first 101 segments of the DBD

- **Expected Result:**

Job must execute successfully and  
SYSOUT Should have this information

```
: NBR RECORD : 00101
: NBR DEALER : 00022
: NBR CATALOG : 00042
: NBR ORDERS : 00018
: NBR SALES : 00019
: END TIME 2024/10/10 14:17:01:77
: final PCB STATE:
: PCB STATE :
: DB-LEVEL : 3
: DB-KFBA : 00050 -> DEAL00010 CARV00010
: DB-SEGMENT : ORDERS
: DB-STAT :
: DB-PROC : GIRD
```

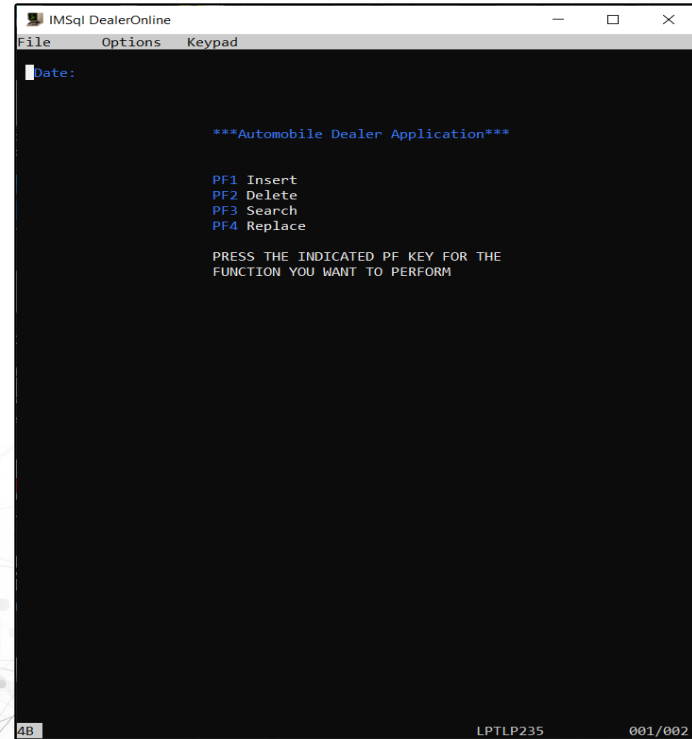


- The IMS/TM is translated using SQL Server Queues and Messages
- This application search/insert DEALER, CATALOG segments
- COBOL/PLI programs using call CBLTDLI, PLITDLI and EXEC DLI

# IMSql – Demo – Online



- Step 1: Double-click on “**Start IMSql 3270 demo**” shortcut placed on the desktop
  - Starts two servers: terminal server and processing server
  - Starts WC3270: 3270 Emulator
- Step 2: Enter username and password (demo/demo)
- Step 3: Enter the command “/FOR OMMENU”
- Step 4: Use function keys to navigate to the different screens



The transactions details can also be seen through [Raincode console](#)



## Possible values for the online screens

- Insert (PF1)
  - Insert a new Dealer (PF2)
    - Insert new values
  - Insert a new Model (PF3)
    - Insert new values
- Delete (PF2)
  - Delete a dealer (PF2)
    - DDDD00005
  - Delete a model (PF3)
    - BMWV01323
- Search (PF3)
  - Search for a dealer (PF2)
    - ID Number = DDDD00001
  - Search for a model (PF3)
    - Model Type = CARV00001
  - Display catalogs (PF4)
    - ID Number = DDDD00050
- Replace (PF4)
  - Update a model (PF4)
    - Type = CARV00004
    - Brand = <new data>
    - Details = <new data>

# IMSql – Online (Raincode Console)

## Terminals

| Name     | User   | Host           | Conversation                                         | ConnectionTime      |
|----------|--------|----------------|------------------------------------------------------|---------------------|
| LPTLP207 | NOBODY | raincode360-vm | <a href="#">b1e732a7-849f-ec11-b402-002248584216</a> | 09/03/2022 08:41:04 |
| LPTLP229 | NOBODY | raincode360-vm | <a href="#">7fb88d4a-cd9e-ec11-b402-002248584216</a> | 08/03/2022 10:48:30 |
| LPTLP235 | NOBODY | raincode360-vm | <a href="#">2dbd91c4-5ba3-ec11-b404-002248584216</a> | 14/03/2022 05:58:28 |
| LPTLP365 | NOBODY | raincode360-vm | <a href="#">9f550257-969b-ec11-b401-002248584216</a> | 04/03/2022 08:37:35 |
| LPTLP456 | NOBODY | raincode360-vm | <a href="#">ada99bf7-959f-ec11-b402-002248584216</a> | 09/03/2022 10:45:00 |

LPTLP235 is the same terminal (at the bottom of the 3270 emulator) which starts when you run the [IMSql online demo](#).

# IMSql – Online (Raincode Console)

## Conversations

| conversation_handle                  | local_service                                            | far_service                                              | state_desc           |           |
|--------------------------------------|----------------------------------------------------------|----------------------------------------------------------|----------------------|-----------|
| bc16cf0d-969f-ec11-b402-002248584216 | imsq://DEALER/TRANS/dinsert                              | https://imsq.raincode.com/IMS/D...<br>vm/TerminalService | CONVERSING           | ENDDIALOG |
| b916cf0d-969f-ec11-b402-002248584216 | https://imsq.raincode.com/IMS/D...<br>vm/TerminalService | imsq://DEALER/TRANS/dinsert                              | CONVERSING           | ENDDIALOG |
| ef75ed5b-c89e-ec11-b402-002248584216 | https://imsq.raincode.com/IMS/D...                       | https://imsq.raincode.com/IMS/D...<br>vm/TerminalService | DISCONNECTED_INBOUND | ENDDIALOG |
| a2550257-969b-ec11-b401-002248584216 | https://imsq.raincode.com/IMS/D...                       | https://imsq.raincode.com/IMS/D...<br>vm/TerminalService | DISCONNECTED_INBOUND | ENDDIALOG |
| 8f1793f3-689f-ec11-b402-002248584216 | imsq://DEALER/TRANS/dinsert                              | https://imsq.raincode.com/IMS/D...<br>vm/TerminalService | CONVERSING           | ENDDIALOG |
| 8c1793f3-689f-ec11-b402-002248584216 | https://imsq.raincode.com/IMS/D...<br>vm/TerminalService | imsq://DEALER/TRANS/dinsert                              | CONVERSING           | ENDDIALOG |

Details of the conversation: When you choose to insert a new dealer.



## Message Queues

### Queues

[QueryNotificationErrorsQueue](#) (0)

[EventNotificationErrorsQueue](#) (0)

[ServiceBrokerQueue](#) (0)

[TerminalServer](#) (0)

[ProcessingServer](#) (0)

[IMS\\_PROC\\_DEALER](#) (1)

[IMS\\_DEALER\\_raincode360-vm](#) (2)

[IMS\\_DEALER](#) (0)

When you click on [IMS\\_DEALER\\_raincode360-vm \(2\)](#), it will take you to the details of the pending messages

### Messages

| message_enqueue_time    | service_name                                                         | service_contract_name     | message_type_name |
|-------------------------|----------------------------------------------------------------------|---------------------------|-------------------|
| 2022-03-08T08:46:46.853 | https://imsql.raincode.com/IMS/DEALER/raincode360-vm/TerminalService | ProcessingServiceContract | ErrorMessage      |
| 2022-03-14T05:26:45.967 | https://imsql.raincode.com/IMS/DEALER/raincode360-vm/TerminalService | ProcessingServiceContract | OutputMessage     |



- The alternate PCB can have as output message destination
  - transaction code names (NAME=)
  - logical terminal names (LTERM=)
- In both cases, IMSql uses a program as a destination
  - In COBOL or PLI
  - In C#
- All programs use call CBLTDLI, PLITDLI and EXEC DLI as normal

# IMSql – Demo – Online – Alternate PCB



- Step 1: Double-click on the “**Start IMSql 3270 demo**” shortcut placed on the desktop
  - Starts two servers: terminal server and processing server
  - Starts WC3270: 3270 Emulator
- Step 2: Enter username and password (demo/demo)
- Step 3: Enter the command “/FOR RECMIN”
- Step 4: Enter a target destination number and some Input text.
- Step 5: The mirrored text is returned as Output.

The screenshot shows a terminal window titled "IMSql DealerOnline" with a menu bar containing "File", "Options", and "Keypad". The terminal content displays a demo session:

```
*** IMS TM online ALT PCB demo ***
Target: (1,2,3) 3
Input: Message 4 you, Sir!
Output:
```

At the bottom of the terminal window, the status bar shows "4B", "LPTLP640", and "007/037".



Target destinations are:

1. A COBOL program logger.cbl registered as a transaction
  - Calls CBLTDLI to get the message and does a DISPLAY of the contents
  - Requires a PSB (LOGGER00)
2. A COBOL program termlogger.cbl registered as a terminal
  - Fundamentally, it is the same code as the logger.cbl
  - Does **not** require a PSB since it is a terminal
- A C# program logger.cs registered as a terminal
  - Construct a call to CBLTDLI as if it were a COBOL program
  - Does **not** require a PSB since it is a terminal

# IMSql – Demo – Checkpoint/Restart



- To demonstrate the checkpoint/restart, we use a program that
  - Reads command from a GSAM file (RCD001GI)
  - Writes a log into a GSAM file (RCD001GO)
  - Set a checkpoint after every 2 commands. Checkpoint are named “D0100002”, “D0100004”, ...
  - The commands can be
    - ADD = insert entry in IMS DB
    - DELETE = delete entry from IMS DB
    - UPDATE = update entry from IMS DB
    - DISPLAY = display entry
    - TADD = same as ADD, but write to console
    - ABEND = If in restart mode, do nothing else - execute a “divided by 0” to abort the program



- The command file used

- ```
ADD      LAST7      BATCHADD  8-111-7777D07/R07      0001000
DISPLAY  LAST7                                     0002000
checkpoint "D0100002" here
ADD      LAST7B      BATCHUPD                                     0003000
DISPLAY  LASTA                                     RESTART
checkpoint "D0100004" here
TADD     LAST7C      BATCHUPD  8-111-7777D07/R07      0004000
ABEND                                         0004100
checkpoint "D0100006" here
DISPLAY  LAST7B                                     0005000
DELETE   LAST7                                     0006000
checkpoint "D0100008" here
DISPLAY  LAST7                                     0007000
TADD     LAST8      BATCHADD  8-111-8888D08/R08      0008000
checkpoint "D0100010" here
DELETE   LAST8                                     0009000
DISPLAY  LAST8                                     0010000
```

IMSql – Demo – Checkpoint/Restart



- Step 1: Open powershell and navigate to directory
“C:\Raincode360\Repositories\Demos\IMSql\CheckpointRestart”
- Step 2: Type command **make demo** to run the first step of the demo:
 - Execute “RCD001.EPRCD003.JCL”
 - Run in non-restart mode
→ ABEND = div/0
 - The 6th command is ABEND
 - Two checkpoints
 - “TADD LAST7C” is between the last checkpoint and the ABEND. So, it’s rollback (not committed to the DB)
 - Return RC = 9000 + div/0

```
**P-INPUT:ADD LAST7 BATCHADD 8-111-7777D07/R07 0001000
CHECKPOINT FREQ:000001:CHKPT-FREQUENCY:00000001:COUNT:00001
CHECKPOINT CHKP-AREA: :COUNTER:00001
**P-INPUT:DISPLAY LAST7 0002000
CHECKPOINT FREQ:000001:CHKPT-FREQUENCY:00000001:COUNT:00002
CHKP-AREA:D0100002
CHECKPOINT CHKP-AREA:D0100002:COUNTER:00000 2 checkpoints
**P-INPUT:ADD LAST7B BATCHUPD 0003000
CHECKPOINT FREQ:000001:CHKPT-FREQUENCY:00000001:COUNT:00001
CHECKPOINT CHKP-AREA:D0100002:COUNTER:00001
**P-INPUT:DISPLAY LA[2021-06-02 16:29:01.102] [1] [ERROR] [ABENDDUMP]: RainCodeLegacyRuntime.Conditions.Cobol.Exception: Division by zero
at RainCodeLegacyRuntime.Conditions.Cobol.ConditionRuntime.throwNonMasquableException(ErrorCodeEnum errorCode, String message)
at RainCodeLegacyRuntime.Conditions.Cobol.ConditionRuntime.signalZeroDivide(String message)
at RainCodeLegacyRuntime.Types.FixedDecValue.Divide(ExecutionContext ctx, FixedDecValue one, FixedDecValue two, FixedDecimalDescriptor de
at RainCode.Generated.PRCDD003.proc$ABEND_IO(ExecutionContext v_ec 591, MemoryArea v_wa 590, MemoryArea& AUTO_MEM_cob_PRCDD003_179, CTX v
at RainCode.Generated.PRCDD003.rc_dispatch(ExecutionContext v_ec 219, MemoryArea v_wa 218, MemoryArea& AUTO_MEM_cob_PRCDD003_179, CTX v_p
at RainCode.Generated.PRCDD003.proc$PROCESS_INPUT(ExecutionContext v_ec 304, MemoryArea v_wa 303, MemoryArea& AUTO_MEM_cob_PRCDD003_179, C
at RainCode.Generated.PRCDD003.rc_dispatch(ExecutionContext v_ec 219, MemoryArea v_wa 218, MemoryArea& AUTO_MEM_cob_PRCDD003_179, CTX v_p
at RainCode.Generated.PRCDD003.proc$READ_INPUT(ExecutionContext v_ec 206, MemoryArea v_wa 205, MemoryArea& AUTO_MEM_cob_PRCDD003_179, CTX
at RainCode.Generated.PRCDD003.rc_dispatch(ExecutionContext v_ec 219, MemoryArea v_wa 218, MemoryArea& AUTO_MEM_cob_PRCDD003_179, CTX v_p
at RainCode.Generated.PRCDD003.cob_PRCDD003(ExecutionContext v_ec 11, MemoryArea v_wa 10, MemoryArea v_IOPCB_12, MemoryArea v_DBPCB_13, Mem
at RainCode.Generated.PRCDD003.Execute(ExecutionContext v_ec 8, MemoryArea v_wa 7, CallParameters v_args)
at RainCodeLegacyRuntime.Module.Executable.Execute(ExecutionContext ec, CallParameters paramList)
at RainCodeLegacyRuntime.Module.Executable.ExecuteMain(ExecutionContext ec, CallParameters paramList)
STA RESTART
CHECKPOINT FREQ:000001:CHKPT-FREQUENCY:00000001:COUNT:00002
CHKP-AREA:D0100004
CHECKPOINT CHKP-AREA:D0100004:COUNTER:00000
**P-INPUT:TADD LAST7C BATCHUPD 8-111-7777D07/R07 0004000
INSERT IS DONE, no REPLY nec
CHECKPOINT FREQ:000001:CHKPT-FREQUENCY:00000001:COUNT:00001
CHECKPOINT CHKP-AREA:D0100004:COUNTER:00001
**P-INPUT:ABEND 0004100
ABEND-IO NOW WE ABEND THIS JOB
TO TEST THE CHKP RESTARTED:
JOB0000000030 : DFSRRC00: return with rc:9000
```

IMSql – Demo – Checkpoint/Restart



- Step 2: Type the command **make restart** to run the second step of the demo:
 - Execute “RCD001.EPRCDR03.JCL”
 - Run in Restart mode → ABEND = no op
//*HIDAMUPD EXEC PROC=IMSBATCH, MBR=DFSIVA64, PSB=DFSIVP64, IMSID=IVP1,
//* SOUT='*', TIME=(60), CKPTID=LAST
 - (Re)start after the last checkpoint
 - Succeed with RC = 0

```
CHKPT-XRST-AREA:D0100004 :RESTARTED:Y
AFTER XRST STATUS:
IOPCB: RCPC01 L L Jean.Hen
FOR READ-CHKPT-FREQ
FREQ:000001:CHKPT-FREQUENCY:00000001
AFTER READ-CHKPT-FREQ:000001
**P-INPUT:TADD LAST7C BATCHUPD 8-111-7777D07/R07 0004000
INSERT IS DONE, no REPLY nec
CHECKPOINT FREQ:000001:CHKPT-FREQUENCY:00000001:COUNT:00001
CHECKPOINT CHKP-AREA: :COUNTER:00001
**P-INPUT:ABEND 0004100
ABEND-IO NOW WE ABEND THIS JOB
TO TEST THE CHKP RESTARTED:Y
NO ABEND BECAUSE WE ARE IN RESTART
CHECKPOINT FREQ:000001:CHKPT-FREQUENCY:00000001:COUNT:00002
CHKP-AREA:D0100006

**P-INPUT:DISPLAY LAST8 0010000
CHECKPOINT FREQ:000001:CHKPT-FREQUENCY:00000001:COUNT:00002
CHKP-AREA:D0100012
CHECKPOINT CHKP-AREA:D0100012:COUNTER:00000
JOB00000000031 : DFSRRC00: return with rc:0
```

IMSql – Demo – Checkpoint/Restart



RCD001GO after first run (ABEND)

```
*****
* TEST FOR CHECKPOINT RESTART
*****
TRANSACTION TYPE : BMP/DLI (HIDAM DB)
DATE : 06/11/21 09:52:36
PROCESS CODE (*1) : DISPLAY
LAST NAME : LASTA
FIRST NAME :
EXTENSION NUMBER :
INTERNAL ZIP CODE :
(*1) PROCESS CODE
ADD
DELETE
UPDATE
DISPLAY
TADD

SPECIFIED PERSON WAS NOT FOUND
SEGMENT# : 0004

*****
* TEST FOR CHECKPOINT RESTART
*****
TRANSACTION TYPE : BMP/DLI (HIDAM DB)
DATE : 06/11/21 09:52:36
PROCESS CODE (*1) : TADD
LAST NAME : LAST7C
FIRST NAME : BATCHUPD
EXTENSION NUMBER : 8-111-7777
INTERNAL ZIP CODE : D07/R07
(*1) PROCESS CODE
ADD
DELETE
UPDATE
DISPLAY
TADD

ENTRY WAS ADDED
SEGMENT# : 0005
```

Last checkpoint (D0100004)

RCD001GO after second run (SUCCEED)

```
*****
* TEST FOR CHECKPOINT RESTART
*****
TRANSACTION TYPE : BMP/DLI (HIDAM DB)
DATE : 06/11/21 09:52:36
PROCESS CODE (*1) : DISPLAY
LAST NAME : LASTA
FIRST NAME :
EXTENSION NUMBER :
INTERNAL ZIP CODE :
(*1) PROCESS CODE
ADD
DELETE
UPDATE
DISPLAY
TADD

SPECIFIED PERSON WAS NOT FOUND
SEGMENT# : 0004

*****
* TEST FOR CHECKPOINT RESTART
*****
TRANSACTION TYPE : BMP/DLI (HIDAM DB)
DATE : 06/11/21 09:54:52
PROCESS CODE (*1) : TADD
LAST NAME : LAST7C
FIRST NAME : BATCHUPD
EXTENSION NUMBER : 8-111-7777
INTERNAL ZIP CODE : D07/R07
(*1) PROCESS CODE
ADD
DELETE
UPDATE
DISPLAY
TADD

ENTRY WAS ADDED
SEGMENT# : 0001
```

First run

Second run

IMSql – Checkpoint (Raincode Console)



The screenshot below shows the Checkpoints created after the **first run** of the [checkpoint/restart](#) demo.

Check Points

Checkpoint Name	Program Name	Jcl Name	Psb Name	
D0100002	PRCD003	DR01EPR3	RCPC01	<button>× DELETE</button>
D0100004	PRCD003	DR01EPR3	RCPC01	<button>× DELETE</button>

1

1 - 2 of 2 items

IMSql – Checkpoint (Raincode Console)



The screenshot below shows the Checkpoints created after the **second run** of the [checkpoint/restart](#) demo.

Check Points

Checkpoint Name	Program Name	Jcl Name	Psb Name	
D0100002	PRCD003	DR01EPR3	RCPC01	<button>× DELETE</button>
D0100004	PRCD003	DR01EPR3	RCPC01	<button>× DELETE</button>
D0100006	PRCD003	DR01EPR3	RCPC01	<button>× DELETE</button>
D0100008	PRCD003	DR01EPR3	RCPC01	<button>× DELETE</button>
D0100010	PRCD003	DR01EPR3	RCPC01	<button>× DELETE</button>
D0100012	PRCD003	DR01EPR3	RCPC01	<button>× DELETE</button>

1

1 - 6 of 6 items



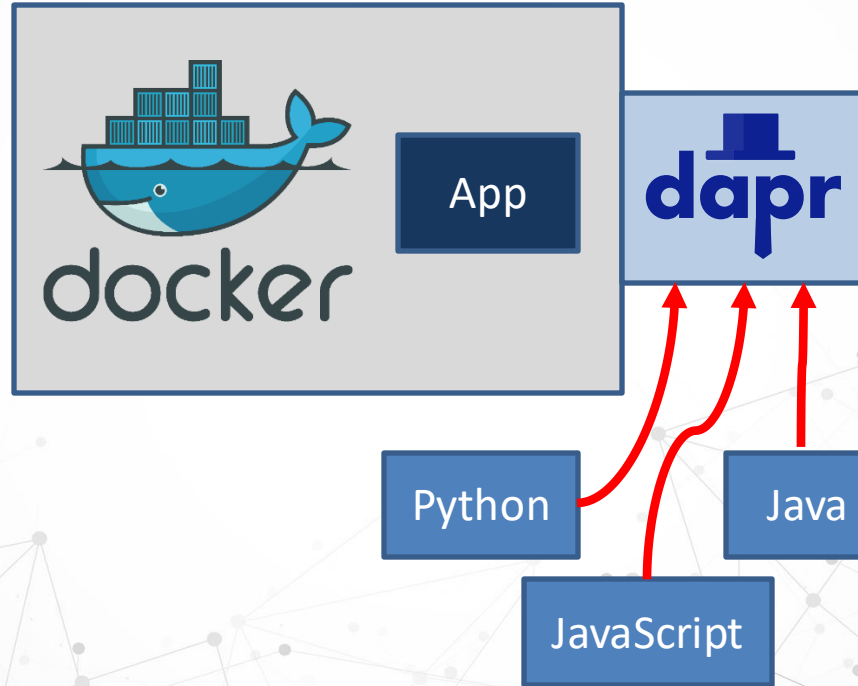
Cloud Native Deployment

More than your good old mainframe migration



- Deployment of mission-critical business logic at scale
 - Hundreds of transactions
 - Hundreds of batch jobs
- Using state-of-the-art cloud-native technologies
 - .NET 10.0
 - Azure Container Apps
 - Kubernetes + Dapr
- Without the burden of antiquated infrastructure
- Easy consumption by applications written in modern languages
- Fulfilling the promise of digital transformation

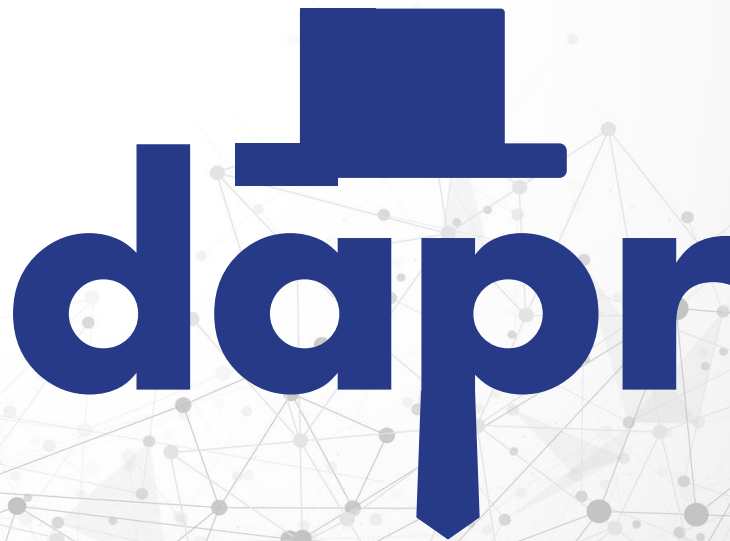
Exposing mainframe business logic for non-mainframe consumption



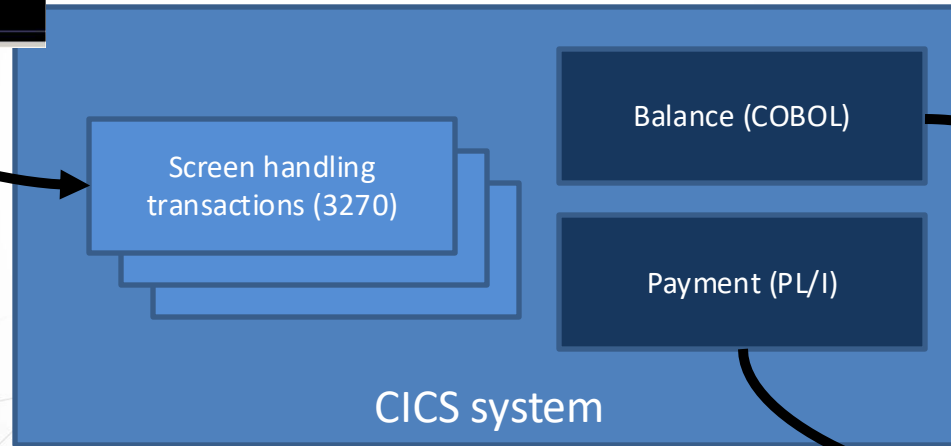
The mainframe code becomes a service
Usable from any application with a HTTP client

- Portable event-driven runtime for building microservices
- Open source, driven by Microsoft
- Language agnostic
- Platform agnostic
 - Cloud
 - Edge devices

<https://dapr.io>



Cloud Native Demo architecture: the starting point



Cloud Native Demo architecture: keeping the business logic



Balance (COBOL)

Payment (PL/I)

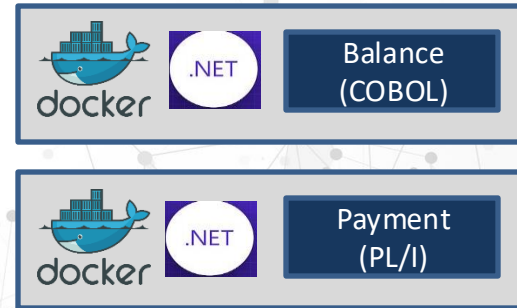
Cloud Native Demo architecture: keeping the business logic



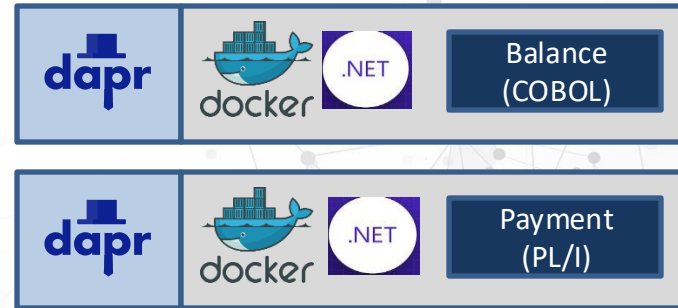
Balance
(COBOL)

Payment
(PL/I)

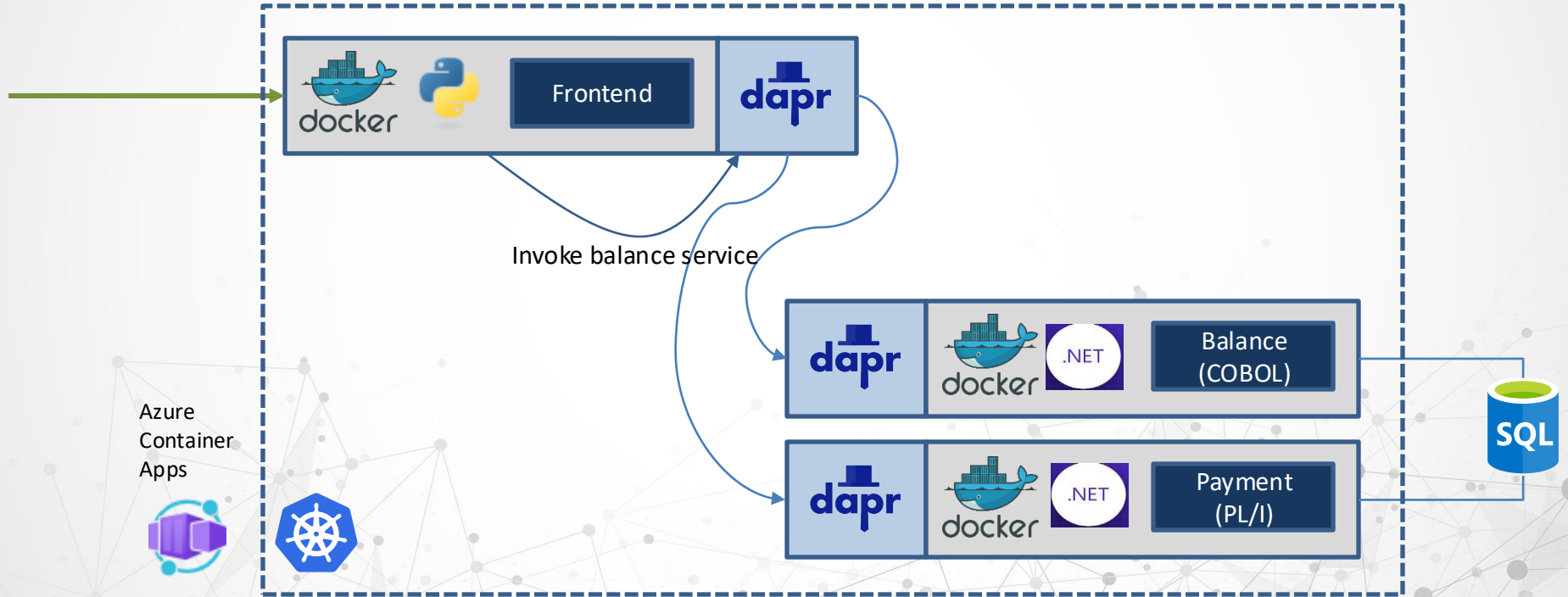
Cloud Native Demo architecture: encapsulation in containers



Cloud Native Demo architecture: injecting a Dapr sidecar



Cloud Native Demo architecture



Cloud Native Demo – Automated structure/JSON mapping

01 W-COMMAREA.

...

10 C007.

20 C007-ACCOUNTNUMBER PIC X(20).

20 C007-AMOUNT PIC ---.-----.--9,99.

...

JSON input request body to balance service:

```
{
  "C007": {
    "C007-ACCOUNTNUMBER": "BE28-..."
  }
}
```

Cloud Native Demo – Automated structure/JSON mapping

01 W-COMMAREA.

...

10 C007.

20 C007-ACCOUNTNUMBER PIC X(20).

20 C007-AMOUNT PIC ---.-----.--9,99.

...

JSON output response body from balance service:

```
{
  "C007": {
    "C007-ACCOUNTNUMBER": "BE28-...",
    "C007-AMOUNT": 123.45
  }
}
```

Cloud Native Demo Setup



- Raincode 360 provides the Cloud Native demo out-of-the-box
 - Azure resources are created (Azure Container Apps Environment, etc.)
 - The Raincode Bank Application demo gets automatically deployed onto the Container Apps
 - Dapr is enabled on each of these Container Apps
- Deployed with the distributed tracing feature of Dapr enabled, allowing for application analysis through Azure Application Insights.
- The Cloud Native demo shares the same SQL database as the CICS demo.
 - Database changes made on one side will be reflected on the other.
- The demo wraps 4 programs from the “Qix - Emulator - Bank Application” demo as Dapr microservices:
 - Account list (COB010A.cob)
 - Account balance (COB011.cob)
 - Payment (PLI001.pli)
 - Customer list (COB004A.cob)

Cloud Native Demo – How to demonstrate



- Click on the Cloud native demo  desktop shortcut to access the demo frontend.



Raincode CICS on Container Apps demo

At any time, feel free to take a peek at the [application map on Application Insights](#) for this demo.

Account balance

BE00-1000-0000-1438

Get account balance

Create transaction

Transfer from

BE00-1000-0000-1438

Transfer to

BE00-1000-0000-1438

Amount

0

Description

Transfer

Customer lookup

Customer last name prefix

Lookup

Interface is divided into three sections



Account Balance

- Select the account from drop-down menu of the account number box
- Click on 'Get account balance' for the balance inquiry

Account balance

BE00-1000-0000-1438

▼

Get account balance

Account balance for BE00-1000-0000-1438

EUR 1064.80

Account balance

BE00-1000-0000-1438 ▼

BE00-1000-0000-1438 ▲

BE00-1000-0000-2643

BE00-1000-0000-3557

BE00-1000-0000-4464

BE00-1000-0000-4488

BE00-1000-0000-5387

BE00-1000-0000-6168

BE00-1000-0000-6172

BE00-1000-0000-6320



Create Transaction

- Creates the payment
- Select **Transfer from** to **Transfer to**
- Add **Amount** and **Description**, and press the transfer button

Create transaction

Transfer from	Transfer to	Amount	Description
BE00-1000-0018-4578 ▾	BE00-1000-0018-4578 ▾	0	

Transfer

Transaction report

Executed transaction for amount: 100

Balances before transaction

BE00-1000-0018-4578	EUR 6715.85
BE00-1000-0079-1517	EUR 9294.49

Balances after transaction

BE00-1000-0018-4578	EUR 6615.85
BE00-1000-0079-1517	EUR 9394.49



Customer lookup

- Looks for the customer in the database that matches the prefix entered.
- Click Lookup to populate the list on UI

Customer lookup

Customer last name prefix

Lookup

Customer lookup result

#	Last name	First name	Date of birth	City
612	Webb	Aimee	Thu 17 Apr 1997	1300 Limal
932	Webster	Elma	Thu 13 Oct 1960	7000 MONS
199	Webster	Kristy	Mon 20 Apr 1959	3000 LEUVEN
1018	Weeks	Cecelia	Sun 15 Jun 1997	8000 BRUGGE
284	Weeks	Milagros	Thu 17 Mar 1983	7000 MONS
925	Welch	Olin	Tue 19 Mar 1985	2000 ANTWERPEN
641	Werner	Susanne	Fri 4 Oct 1996	1000 BRUSSEL

Cloud Native Demo – Dapr distributed tracing

- At any point, the Application Map can be reached through the link on the demo frontend.



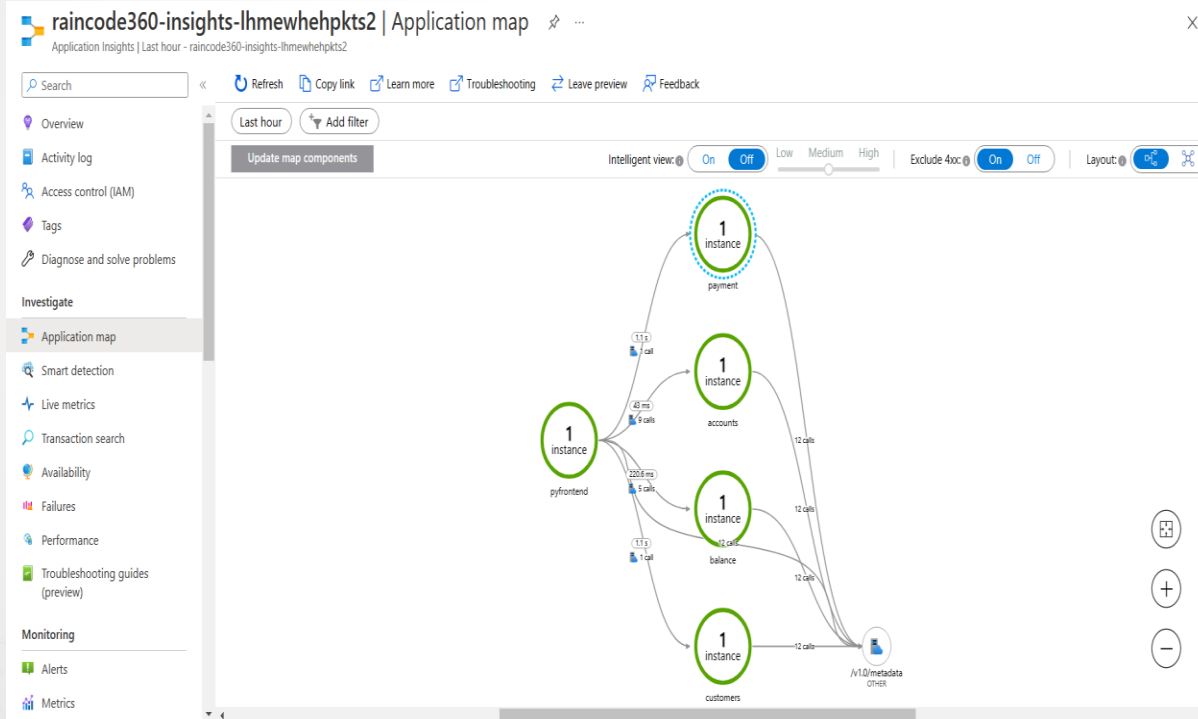
Raincode CICS on Container Apps demo

At any time, feel free to take a peek at the [application map on Application Insights](#) for this demo.

Cloud Native Demo – Dapr distributed tracing



- Each node represents an application component
- Each edge represents a relationship (i.e., call) between identified components



Application Map identifies:

- Web frontend (pyfrontend)
- Legacy services:
 - Accounts (COBOL)
 - Balance (COBOL)
 - Customers (COBOL)
 - Payment (PL/I)
- User can see how many calls have been made, and their performance.

There will be a slight lag whilst information is populated in the Application Map after an action is made through the Cloud Native Demo frontend (usually less than 5 minutes).

Cloud Native Demo – Sources



- The sources for the Cloud Native demo are available on the Raincode 360 environment
- Contents:
 - The frontend written in Python 3 using the Django 3.2 framework
 - C:\Raincode360\Repositories\Demos\Azure\Dapr\pyfrontend
 - Our generated wrapper images for legacy programs & frontend from the Bank Application demo
 - <https://hub.docker.com/search?q=raincode360>
- Raincode tooling support for automated containerization of legacy applications is currently under development.

Cloud Native Demo – Rehost QIX-Emulator bank demo on AKS



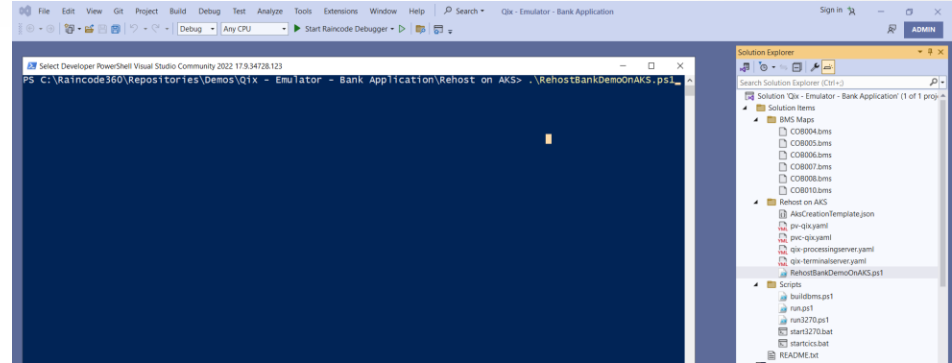
- Rehost QIX emulator bank demo on AKS using containerization
- Prerequisites:
 - You must be a contributor to the Azure subscription where the Raincode 360 VM is hosted.
 - You must execute the PowerShell script (**RehostBankDemoOnAKS.ps1**) in the Administrator mode

Cloud Native Demo – Rehost QIX-Emulator bank demo on AKS



There are two ways to execute the demo:

1: Open the solution from the folder
C:\Raincode360\Repositories\Demos\Qix - Emulator - Bank Application\ Rehost on AKS, and execute the PowerShell script (***RehostBankDemoOnAKS.ps1***) by navigating to the Tools -> Command line options -> Developer Powershell



OR

2: Execute the demo through a shortcut placed on the desktop



Cloud Native Demo – Rehost QIX-Emulator bank demo on AKS



- Once you execute the script, it will ask you for the login method. There are two types of login methods:
 - User mode (U) – Login with the user credentials through the Azure portal. Azure portal opens in an explorer.
 - Tenet mode (T) - The tenant credentials required to Log in to Azure.
- Once you input the login method, it will ask for a SubscriptionId.
- Next, it will ask for the Raincode360-VM resource group's name for VNET peering.

```
Developer PowerShell Visual Studio Community 2022 17.9.34728.123
*****
** Visual Studio 2022 Developer PowerShell v17.9.6
** Copyright (c) 2022 Microsoft Corporation
*****
PS C:\Raincode360\Repositories\Demos\Qix - Emulator - Bank Application> cd '.\Rehost on AKS\'
PS C:\Raincode360\Repositories\Demos\Qix - Emulator - Bank Application\Rehost on AKS> .\RehostBankDemoOnAKS.ps1

cmdlet RehostBankDemoOnAKS.ps1 at command pipeline position 1
Supply values for the following parameters:
AzureLoginModeTorU: U
SubscriptionId: 7c12f307-10c0-4aa1-bcea-89f1f9dab193
Raincode360VMResourceGroupName: raincode360_
```

Cloud Native Demo – Rehost QIX-Emulator bank demo on AKS



- SubscriptionId and the resource group name are copied from the Resource group of raincode360-vm on the Azure portal.

Home > raincode360 >

raincode360-vm Virtual machine

Search Connect Start Restart Stop Hibernate Capture Delete Refresh Open in mobile Feedback CLI / PS

Overview

- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Connect
- Networking
- Settings

Essentials

Resource group (move) : **raincode360**

Status : Running

Location : East US

Subscription (move) : [Microsoft Azure Sponsorship](#)

Subscription ID : **7c12f307-10c0-4aa1-bcea-89f1f9dab193**

[JSON View](#)

Operating system : Windows (Windows Server 2022 Datacenter Azure Edition)

Size : Standard B2ms (2 vcpus, 8 GiB memory)

Public IP address : [40.114.48.64](#)

Virtual network/subnet : [raincode360-vnet/BaseSubnet](#)

DNS name : [Not configured](#)

Health state : -

Time created : 27/6/2024, 11:26 pm UTC

It is recommended that the AKS cluster and file storage be deployed within the same subscription ID.

Cloud Native Demo – Rehost QIX-Emulator bank demo on AKS



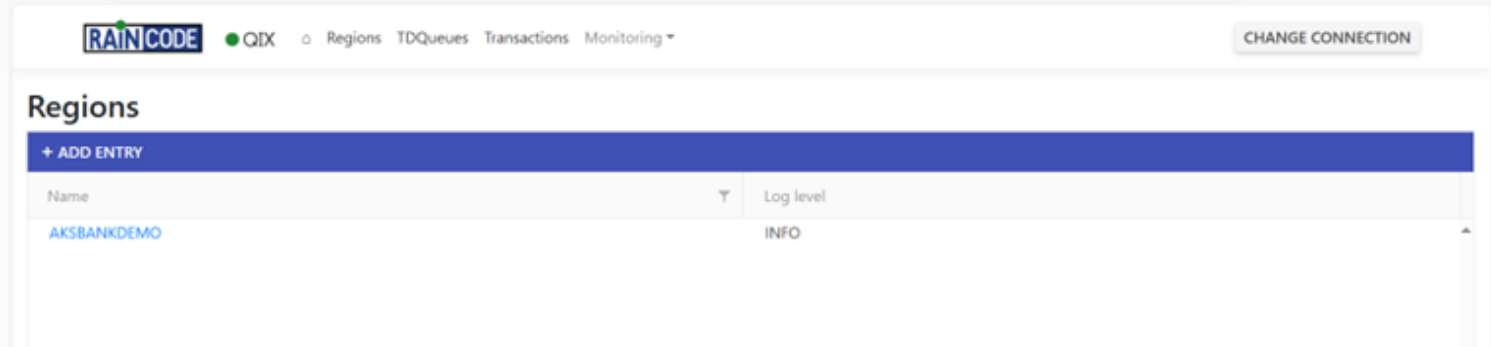
- When you press enter after providing the resource group name, it will set up a QIX region AKSBANKDEMO.

```
cmdlet RehostBankDemoOnAKS.ps1 at command pipeline position 1
Supply values for the following parameters:
AzureLoginModeTorU: U
SubscriptionId: 7c12f307-10c0-4aa1-bcea-89f1f9dab193
Raincode360VMResourceGroupName: raincode360
[2024-07-01 07:47:59] : Started the 'Rehosting of BankDemo Emulator Application to AKS' at: 07-01-2024 07:47
[2024-07-01 07:47:59] : Building solution C:\Raincode360\Repositories\Demos\Qix - Emulator - Bank Application\Qix - Emulator - Bank Application.sln in Release
mode...
[2024-07-01 07:47:59] : Writing stdout to C:\Raincode360\Logs\msbuild.20240701T0747596597Z.out.log.
[2024-07-01 07:47:59] : Writing stderr to C:\Raincode360\Logs\msbuild.20240701T0747596597Z.err.log.
[2024-07-01 07:48:06] : Solution C:\Raincode360\Repositories\Demos\Qix - Emulator - Bank Application\Qix - Emulator - Bank Application.sln built successfully
in Release mode.
[2024-07-01 07:48:06] : Created DLLs directory and copied the program executables
[2024-07-01 07:48:06] : Created Maps directory and copied the Bms map Xmls
[2024-07-01 07:48:06] : Writing stdout to C:\Raincode360\Logs\regioncreator.20240701T0748068423Z.out.log.
[2024-07-01 07:48:06] : Writing stderr to C:\Raincode360\Logs\regioncreator.20240701T0748068423Z.err.log.
[2024-07-01 07:48:08] : Created 'AKSBANKDEMO' region.
```

Cloud Native Demo – Rehost QIX-Emulator bank demo on AKS



- You can see the created QIX region through the Raincode console.



Name	Region Name	Program	Twa Size	
C004	AKSBANKDEMO	COB004	512	CONFIGURE DELETE
C005	AKSBANKDEMO	COB005	512	CONFIGURE DELETE
C006	AKSBANKDEMO	COB006	512	CONFIGURE DELETE
C007	AKSBANKDEMO	COB007	512	CONFIGURE DELETE
C008	AKSBANKDEMO	COB008	512	CONFIGURE DELETE
C010	AKSBANKDEMO	COB010	512	CONFIGURE DELETE

- You can check the transactions through QIX transactions tab in the Raincode console

For more details, refer to [Raincode Console](#)

Cloud Native Demo – Rehost QIX-Emulator bank demo on AKS



- The script will create a separate resource group for the AKS cluster and storage account.

Home > Resource groups Raincode SRL (phidani.be)

+ Create Manage view Refresh Export to CSV Open query Assign tags

Filter for any field... Subscription equals all Location equals all Add filter

Showing 1 to 35 of 35 records.

Name ↑↓	Subscription ↑↓	Location ↑↓
☒ AKS-raincode360	Microsoft Azure Sponsorship	East US
☐ cbmrc360	Microsoft Azure Sponsorship	France Central
☐ cloud-shell-storage-centralindia	Azure Sponsorship (exp 2024-03-01)	Central India
☐ cloud-shell-storage-west europe	Martin	West Europe

- Under the new resource group, a storage account and a file share folder named **qixfileshare** are created

```
[2024-07-01 07:48:16] : Created the resource group: AKS-raincode360
[2024-07-01 07:48:16] : Creating Storage account: qixstorageaccountwqvst
[2024-07-01 07:48:36] : Storage account: qixstorageaccountwqvst created under resource group: AKS-raincode360 at location: eastus
[2024-07-01 07:48:36] : Created the Fileshare 'qixfileshare'
```

Cloud Native Demo – Rehost QIX-Emulator bank demo on AKS



- In qixfileshare, two directories, a processingserver and a terminalserver, are created, and license files are uploaded to both. BMS XMLs are uploaded to the terminalserver, and program executables are uploaded to the processingserver.

```
[2024-07-01 07:48:36] : Created the directory 'processingserver' under FileShare 'qixfileshare'
[2024-07-01 07:48:36] : Created the directory 'terminalserver' under FileShare 'qixfileshare'
[2024-07-01 07:48:36] : Uploaded license to both 'terminalserver' and 'processingserver' folders under fileshare 'qixfileshare'
[2024-07-01 07:48:38] : Uploaded BMS xmls to 'terminalserver' folder under fileshare 'qixfileshare'
[2024-07-01 07:48:42] : Uploaded program executables to 'processingserver' folder under fileshare 'qixfileshare'
[2024-07-01 07:48:42] : Installing AZ module....
```

- In the AKS resource group, navigate to the file share folder. Within this folder, you will find the processingserver and terminalserver directories, which contain the License file, BMS XMLs, and DLLs.

The screenshot shows the Azure Storage browser interface for the storage account 'qixstorageaccountwqvst'. The left sidebar contains navigation options: Overview, Activity log, Tags, Diagnose and solve problems, Access Control (IAM), Data migration, Events, Storage browser (selected), and Storage Mover. The main pane shows the 'qixfileshare' folder under 'File shares'. It lists two subdirectories: 'processingserver' and 'terminalserver', both of type 'Directory'. The authentication method is 'Access key'.

Name	Type	Size
processingserver	Directory	...
terminalserver	Directory	...

Cloud Native Demo – Rehost QIX-Emulator bank demo on AKS



- Further, the script installs the Kubernetes cluster and creates peer networking between sqlserver and AKS virtual network.

```
[2024-07-01 07:48:43] : Installing Kubectl ....
[2024-07-01 07:48:43] : Kubectl installation completed successfully.
[2024-07-01 07:48:43] : Creating kubernetes service using AksTemplateFile 'C:\Raincode360\Repositories\Demos\Qix - Emulator - Bank Application\Rehost on AKS\
ksCreationTemplate.json' under resouce group 'AKS-raincode360'.
[2024-07-01 07:53:33] : Kubernetes service created successfully.
[2024-07-01 07:53:33] : Creating network peer between SqlServer and AKS virtual networks
[2024-07-01 07:53:45] : Peer networking added between 'raincode360-vnet' & 'aks-vnet-35064155'
[2024-07-01 07:53:56] : Peer networking added between 'aks-vnet-35064155' & 'raincode360-vnet'
[2024-07-01 07:53:56] : Connecting to Kubernetes cluster to deploy resources...
[2024-07-01 07:53:57] : NameSpace created: 'raincodeqix'
[2024-07-01 07:53:57] : Secret created: 'qix-storage-secret'
[2024-07-01 07:53:57] : Secret created: 'qix-config-secret'
[2024-07-01 07:53:58] : Qix Persistant volume created
[2024-07-01 07:53:58] : Qix Persistant volume claim created
[2024-07-01 07:53:58] : Qix processing server deployment created
[2024-07-01 07:53:59] : Qix terminal server deployment & service created
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED	NODE	READINESS
GATES									
qix-processing-server-5cdd5bc7c8-k8rr1	0/1	ContainerCreating	0	1s	<none>	aks-agentpool-11006533-vmss000001	<none>		<none>
qix-terminal-server-74b5d78f6b-7sscm	0/1	Pending	0	0s	<none>	<none>	<none>		<none>

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE	SELECTOR
qix-terminal-server	LoadBalancer	10.0.140.141	<pending>	993:30485/TCP	0s	app=qix-terminal-server

[2024-07-01 07:53:59] : Waiting 45 seconds for containers & service to be live...

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED	NODE	READINESS	GATES
qix-processing-server-5cdd5bc7c8-k8rr1	1/1	Running	0	47s	10.244.0.11	aks-agentpool-11006533-vmss000001	<none>		<none>	
qix-terminal-server-74b5d78f6b-7sscm	1/1	Running	0	46s	10.244.0.12	aks-agentpool-11006533-vmss000001	<none>		<none>	

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE	SELECTOR
qix-terminal-server	LoadBalancer	10.0.140.141	4.156.121.63	993:30485/TCP	46s	app=qix-terminal-server

[2024-07-01 07:54:45] : About to wait 120 seconds for you to inspect and ready for disconnecting....

Cloud Native Demo – Rehost QIX-Emulator bank demo on AKS



- Under the deployments tab, you can find the AKSCreation template.

Home > AKS-raincode360

AKS-raincode360 | Deployments ☆ ...

Resource group

Search

Refresh Cancel Redeploy Delete View template

Filter by deployment name or resources in the deployment...

Deployment name	Status	Last modified	Duration	Related events
AksCreationTemplate	Succeeded	28/6/2024, 6:02:40 am	4 minutes, 37 seconds, and 836 ...	Related events

Overview

Activity log

Access control (IAM)

Tags

Resource visualizer

Events

Settings

Deployments

Home > Resource groups > AKS-raincode360 > qixakscuster

qixakscuster | Workloads ☆ ...

Kubernetes service

Search

Create Scale Delete Refresh Show labels Give feedback

Deployments Pods Replica sets Stateful sets Daemon sets Jobs Cron jobs

Filter by pod name Enter the full pod name

Status All statuses

Filter by namespace raincodeqix

Add label filter

Name	Namespace	Ready	Status	Restart count	Age	Pod IP	Node
qix-processing-server-5cdd...	raincodeqix	1/1	Running	0	9 minutes	10.244.0.11	aks-agentpool-1100653...
qix-terminal-server-74b5d7...	raincodeqix	1/1	Running	0	9 minutes	10.244.0.12	aks-agentpool-1100653...

Tags

Diagnose and solve problems

Microsoft Defender for Cloud

Cost analysis (preview)

Kubernetes resources

Namespaces

Workloads

Services and ingresses

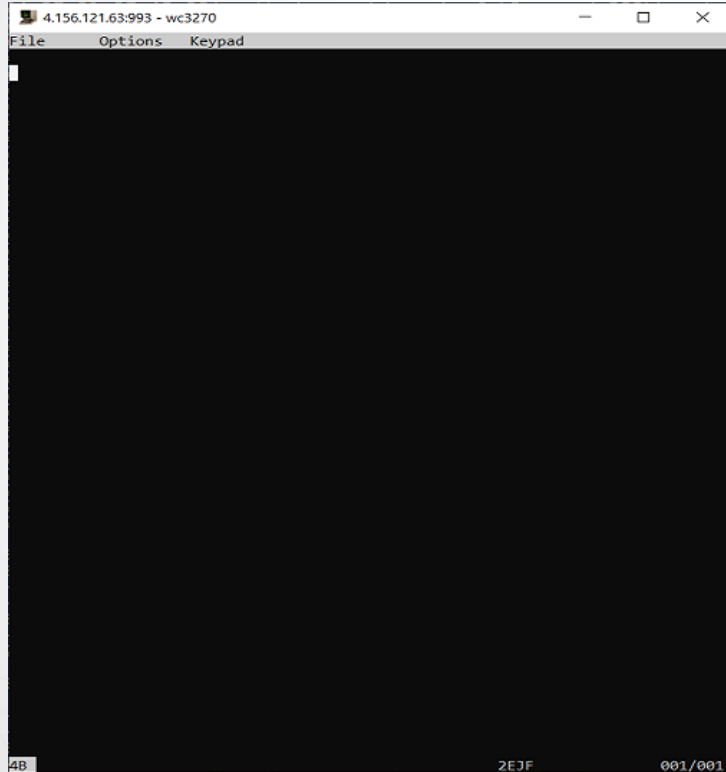
Storage

- If you choose the Pods tab under the Workloads, you can see the running containers.

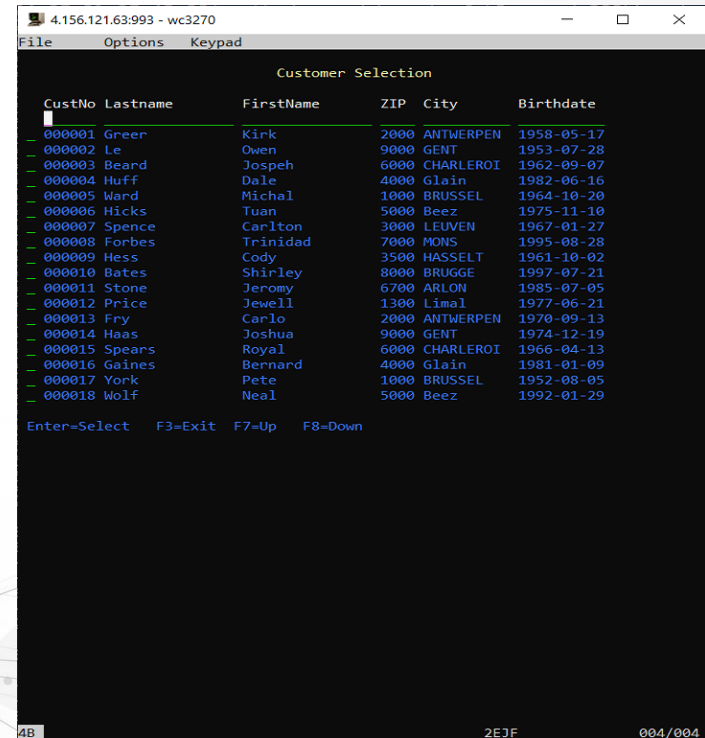
Cloud Native Demo – Rehost QIX-Emulator bank demo on AKS



Once the script is successfully executed, it will connect you to the 3270 terminal



Type C004 the first transaction id of QIX emulator bank application and press enter



Cloud Native Demo – Rehost QIX-Emulator bank demo on AKS



You can check the transaction logs by navigating to the QIX Monitoring tab → Transaction Logs in the Raincode console

RAINCODE ● QIX ▾ Regions TDQueues Transactions Monitoring ▾ CHANGE CONNECTION

Transaction Logs

1 - 20 of 20 items

Region	Server	User	Transac...	Conversation	Term...	R	Begin	End
AKSBANKDEMO	6e84ea67-8356-4c3f-a0f7-0df7758d8ce3	CICSUSER	C008	c494b6c8-8037-ef11-b679-6045bdf8c1b	2EJF	0	01/07/2024 08:06:16:756	01/07/2024 08:06:16:856
AKSBANKDEMO	6e84ea67-8356-4c3f-a0f7-0df7758d8ce3	CICSUSER	C008	c094b6c8-8037-ef11-b679-6045bdf8c1b	2EJF	0	01/07/2024 08:06:14:456	01/07/2024 08:06:14:513
AKSBANKDEMO	6e84ea67-8356-4c3f-a0f7-0df7758d8ce3	CICSUSER	C010	85b177c2-8037-ef11-b679-6045bdf8c1b	2EJF	0	01/07/2024 08:06:11:020	01/07/2024 08:06:12:090
AKSBANKDEMO	6e84ea67-8356-4c3f-a0f7-0df7758d8ce3	CICSUSER	C008	81b177c2-8037-ef11-b679-6045bdf8c1b	2EJF	0	01/07/2024 08:06:09:096	01/07/2024 08:06:09:143
AKSBANKDEMO	6e84ea67-8356-4c3f-a0f7-0df7758d8ce3	CICSUSER	C007	7db177c2-8037-ef11-b679-6045bdf8c1b	2EJF	0	01/07/2024 08:06:08:586	01/07/2024 08:06:08:616
AKSBANKDEMO	6e84ea67-8356-4c3f-a0f7-0df7758d8ce3	CICSUSER	C005	79b177c2-8037-ef11-b679-6045bdf8c1b	2EJF	0	01/07/2024 08:06:05:536	01/07/2024 08:06:05:723
AKSBANKDEMO	6e84ea67-8356-4c3f-a0f7-0df7758d8ce3	CICSUSER	C004	75b177c2-8037-ef11-b679-6045bdf8c1b	2EJF	0	01/07/2024 08:06:03:963	01/07/2024 08:06:04:003
AKSBANKDEMO	6e84ea67-8356-4c3f-a0f7-0df7758d8ce3	CICSUSER	C004	90484aae-8037-ef11-b679-6045bdf8c1b	2EJF	0	01/07/2024 08:05:30:826	01/07/2024 08:05:30:960

RAINCODE ● QIX ▾ Regions TDQueues Transactions Monitoring ▾ CHANGE CONNECTION

Servers Heartbeat

1 - 2 of 2 items

Server	Type	Region	Host	Last Active Timesta...	Heartbeat	Status
c8aae300-4811-49c0-b1da-a8e9ffd2c25c	Terminal	AKSBANKDEMO	qix-terminal-server-74b5d78f6b-7sscm	07/01/2024 08:07:28:813	29 seconds ago	Running
6e84ea67-8356-4c3f-a0f7-0df7758d8ce3	Processing	AKSBANKDEMO	qix-processing-server-5cdd5bc7c8-k8rri	07/01/2024 08:07:27:926	30 seconds ago	Running

You can check the server heartbeat by navigating to the QIX monitoring tab → Servers Heartbeat in the Raincode console

For more details, refer to [Raincode Console](#)

Cloud Native Demo – Rehost QIX-Emulator bank demo on AKS



- After completing your job, select Y to disconnect from the Azure session and Y to clean up the created AKS resources

```
[2024-07-01 07:54:45] : About to wait 120 seconds for you to inspect and ready for disconnecting....  
Disconnect from the Azure Session(Y/N)? : y  
Clean-up the created AKS resources(Y/N)? : y  
[2024-07-01 08:09:16] : Removing Vnet peering from network - raincode360-vnet, Resource group - raincode360  
[2024-07-01 08:09:27] : Cleaning up the resource group AKS-raincode360  
[2024-07-01 08:15:44] : Logging out of Azure...  
[2024-07-01 08:15:44] : Rehosting completed at:07-01-2024 08:15  
Waiting for 0 seconds, press a key to continue ...
```

- If you select N, you must manually clean the created AKS resources



Expert mode

User-defined function



- The purpose of this demo is to demonstrate the usage of the Raincode scripting language to create a user-defined function to implement a coding guideline
- The user-defined function adds extra validation to the compiler so that it throws an error when the “GO TO” statement is encountered in the COBOL module

User defined function: Normal Flow



- Step 1: Open the solution from the folder Compiler – User Defined Coding Guideline

- Step 2: Select Build -> Rebuild Solution

- Step 3: In GOTOCLB.cob, set a breakpoint at GOBACK statement in PARA-1

- Step 4: Start the Raincode Debugger

Expected output: The execution will print “IN PARA-0” and “IN PARA-1” in consecutive lines.

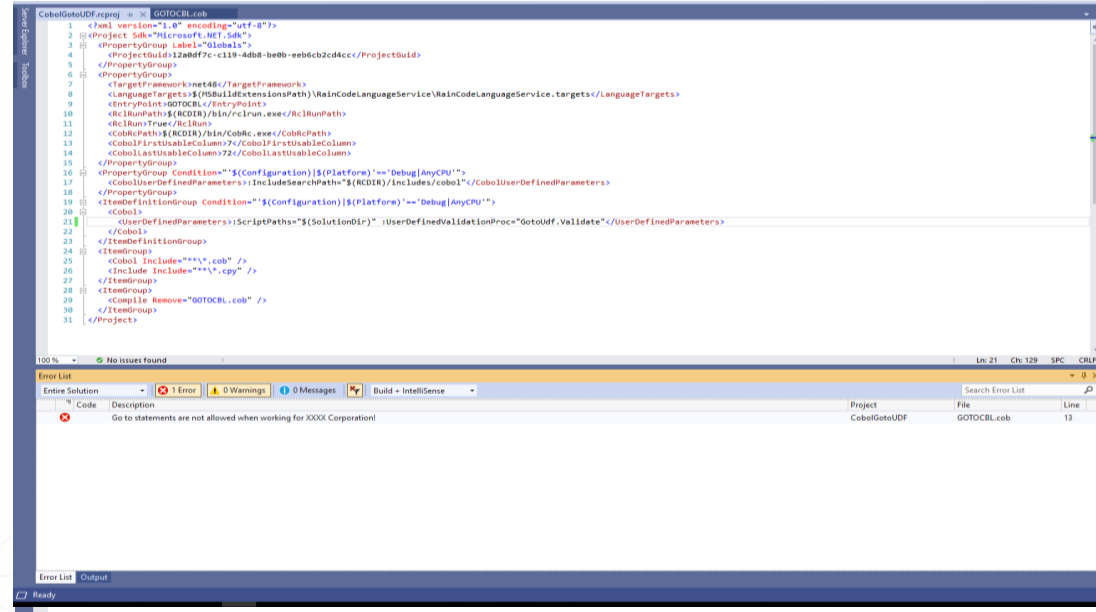
- Step 5: Click continue or close the application window to exit the debugger

The screenshot displays the Raincode IDE interface. The top pane shows a code editor with the following text:
[2021-09-30 11:30:01.369] [1] [WARNING] [RainCode.Core.CommandLine]: 'PauseOnAbnormalExit' is an internal argument and s
ubject to change at any time.
IN PARA-0
IN PARA-1
The bottom pane shows the 'Output' window with the text:
100% - No issues found
Line 21 On 21 SPC CRUF
The status bar at the bottom indicates 'Ready' and 'raincode-360'.

User defined function: Error Flow



- Step 1: Open the solution from the folder Compiler – User Defined Coding Guideline and click on CobolGotoUDF project from the solution explorer
- Step 2: From the displayed XML lines, uncomment "`<UserDefinedParameters>:ScriptPaths=\"$ (SolutionDir)\"`"
`:UserDefinedValidationProc="GotoUdf.Validate"</UserDefinedParameters>`"
- Step 3: Select Build -> Rebuild Solution
- Step 4: Start the Raincode Debugger



The compilation will fail, and an error message "Goto statements are not allowed when working for XXXX corporation!" will appear in the output tab

- Step 5: Click continue or close the application window to exit the debugger



Advanced Showcase

RAINCODE HEADQUARTERS



Rue de la Caserne 45
1000 Brussels, Belgium



+32 2 522 06 63



info@raincode.com

RAINCODE INDIA



#1144, Guru Nilayam, 3rd Floor, Phase 2,
HSR Layout, Bangalore – 560 102, India



+91 99450 47258



info@raincode.com

www.raincode.com