International Journal Research of Leading Publication (IJLRP)

E-ISSN: 2582-8010 ● Website: www.ijlrp.com ● Email: editor@ijlrp.com

CICS Transaction Processing on zOS: Core Concepts and Workflow

Chandra mouli Yalamanchili

chandu85@gmail.com

Abstract

For over five decades, IBM's Customer Information Control System (CICS) has remained a foundational transaction processing platform on IBM z/OS. This paper offers a detailed review of CICS by exploring its historical evolution, diverse architectural models, and the role of its specialized regions (TOR, AOR, DOR, and FOR). It discusses how resources are defined, stored, and managed via CSD and CEDA and explains CICS data objects and storage areas. A significant focus is given to the mechanics of transaction processing, supported by discussions on ACID properties, commit and rollback mechanisms, and recovery and restart features.

The paper also examines various CICS components like Task Control Blocks (TCBs), internal, inter, and external communication interfaces, and integration with external resource managers for MQ & DB2. Finally, the paper explores transaction processing with coding examples and explanations.

This paper represents a personal interpretation and synthesis of IBM documentation intended to aid those seeking to deepen their knowledge of CICS.

Keywords: CICS; z/OS; transaction processing; TCB; zIIP; DB2 integration; MQ integration; mainframe systems

Introduction

IBM CICS Transaction Server is a powerful transaction server that supports applications written in different languages. CICS provides online transaction management and connectivity for business-critical services. It can process incredibly high workloads in a scalable, performant, cost-efficient, secure environment. [2]

IBM introduced CICS in 1968, and since that time, billions of transactions a day have been processed on CICS in finance, insurance, retail, and government systems. According to IBM's reports, CICS handles billions of daily transactions across key industries, underpinning critical financial services, government, and retail systems. [1]Suchwidespreaduseunderscores the significant role of CICS as a fault-tolerant, high-throughput transaction processor.

Businesses today rely on CICS for their mission-critical workloads due to its scalability, security, and robust integration with other mainframe and distributed environments.

History of CICS

The journey of the CICS transaction server began in 1968 as a simple batch processing control system, evolving into a multi-threaded, multi-transaction processing engine. CICS was initially created to help airlines manage reservation systems; it has grown indispensable across banking, retail, insurance, and government sectors. Over the decades, CICS adopted support for multi-region operations, Distributed Program Link (DPL), Java integration, RESTful services, and cloud deployment support. The evolution of CICS has been marked by its ability to embrace modernization while maintaining backward compatibility [2].

CICS Architecture and Resources

CICS Architecture

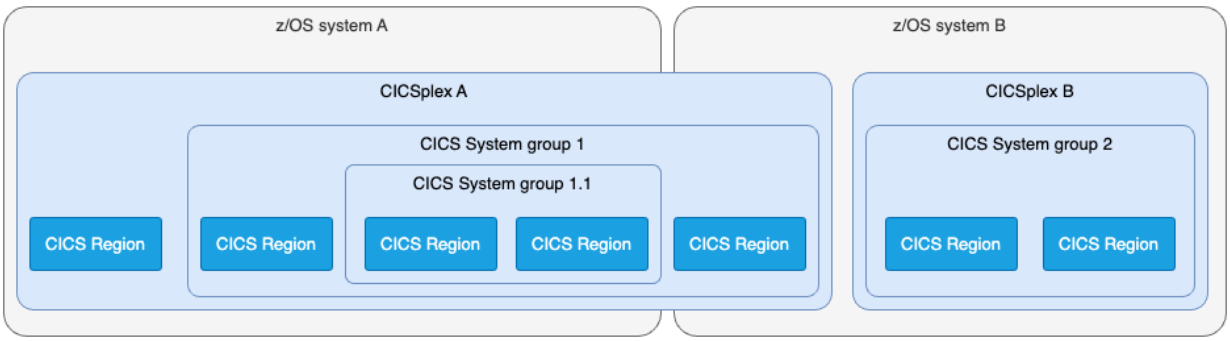

Figure 1: Illustrating the CICS regions within sysplex. [2]

IBM z/OS offers flexible options for setting up the CICS environment and deploying the CICSapplications and the corresponding resources across multiple CICS regions (servers). [2]Below are a few configuration components that would constitute a CICS application server setup:

- **Sysplex:** A set of z/OS systems that work together through the IBM mainframe's multisystem hardware components and software services. [2]
- **CICSplex:** A collection of CICS regions that can be managed together. Each CICS region can belong to a single CICSplex. CICSplex eases the CICS region management as it provides the single gateway for resource management across all CICS regions that are part of that CICSplex. [2]
- **CICS system group**: This is a collection of CICS regions within CICSplex; this will help to manage all related CICS regions as a single entity. For example, all test regions can be part of the test system group, and all production regions can be part of the production system group. One region can belong to more than one system group, and system groups can be part of other system groups. [2]
- **CICS region:** The instance of the CICS Transaction Server that runs as its own address space.

Types of CICS regions

While a CICS region can perform all roles, it's easy to manage when regions are assigned certain roles to provide effective resource isolation and sharing. Depending on the role they play below are the roles CICS regions play in a standard setup. [2]

- **Transport-owning region**: Region responsible for providing a communication interface to the CICS applications. Transaction requests this region receives are passed to an application-owning region for processing. [2]
- **Terminal-owning region (TOR)**: Region responsible for connecting terminals and other devices like printers to the CICSplex. Transaction requests this region receives are also passed to an application-owning region for processing. [2]
- **Application-owning region (AOR)**: A CICS region that manages the application program. [2]
- **Data-owning region (DOR)**: A CICS region that manages shared access among several application-owning regions to files and databases. [2]
- **File-owned region (FOR)**: A specific type of data-owning that manages access to files. [2]

CICS Resources

- Every resource CICS needs to execute the application, like programs, files, terminals, URI maps, DB connection, TDQ, temp storage, etc., needs to be defined to CICS resource definitions and have to be configured properly to make them available to the application during run time. [2]
- CICS provides several options for defining these resources through CEDA or DFHCSDUP. Some of the options include CICS explorer, bundles, RDO (Resource Definition online), CICSPlex SM Business application services, using CICS APIs like EXEC CICS SPI or EXEC CICS CSD, DFHCSDUP offline utility, Auto install, and Macro definition. [2]
- These resource definitions are stored in the CICS system definitions (CSD) VSAM file. [2]
- CICS manages some special resource definitions and configurations through several control tables. Below are the details of a few control tables available in CICS that control CICS execution and setup.
    - System initialization table (SIT): System initialization parameters.
    - File control table (FCT): Manages the BDAM file definitions needed for the region. VSAM files can also be present in FCT but won't be installed. We must use CSD to define and install VSAM and data table files. [2]
    - Program list table (PLT): A list of related programs. We use this table to specify the list of programs that are to be executed in the initialization programs phase of CICS startup, executed during the first or second quiesce stages of controlled shutdown, or both, or enabled or disabled as a group by a CEMT ENABLE or DISABLE command. [2]

- Data conversion table -
- Temporary storage table (TST): Generic names (or prefixes) for temporary storage queues. [2]
- Terminal control table (TCT): Store the information about non-SNA LU networks. [2]
- Transaction list table (XLT): Sets of logically related transaction identifications. [2]

CICS Interfaces

While in most cases, CICS is set up as a single system with all necessary resources, it can also be used in multiple system environments, in which it can communicate with other systems with similar communication facilities. [2]

Multiregion Operations (MRO)

MRO (Multiregion Operation) supports communication between CICS systems that run on the same MVS image or in the same MVS sysplex. MRO doesn't need networking support, and CICS support for region-to-region communication is called interregion communication (IRC). Below are different options available to establish MRO/IRC between two different CICS regions. [2]

- MRO/IRC: While defining the connection, we can use 'ACCESSMETHOD(IRC)` to use the provided interregion program DFHIRP. [2]
- MVS cross-memory (XM) services can be selected as a connection option between two CICS regions; in this case, DFHIRP is used only to open and close the interregion links. [2]
- A cross-system coupling facility (XCF) is required for MRO links between CICS regions in different z/OS images or sysplex. [2]

Intercommunications

CICS offers a couple of options for communicating with non-CICS remote systems or CICS systems that are not in the same OS or sysplex.[2]

- TCP/IP, also known as IP interconnectivity (IPIC). This option allows the communication and transfer of control between two CICS systems or from CICS TG through DPL or ECI requests. IPIC needs IPCONN and TCPIPSERVICE resource definitions in both systems. [2]
- ISC (Intersystem Communication) over SNA allows communication between CICS and non-CICS or CICS systems not in the same z/OS image or sysplex. [2]

External Communications

CICS offers the options below to communicate with non-CICS systems in peer-to-peer and client/server configurations. [2]

- TCP/IP: CICS programs can communicate over TCP/IP using the UNIX socket interface or application layer protocols. [2]
- Web services protocols: CICS provides comprehensive web services support, enabling CICS applications to support client/server web services functions. [2]
- HTTP (HyperText Transfer Protocol): CICS supports application programs acting as HTTP servers or clients. [2]
- SNA (Systems Network Architecture): CICS can connect to other systems using SNA protocols, including advanced program-to-program communication (APPC). [2]
- Remote procedure call (RPC): Client applications that support Open Network Computing Remote Procedure call (ONC RPC) or distributed Computing Environment/ RPC (DCE/ROC) can make remote procedure calls to invoke a CICS program. [2]

CICS also provides several interfaces, as mentioned below, to allow non-CICS environments to submit transaction requests:

- CICS Transaction Gateway (CICS TG): CICS TG enables Java applications to run programs in CICS using the Java EE interfaces. [2]
- CICS client applications: A CICS client is a member of the family of CICS workstation products that provide a standard set of functions for client/server computing. CICS clients can initiate transactions in CICS using the external call interface (ECI) or external presentation interface (EPI). [2]
- CICS-MQ bridge: WebSphere MQ applications can use the CICS-MQ bridge to initiate CICS transactions and process messages. [2]
- TCP/IP socket interface: The TCP/IP socket interface for CICS is part of the z/OS communications server component of z/OS. [2]
- External CICS Interface (EXCI): A z/OS program can use the EXCI to invoke a CICS program using DPL (Distributed Program Link). [2]

CICS Integrations with external resources

CICS provides the connection between the application programs and external resource managers through adapters; this CICS feature allows applications to include commands to be executed by these external resource managers. [2]

- CICS-DB2 interface: Allows the CICS application programs to execute SQL commands and access data in DB2.
- CICS-WebSphere MQ adapter: It allows CICS applications to communicate with WebSphere MQ as an external resource manager using the CICS Resource Manager Interface (RMI). [2]

- CICS-WebSphere MQ bridge: It enables non-CICS applications to use MQ messages to run a program or transaction on CICS and get a response. [2]

CICS Applications

Application Architecture

The CICS Transaction server offers various services, as depicted in the below picture, to the applications running within the CICS environment.

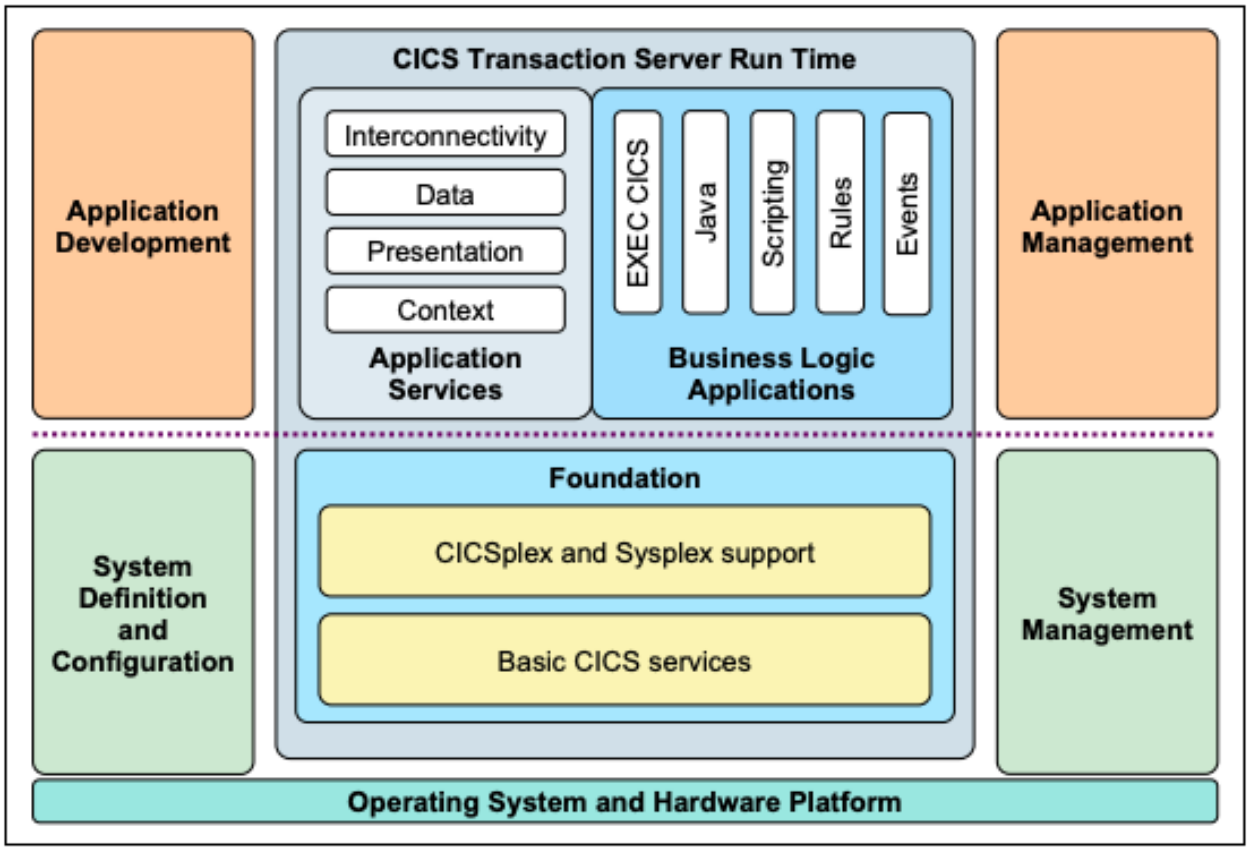

Figure 2: Illustrating CICS services from application context. [1]

Design principles

As per IBM, below are key design principles application architects should consider while building applications for CICS. [1]

- Simplicity in the API model: CICS offers an extensive set of programming commands that application programs can use to request CICS services. This set of CICS commands is known as API. These commands are macro statements supported in various languages and translated by a CICS translator before compilation. [1]

- Agility for development languages: Traditionally, CICS application development has focused on procedural languages, such as COBOL, PL/I, C, and assembly language. However, CICS

Transaction Server has a range of capabilities supporting technologies such as Java, web services, PHP, and Web 2.0. [1]

- Reliability in the design: Design the application to provide reliability in terms of providing transactional integrity when accessing data and the ability to respond acceptably and reliably to incoming requests. Handle the errors and recovery by taking advantage of a rich set of errors provided by CICS API for all functions. [1]
- Scalability for performance and availability: CICS Transaction Server is designed to coordinate with System Z to provide applications with a scalable application environment. In addition, a rich set of scheduling and administrative controls allows us to control the application's runtime behavior without modifying the business application. [1]

CICS API

- CICS application programs can use CICS API (Application Programming Interface) to request CICS services. Below is the standard format for the CICS API command: [2]
    - 'EXEC CICS commandname option option' [2]
    - These commands can be categorized into the following areas:
        - **Presentation services**: These commands communicate between the user and the application directly or through an intermediate system. These services work with the system's presentation management facilities, such as 3270 display devices, web browsers, etc. [1]
        - **Data services**: These services retrieve and update data and provide access to files, temporary storage or transient data queues, and CICS journals. [1]
        - **Business Services**: These services manipulate data from when it is retrieved from storage to when it is either presented or updated and cover a wide range of functions. [1]

CICS data objects

CICS provides several objects that CICS application programs can use for transaction processing. [2]

- **Communication Areas (COMMAREA)**: COMMAREA is a single block of storage used to pass data between programs. The recommended maximum size of communication areas is 24 KB, although the absolute maximum size is 32 KB. [2]
- **Channels and Containers**: A container is a named block of data that passes information between programs. A channel is a logical grouping of multiple containers. Containers don't have the hard limit of 32 KB. [2]

- **Temporary storage queues**: A temporary storage queue is a named queue of data items that can be written, rewritten, read, and reread in any sequence. The maximum size is limited to 32 KB. [2]

- **Transient data queues**: A transient data queue is a named queue of data items that can be written and read only once. The maximum size is limited to 32 KB. [2]

Transaction Processing

Large server computers like CICS typically support transaction processing to support interactive applications. In transaction processing, work is divided into individual, invisible operations called transactions. [2]

Below are some key transaction components to building and running CICS applications.

- **Transaction**: A transaction is a piece of processing initiated by a single request. An end user usually makes transactions at a terminal, but they might also be made from a web page, a remote workstation program, an application in another CICS system, or triggered automatically at a predefined time. A CICS transaction is given a 4-character name defined in the program control table (PCT). [https://www.ibm.com/docs/en/zos-basic-skills?topic=cics-programs-transactions-tasks]

- **Task**: In a CICS system, many instances of a transaction can run at the same time. A single instance of a running transaction is known as a task. [2]

- **Units of work**: The consistency property of a transaction ensures that data is in a consistent state when a transaction starts and when it ends. The recoverable sequence of operations performed by a transaction between two points of consistency is known in CICS as a unit of work. [2]

Below are some of the features CICS supports for transaction processing:

- **ACID Properties**:
    - **Atomicity**: All changes to data are performed as if they are a single operation. That means either all the changes are performed or none of them are. [2]
    - **Consistency**: Data is in a consistent state when a transaction starts and when it ends. [2]
    - **Isolation**: The intermediate state of a transaction is invisible to other transactions. As a result, transactions that run concurrently appear to be serialized. [2]
    - **Durability**: After a transaction completes successfully, changes to data persist and are not undone, even in the event of a system failure. [2]
- **Commit and rollback**: To assure the ACID properties of a transaction, any changes made to data during a transaction must be committed or rolled back. [2]

- When a transaction completes normally, a transaction processing system commits changes to the data, making them permanent and visible to other transactions. [2]
- When a transaction does not complete normally, the system rolls back (or backs out) the changes; that is, it restores the data to its last consistent state. [2]
- Resources that can be rolled back to their state at the start of a transaction are known as recoverable resources; resources that cannot be rolled back are nonrecoverable. [2]

- **Recovery and restart**: CICS continually records information about the region's state and each unit of work. This information is preserved and used when a region is restarted, thus enabling CICS to restart without losing data integrity. If CICS shuts down, leaving any units of work in-flight, these units of work are backed out when CICS restarts, and locks held by these units of work are released. [2]

Task Control Blocks (TCBs)

A task control block (TCB) represents a z/OS task. A running CICS system is a z/OS address space containing multiple TCBs and, hence, multiple z/OS tasks. Within CICS, a CICS task itself runs on a TCB. [2]

A TCB is a dispatchable unit of work in z/OS. The z/OS dispatcher dispatches TCBs. The CICS dispatcher dispatches CICS tasks to run on the QR TCB and other TCBs. [2]

The type of TCB used for transaction processing is determined by how the application program resource is defined within CICS.

Below are different TCBs supported by CICS for processing transactions:

- **QR TCB (quasi-reentrant task control block)**:
  - Before the open transaction environment was implemented in CICS, all CICS tasks running user applications ran on the QR TCB. The tasks were given control by the CICS dispatcher, which dispatched the use of the TCB between the tasks. Programs running on the QR TCB are called quasi-reentrant. [2]
  - A quasi-reentrant program is a program that is in a consistent state when control is passed to it, both on entry and before and after each **EXEC CICS** command. Such quasi-reentrancy guarantees that each invocation of an application program is unaffected by previous runs or by concurrent multithreading through the program by multiple CICS tasks. [2]
- **Open TCBs**:
  - The open transaction environment (OTE) is an environment where threadsafe application code can run on its own TCB inside the CICS address space without interference from other transactions. Applications that use the OTE can run on a class of TCB called an

open TCB or on the quasi-reentrant TCB (QR TCB). [2] [https://www.redbooks.ibm.com/redbooks/pdfs/sg248298.pdf]

- CICS manages open TCBs in separate pools, each containing a different type or mode of open TCB. Each mode has a 2-character identifier to indicate its specific purpose and is handled by CICS differently. [2] Below are different open TCBs used by CICS:
    - L8 & L9 - CICSE uses these modes to run thread-safe programs. [2]
    - SP & S8 - CICS uses these TCBs to manage SSL connections and requests to LDAP by using the DFHDDAPX XPI interface. [2]
    - TP & T8 - the JVM server uses TP to perform system processing, and T8 is used to execute Java programs in the JVM server resource. [2]
    - X8 & X9 - CICS uses these TCBs to execute C and C++ programs compiled with the XPLINK option. [2]

Transaction storage areas

In addition to the data objects discussed above, CICS also provides a variety of storage areas that application programs can use for transaction processing.

Below are a few transaction storage areas:

- **Transaction work area (TWA)**: The transaction work area (TWA) is allocated when a transaction is initiated and initialized to binary zeroes. It lasts for the entire transaction duration and is accessible to all local programs in the transaction context. [2]
- **User storage**: User storage is available to all the programs in a transaction. Application programs need to pass it across to different programs. It can be obtained using the GETMAIN command when the transaction requires it and returned through the FREEMAN command as soon as it is not needed. [2]
- **Program storage**: CICS creates a separate copy of the variable area of a CICS program for each transaction using the program. This area is known as program storage. [2]

Transaction Flow

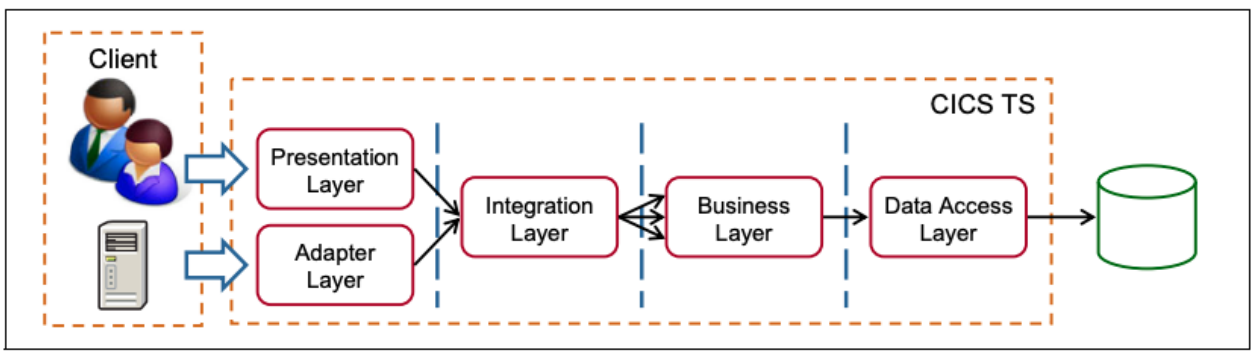

Figure 3: Illustrating the modular application architecture layers. [1]

Below is how the request flows through the key architecture layers of the CICS applications illustrated above:

- **Adapter layer**: Processes the protocols and data with the client, establishes the transaction and security context in CICS, and works with the integration or business layer. [1]
- **Presentation layer**: This layer would be special for web or 3270 applications, where the CICS application provides the user interface. This layer will facilitate communication with the terminal or web. [1]
- **Integration layer:** This layer is optional, and when present, it implements a sequence of calls to business logic to provide better efficiency and encapsulation or makes the services easier to consume between internal CICS application calls and external client calls. [1]
- **Business layer**: This layer implements the business functions of the service and is often the most extensive layer with the most processing requirements. [1]
- **Data access layer**: As the name states, this layer provides the logic to access the data stores in use, such as DB2, VSAM, or other resources.

Below is the high-level flow of how a sample user interaction CICS transaction would function:

1. CICS Transaction is initiated through the terminal or one of the interfaces provided by CICS.
2. Based on the transaction definition, CICS system services will initiate the task using the right TCB based on the PCT definition for the respective transaction.
3. The application program receives the user's input using the CICS API commands.
4. The application will execute the business function by invoking the CICS API commands to access CICS or data resources.
5. The application execution could involve multiple programs invoked using CICS LINK commands by passing the required data using CICS data objects.
6. CICS will monitor the transaction execution and apply commits and rollbacks as necessary.
7. Once the application is done with the processing, it uses the CICS commands again to send the final response back to the user.
8. CICS system services terminate the task attached to this transaction, releasing any resources and issuing commits as needed.

CICS supports multithreading to handle multiple user requests concurrently. CICS maintains a separate thread of control for each task. For example, when one task is waiting to read a disk file or to get a response from a terminal, CICS gives control to another task. The CICS task control program manages tasks. [https://www.ibm.com/docs/en/zos-basic-skills]

Code example

This sample CICS COBOL program reads a number from a 3270 terminal, writes it into a VSAM file, and calls a sub-program with a commarea. The sub-program then writes the data into a DB2 table.

Main Program:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MAINPGM.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 WS-DATA PIC 9(5).
01 COMMAREA PIC X(100).
EXEC CICS DCLGEN (MYDB2TABLE) END-EXEC.
PROCEDURE DIVISION.
EXEC CICS RECEIVE MAP('MAPNAME') INTO(WS-DATA) END-EXEC.
EXEC CICS WRITE FILE('VSAMFILE') FROM(WS-DATA) LENGTH(5) END-EXEC.
MOVE WS-DATA TO COMMAREA.
EXEC CICS LINK PROGRAM('SUBPGM') COMMAREA(COMMAREA) LENGTH(100) END-EXEC.
EXEC CICS RETURN END-EXEC.
STOP RUN.
```

Sub-Program:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. SUBPGM.
DATA DIVISION.
LINKAGE SECTION.
01 DFHCOMMAREA.
  05 CA-DATA PIC 9(5).
PROCEDURE DIVISION.
EXEC SQL INSERT INTO MYDB2TABLE (DATA_COLUMN) VALUES (:CA-DATA) END-EXEC.
EXEC CICS RETURN END-EXEC.
STOP RUN.
```

Sample code Explanation

1. **Reading Input:** The main program uses EXEC CICS RECEIVE MAP to read input from the 3270 screen.
2. **Writing to VSAM:** The program writes the input number to a VSAM file.
3. **Calling Sub-Program:** The input is moved to a COMMAREA and passed to the subprogram.
4. **DB2 Write:** The sub-program receives and writes the data into a DB2 table.

Conclusion

CICS remains the backbone for big enterprise transaction processing and running mission-critical workloads, with solid legacy support and high-end API and integration capabilities. The platform continues to advance to address modern enterprise requirements, including RESTful services, container integration, and advanced workload management.

Future research topics must encompass container-native CICS deployment, more advanced monitoring using AI-driven observability, and improved security integration for future hybrid cloud environments.

Additionally, additional investment in enhancing zIIP offloading efficiency
and opensource support enhancement will help keep the vitality of CICS in digital transformation initiatives.

References

[1] IBM Corporation, "Architect's Guide to IBM CICS on System z" IBM Redbooks, 2012.

[2] IBM Knowledge Center, "CICS Transaction Server for z/OS 5.2" IBM, [Online]. Available: https://www.ibm.com/docs/en/cics-ts/5.2.Accessed March. 2021

Copyright Notice