## CICS concepts and facilities

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>2</td>
</tr>
<tr>
<td>Typical uses</td>
<td>4</td>
</tr>
<tr>
<td>CICS and VTAM</td>
<td>7</td>
</tr>
<tr>
<td>CICS evolution</td>
<td>8</td>
</tr>
<tr>
<td>User’s view of CICS</td>
<td>11</td>
</tr>
<tr>
<td>CICS structure</td>
<td>12</td>
</tr>
<tr>
<td>Handling a task</td>
<td>15</td>
</tr>
<tr>
<td>Program types</td>
<td>19</td>
</tr>
<tr>
<td>Passing data from one program to another</td>
<td>21</td>
</tr>
<tr>
<td>Integrity and transaction isolation</td>
<td>22</td>
</tr>
<tr>
<td>Security</td>
<td>24</td>
</tr>
<tr>
<td>Operations tasks</td>
<td>25</td>
</tr>
</tbody>
</table>
Introduction

CICS is IBM’s Customer Information Control System. It is a Terminal Control Program or Terminal Monitor or Transaction Processing monitor. Its principal function is to control multiple terminal users and enable them to access their files using a set of programs.

Its original implementation was for a single region or partition (AIO: All In One). Initially, IBM introduced CICS for small and medium mainframe systems with, perhaps, a maximum of 50 terminals. Larger sites would use IMS/DC. As CICS has evolved, the maximum number of terminals has grown to many thousands.

Production CICS regions and most development CICS regions are made non-swappable for performance reasons.

CICS versions include:

- CICS/VS Version 4.1
- CICS Transaction Server Version 1.3 (support ends April 2006)
- CICS Transaction Server Version 2.2 (for all users)
- CICS Transaction Server Version 2.3 (current)
CICS is now IBM’s principal online control system for System/370, System/390 and z/OS environments. It is available for several environments including:

- MVS, OS/390, z/OS
- VSE
- AIX
- HP
- OS/2
- Apple Macintosh

CICS platforms

Mainframes running z/OS, OS/390, MVS, VSE
RS/6000 running AIX
HP/3000
OS/2
AS/400
Apple Mac
Typical uses

**Message switching**

**Batch job submission**
*e.g. print*
Many Data Entry systems collect data from several terminals during daytime CICS sessions. The data can be validated at the time of entry. Subsequently, the data will be sorted (typically) prior to being processed in batch job suites.

Data Entry systems need short response times if they are to be effective. Quite often, the data entered is encoded in some way to reduce data transmission, with the result that the systems are used by skilled keyboard operators.
Inquiry

Inquiry systems are read only and typically use centrally held files.

Inquiry and Update

These systems are the most complex of all, especially if several files are to be updated at once. CICS must also ensure that two users do not try to update the same record at the same time. CICS has recovery routines which protect data integrity if there is a program or system failure before the updates have been completed.
CICS and VTAM

CICS controls the terminal users and their requests, but makes use of VTAM for network connections to the terminals themselves. If VTAM is active when you start CICS, the session between CICS and VTAM is started automatically. If VTAM is not yet active, you see appropriate messages.

Older versions of CICS could use BTAM, which was loaded as part of the CICS region. Unfortunately, a BTAM terminal would then be dedicated to the CICS region, and the terminal would not be dynamically allocated. This led to problems for sites with more than one CICS region (e.g. production and testing), since testers would need a second terminal or a set of coaxial cable switches to access multiple CICS systems.

VTAM does not have this problem since it acts as a software switch and allows a terminal to access multiple applications (TSO, test CICS, production CICS, IMS, etc.). Many sites use a session manager (e.g. TPX) to provide a convenient front-end to VTAM.

![Diagram of CICS using VTAM](image-url)
CICS evolution

CICS Version 1

The first releases of CICS used 24-bit addresses for all system and user programs. This meant that the largest amount of Virtual Storage which could be addressed was 16 Megabytes. All programs, file buffers, and CICS internal storage had to fit inside this amount of Virtual Storage.

BMS maps still have 24-bit addresses.

In practical terms, the address space had to allow for the MVS system area and the MVS Common Area, so the available Private Area which CICS could use was rarely above 11 MB.

The Private Area is used by CICS for its own modules and the user programs and file buffers. Each user is allocated storage as required; when the storage is no longer needed, CICS reclams it. The Dynamic Storage Area is used to service these requests for virtual storage.
Address Space structure for CICS Version 2

When IBM introduced MVS/XA, programs could use 31-bit addressing which meant that each address space could access up to 2048 Megabytes (2 Gigabytes) of Virtual Storage. For compatibility with old systems, the first 16 MB had the same structure as old MVS; areas of virtual storage above 16 MB could be accessed by new programs. The Private Area used by programs also had an Extended Private Area above the 16 MB line.

IBM introduced CICS Version 2 to take advantage of the new Virtual Storage structure and thus provide some Virtual Storage Constraint Relief. This meant that CICS modules and file buffers would be loaded above the line. User programs would therefore have more room below the line; they could only be loaded above the line if they were converted to 31-bit addressing.

CICS now had:

- Dynamic Storage Area (DSA) below 16 MB line
- Extended DSA above 16 MB line

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>TTR</th>
<th>Alias-of</th>
<th>AC</th>
<th>AM</th>
<th>RM</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>.CCINQM</td>
<td>00000448</td>
<td>00042B</td>
<td></td>
<td>00</td>
<td>24</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>.CCMENM</td>
<td>000002C8</td>
<td>000432</td>
<td></td>
<td>00</td>
<td>24</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>.CCUPDM</td>
<td>00000498</td>
<td>000439</td>
<td></td>
<td>00</td>
<td>24</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>.DFHPRGA</td>
<td>000015B0</td>
<td>003917</td>
<td></td>
<td>00</td>
<td>31</td>
<td>ANY</td>
<td></td>
</tr>
<tr>
<td>.PRGCMMSG</td>
<td>00001630</td>
<td>000506</td>
<td></td>
<td>00</td>
<td>31</td>
<td>ANY</td>
<td></td>
</tr>
<tr>
<td>.PRGCTIM</td>
<td>00001310</td>
<td>000514</td>
<td></td>
<td>00</td>
<td>31</td>
<td>ANY</td>
<td></td>
</tr>
</tbody>
</table>
CICS Version 3

CICS concepts and facilities

As MVS evolved, user installations wanted to be able to load data files into storage for faster access. In the case of DB2 tables, the largest single table could be 64 GB, which is already larger than virtual storage for MVS/XA. IBM produced MVS/ESA with the ability to hold more data in storage.

An individual address space is divided into *data spaces*. The first data space has the same structure as MVS/XA for compatibility with older systems; the other data spaces can be used for holding data.

CICS can make use of this extra virtual storage by holding certain files as *data tables*. The records from the target files are read in once and used by any relevant transaction.

**From CICS Version 4 onwards**

CICS has the same structure of virtual storage as MVS/ESA, but now has eight DSAs:

- **CDSA** and **ECDSA** — storage areas for CICS modules
- **RDSA** and **ERDSA** — read-only area (by user programs)
- **SDSA** and **ESDSA** — shared DSA for user programs
- **UDSA** and **EUDSA** — non-shared unique DSA (see later); user programs
User’s view of CICS

Access is through character-based terminals. From the user’s point of view, it appears that he/she is the only user of the system. The user responds to information on a screen by entering values or pressing a PF or PA key. The values are passed to CICS and a new screen is transmitted to the user’s terminal. Ideally, the delay would be less than one second.
CICS concept and facilities

CICS structure

CICS is made up of many functional components which perform the various tasks needed to process the user’s requirements. The components are known as Domains. When a user requests CICS services, the different components are invoked as required.

CICS region

CICS functions are performed by Domains

The domains replace the management modules used by old versions of CICS. Each domain has a two-character identifier. The most important domains are:

KE kernel - the main CICS control structure
DS dispatcher domain
DU dump domain
LC local catalog domain
ME message domain
PA parameter domain
SI systems initialization domain
TI timer domain
XM transaction manager domain
AP application domain: everything else in CICS

e.g. Terminal control, Temporary Storage control, Transient Data control

DM domain manager domain
GC global catalog domain
LM lock domain
MN monitoring domain
ST statistics domain
SM storage domain
TR trace domain
XS security domain

You will see these codes in messages:

DFHSM0115
In addition to the domains, CICS uses lists of **resources** such as files, programs, terminals, etc. and holds them in **control tables**. In the past, the table entries had to be assembled which meant that CICS had to be shut down and restarted before new entries could be added. Now we add files, programs, and other resources through Resource Definition Online which means that CICS can continue to run as enhancements are made to the programs.

Terminals and programs can be auto-installed (i.e. dynamically added)

**CICS Tables identify files, programs, transactions, etc.**

Important internal control tables include:

- **TCT**  
  Terminal Control Table (auto-install)

- **PCT**  
  Program Control Table (transaction codes)

- **PPT**  
  Processing Program Table (progs and maps, may be auto-install)

- **FCT**  
  File Control Table (files and LSR pools)

- **DCT**  
  Destination Control Table (Extrapartition Transient Data queues)

- **SIT**  
  Systems Initialization Table (start-up options)

Resource definitions (including those for CICS-DB2 programs) are held in the CICS system definition file (CSD). The CSD holds definitions for most of the CICS resources.
User data can be held on VSAM files. Normally they would be **dynamically allocated** to the CICS region when they are **enabled** (made available to CICS). If you need to run a batch job which updates or reorganizes a file, you will need to close and de-allocate a file if it is to be processed by a batch program. For most files this will happen automatically when you **disable** the file.

However, if a file is allocated via JCL in the CICS startup deck (the old-style method), you would need to run the ADYN transaction to de-allocate it from the CICS region.

CICS can also access data from IMS and DB2 databases, though these databases are located in different MVS Address Spaces.

When a program makes changes to a record, CICS records the changes in a Journal. CICS uses the z/OS System Logger for journaling activities.
Handling a task

When a user keys a transaction code (e.g. ABCD) and presses ENTER, data is transmitted via VTAM to CICS. The Terminal Control part of CICS determines that there is work to be done, and places the received data into CICS storage.

CICS now schedules a task which will process the user’s request. A task is a unit of work in CICS in much the same way as a job is the unit of batch work in MVS. Each task is allocated a task number.

CICS system services now examine the transaction code entry (in the PCT) to find the name of the first program of this transaction. CICS now checks with the PPT to see if the program is already in storage. If it isn’t already in memory, the CICS Storage Domain obtains enough storage for the program and then Program Control loads the program. The application program now starts to execute.
At some stage, the program may want to read a record from a file before displaying it on the user’s screen. File Control will need to read in the record, first placing it in a buffer in more storage (acquired via the Storage Domain).

Tasks use several storage areas

The program now needs to merge the record details with a screen layout which will be sent to the user. Screen layouts are called MAPs in CICS. Each map is held as a program (Assembler). The map will have to be loaded into storage - more storage which must be acquired via the Storage Domain.

The program now sends the map to the user and returns control to CICS. The task has completed its purpose, so CICS releases the storage areas which were used by the program.

When the user next presses ENTER, CICS will start a new task to continue the ABCD transaction. Tasks are typically very short (sub-second).
If a second user starts the ABCD transaction, CICS will use the same copy of the program, but will acquire a separate copy of the program’s data areas. In this way, tasks do not interfere with each other’s data. Each user gets the same copy of the program: the program is said to be **reentrant**. (Actually, they are quasi-reentrant.)

**CICS programs are reentrant**

If two or more users are processing the same program code, typically they will be at different parts of the program. CICS allows them to find their own ways through the program and is said to be **multi-threading**. When one task is delayed (by asking CICS to carry out some function) another task can execute the program.
CICS can handle several tasks at a time and is regarded as a **multi-tasking** system. If a task requests I/O from a file, that task will be **suspended** until the I/O request has been completed when it will once again become **active**.
Program types

There are two types of programs running on CICS:

- conversational
- pseudo-conversational

With a conversational program, the program remains in memory and retains all its storage areas across terminal I/O. This means that you will see the same task number for the transaction. This type of program is not common because:

- the program size tends to be large
- CICS cannot release the resources during terminal I/O
- storage requirements increase as the program may be idle for many seconds or even minutes

![Conversational program diagram](image-url)
With pseudo-conversational design, a big program is divided into several smaller modules. When a screen is sent to the terminal, the program terminates. This means that CICS can use the resources for other tasks, and is therefore a more efficient approach at run-time. The programmers will therefore develop several small programs which perform the overall task of a large program.

**Pseudo-Conversational program**

![Pseudo-Conversational program diagram](image)

Programs can be written in several languages including:

- Assembler
- C/C++
- LE COBOL
- COBOL for MVS and VM/370
- Enterprise COBOL for z/OS and OS/390
- Java
- PL/1
- REXX
Passing data from one program to another

One consequence of the pseudo-conversational approach is the problem of passing information from one program to another across terminal I/O. Remember: at the end of a task, CICS releases the storage areas used by that task.

The available options include:

- hold it in a TS queue (Temporary Storage)
- place it in a TD queue (Transient Data)
- use a special storage area in CICS (COMMAREA)
- hide the data on the screen, transmit it to the terminal, and collect it again when the user continues
- use of containers and channels (CICS TS V3)

It is difficult to take the last method seriously: data transmission is increased in both directions, the hidden data limits the displayable part of the screen, and if the user presses the CLEAR key the data will be lost! Amazingly, an IBM product once used this approach (but nobody uses that product any more ...).
Integrity and Transaction Isolation

In an MVS system, regions are protected from interference by other regions. Within a single region, a program is prevented from modifying parts of storage which belong to MVS by using Storage Protect Keys. For example, Storage Key 9 (key-9) is open key storage which can be updated by any task, and key-0 is read-only storage used by reentrant programs and tables.

The problem which arises within CICS is that many user programs and CICS routines are processed as tasks in the same region. This means that the MVS Storage Protect Keys method cannot distinguish between one CICS processing program and another.

In early versions of CICS, it is feasible for one task’s program to overwrite and thus corrupt the storage belonging to a different task (or even its own storage). As a means of detecting such a problem, each storage area is given some flag bytes at the beginning and end. CICS subsequently checks that the bytes match. If they do not, CICS knows that there has been a problem and declares a STORAGETAGE VIOLATION.

Detecting Storage Violations

This method is simplistic, and does not detect a program which corrupts storage but does not overwrite the flag bytes.
Introduced with CICS Version 4, Subspaces provide a means of preventing Storage Violations in many (but not all) circumstances.

The **Base Space** is the entire region; programs running there have access to all the storage in the address space. For transactions defined with EXECKEY(CICS). [CDSA and ECDSA]

The **Common Subspace** allows transactions to share access to each other’s storage areas. For transactions defined with ISOLATE(NO). [SDSA and ESDSA]

Other **Subspaces** are unique to individual transactions which can only process their own storage areas. For transactions defined with ISOLATE(YES). [UDSA and EUDSA]

**Transaction Isolation uses Subspaces**

The CICS SIT must contain TRANISO=YES and STGPROT=YES for Transaction Isolation to be active.

Transaction Isolation imposes an overhead of about 2%. Each Subspace needs about 9K of Page and Segment Table space.

With Transaction Isolation, a task’s storage is allocated in DSA units of 1 MB (corresponding to an MVS segment).
Security and RACF/ACF2/Top Secret

CICS uses the MVS security system for controlling access. The users sign on to CICS with their normal password. CICS calls RACF (or ACF2, Top Secret) to obtain a security profile for the user; RACF will then determine whether the user can start transactions. CICS blanks out the password so that it does not show up in a dump.

Normally, the user only needs to enter the password once, though it is possible for a program to request password verification. This could occur, for example, with a salary update program which can only be processed by a payroll supervisor; password verification prevents unauthorized use if the supervisor has left the room.

Security systems such as RACF and ACF2 can control access to resources including:

- CICS transactions
- programs
- files
- transient data
- temporary storage

Resource RACF profiles are held in CICS storage. If they are changed, you will have to **refresh** them for the active CICS system. This is automatic with RACF 2.1 and later releases.

If access is denied, the task is terminated (**not** the CICS address space).
Operations tasks

The CICS systems are controlled by a Master Terminal Operator who would sign on at a CICS terminal and issue commands from there. It is also possible to issue CICS commands from the operator console using the Modify command (see later.)

The Master Terminal Operator controls CICS

The MTO is usually a member of the operations or networking department and is responsible for:

- operating CICS
- interfacing between the users and the operations department
- performing diagnostic procedures

Special transactions (e.g. CEMT) are provided by CICS for controlling resources. The main operations functions include:

- enable/disable transactions and programs
- list active tasks
- terminate a task
- change the status of terminals
- inspect or change the status of a dataset
- obtain statistics from CICS

CEMT can also be used by programmers.
CICSplex

Installations which need to provide 24 x 7 access to online systems may implement a CICSplex. Multiple CICS systems are inter-connected in a way which enables the workload of a failing CICS region to be taken over by other regions in the CICSplex.
Accessing CICS

Users can access CICS applications via several techniques:

- 3270 Display System for access from standard mainframe terminals
- ISC (Inter System Communication) or MRO (Multi Region Operation) for access from other CICS systems
- CICS Sockets feature for access from UNIX systems
- EXCI (External CICS Interface) for applications running in MVS address spaces
- CICS Web Interface for access from web browsers
- ONC RPC Support for access using Open Network Coupling with Remote Procedure Calls
- TN3270 Support for Telnet clients
- Using MQ message queues
CICS and the Web

CICS TS 1.3 enhanced CICS Web Support (CWS) and enabled programmers to create and manage web pages and other blocks of data. Applications can interpret data from an HTTP request (Hyper Text Transmission Protocol) and construct an HTTP response.

CWS is managed and enabled using standard CICS transactions and interfaces such as CEMT, CEDA, and SPI.

CICS Transaction Gateway

The CICS Transaction Gateway enables web browsers and internet-enabled applications to access CICS applications using various methods:

- Converting existing 3270 applications into HTML and transmitting to the browser using HTTP
- Enabling Java applets to access 3270 applications using Java classes and Java beans
- Using CORBA IIOP protocol (Common Object Request Broker Architecture Internet Inter-ORB Protocol). ORB-enabled browsers can run Java beans which interact with server-side beans which use Java classes to process applications