# Section 1

## Introduction to CICS

Before you can start learning the details of coding CICS programs, you need to understand some basic CICS concepts and terms. So in chapter 1, you'll learn about CICS and the services it provides for application programs. Then, in chapter 2, you'll learn how CICS programs work and how you access CICS services from within a COBOL program. This chapter includes a complete interactive application so you can see from the start what's involved in developing CICS programs. When you complete these two chapters, you'll have a good foundation for learning the details of CICS programming that are presented in the remainder of this book.

**Mike Murach & Associates**

2560 West Shaw Lane, Suite 101
Fresno, CA 93711-2765
(559) 440-9071 • (800) 221-5528

murachbooks@murach.com • www.murach.com

# 1

# CICS concepts and terms

This chapter presents the concepts and terms that you need to know when you develop CICS programs. After you're introduced to CICS and its environment, you'll learn how CICS can handle hundreds of users at the same time. Then, you'll learn about the CICS services that you use as you develop COBOL programs that issue CICS commands.

# An introduction to CICS

The CICS environment has evolved over the years since it was first introduced in 1968. At that time, it ran only on IBM mainframes and in a limited capacity. Today, CICS can be used on a variety of platforms and with several different programming languages. And with the introduction of IBM's CICS Transaction Server 1.3, CICS became the premier server for Internet applications.

## What is CICS?

CICS, which stands for *Customer Information Control System*, is a world-class transaction processing system. As such, it can process the transactions from hundreds of users at the same time as they run a variety of application programs. CICS loads those programs, coordinates their execution, manages the data transmissions between programs and terminals, controls the access to data, and maintains the integrity of that data. Because CICS is a transaction processing system, it can be called *online transaction processing* (*OLTP*) software.

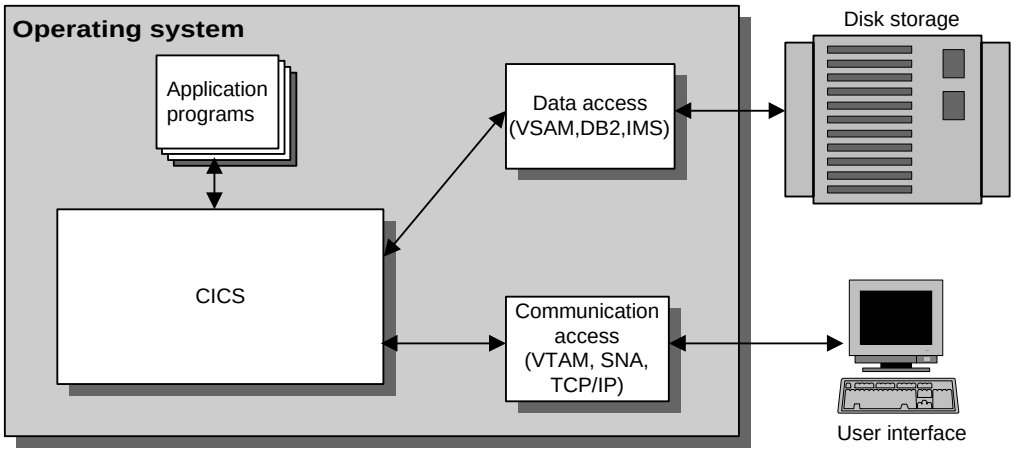
As you can see in the first drawing in figure 1-1, CICS acts as an interface between the application programs and the operating system's services. So when the application program wants to access a terminal or a disk device, it doesn't communicate directly with the device. Instead, it issues commands to communicate with CICS, which communicates with one of the operating system's access methods. Then, the access method communicates with the device. This shelters your application programs from specific device and operating system details.

CICS is needed with operating systems like OS/390 and VSE/ESA because these operating systems were designed to run batch programs, not online programs. As a result, these operating systems work best when a small number of jobs are running at the same time and when each of these jobs has exclusive use of the data files or other resources it requires. In contrast, an online program may need to provide for hundreds of concurrent users who need to share the same files and resources.

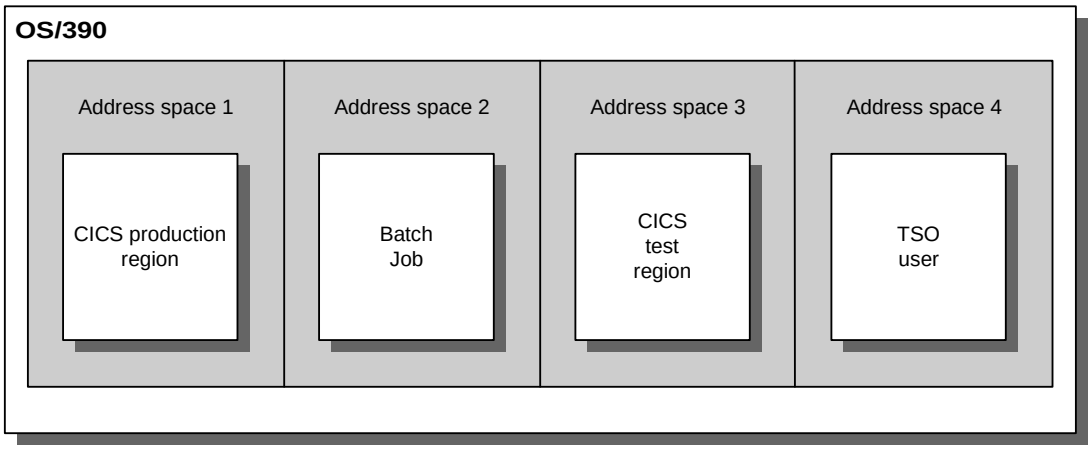
To a large extent, then, CICS is an operating system in itself because it provides many of the functions normally associated with an operating system. For example, CICS manages its own processor storage, provides its own file management functions, and includes a task manager that handles the concurrent execution of multiple programs. So when CICS runs under OS/390 or VSE/ESA, you can think of CICS as an operating system within an operating system.

The second drawing in this figure shows that CICS actually runs as a batch job within the OS/390 operating system. As a result, two or more CICS systems, also called *regions*, can run at the same time. In this example, OS/390 has assigned address spaces to two different CICS regions that are running at the same time. Then, the CICS production region can handle all CICS production programs, while the CICS test region can be used to test new or modified CICS programs. This is a common setup that protects production programs from failures caused by programs that are still being tested.

## A CICS interface




## CICS in an OS/390 address space




## Description

- The *Customer Information Control System*, or *CICS*, is designed to control information in a modern online environment by providing database and data communication functions.
- CICS provides an interface between application programs and operating system services, such as data access and communication access.
- On a mainframe, CICS runs under an operating system like OS/390 or VSE/ESA. Although CICS runs as a batch job, it can handle hundreds of interactive users that are using a variety of applications.
- More than one CICS system (or *region*) can run on the same computer at the same time. Because each region runs in its own address space, programs running in one region won't interfere with programs running in another region.

Figure 1-1 What is CICS?

## CICS platforms and programming languages

Today, CICS can run on the platforms shown in the first table of figure 1-2. By far, the most popular of these is the S/390 running the OS/390 operating system. On a large system like that, CICS can handle millions of transactions per hour. Note, however, that CICS can also be used on smaller systems like AS/400s, RS/6000s, and even PC servers.

The second table in this figure lists the six languages you can use for developing CICS programs. Of these, COBOL is by far the most popular with billions of lines of COBOL code currently in operation. Because COBOL provides a structured language that is easy to understand and maintain, COBOL is likely to remain the most popular language for developing CICS programs for many years to come.

On the other hand, now that languages like Java can be used for CICS applications, some shops are using them to develop the user interface portion of new applications (they can be used to develop entire CICS applications, but in practice, that rarely happens). In that case, though, COBOL is still the language that's most likely to be used for the business processing that the application requires. This modular approach is also a way to provide a new *graphical user interface* (*GUI*) for an existing CICS/COBOL application. You'll learn more about this in chapter 20.

Although CICS can be used on all the platforms shown in this figure, this book focuses on the way CICS is used on IBM mainframes, especially those running under the OS/390 or MVS operating system. That, of course, is the most used platform today. Keep in mind, though, that CICS itself works the same way on all of the platforms that support it.

## CICS transaction server platforms

| Hardware | Operating system |
|---|---|
| IBM zSeries 900 | z/OS |
| IBM S/390 | OS/390, MVS, VSE |
| IBM AS/400 | OS/400 |
| IBM RS/6000 | AIX (UNIX) |
| PC server | Windows NT, OS/2 |

## CICS programming languages

| Language | Description |
|---|---|
| COBOL | The most used programming language for both batch and online applications in a mainframe environment. Today, an overwhelming majority of CICS programs are in COBOL. |
| Assembler language | Assembler language was fairly popular in the 70s and early 80s before COBOL became the dominant business language. Today, assembler language is still used for special-purpose devices like ATM machines. |
| PL/I | In the 70s and 80s, PL/I was an alternative to COBOL that was used by a small percentage of CICS shops. Today, you probably won't find any new program development being done in PL/I. |
| C and C++ | Traditionally used in engineering and math environments, C and C++ can now be used to access CICS services. With C++, an object-oriented language, you can develop object-oriented classes that access CICS services. |
| Java | The newest language for writing CICS applications, Java can also be used for developing object-oriented classes that access CICS services. |

## Description

- Although CICS runs on several computing platforms, the most used platform today is the S/390 mainframe running the OS/390 or MVS operating system.
- Although CICS programs can be written in several languages, the vast majority of CICS programs today are written in COBOL.

Figure 1-2 CICS platforms and programming languages

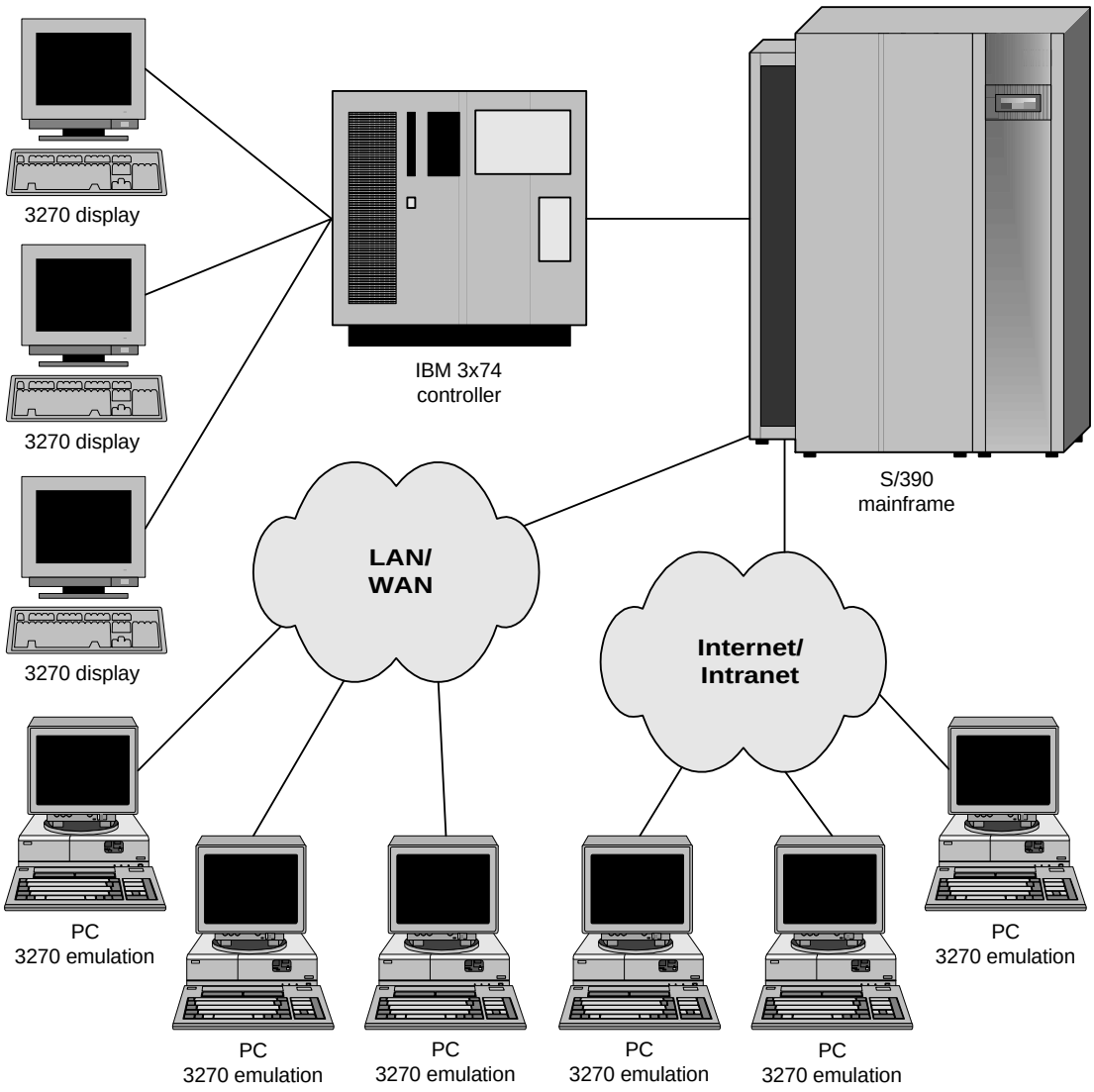
## The 3270 information display system

The *3270 display station* (or *terminal*) has been the standard workstation for CICS systems since 1971. The 3270 information display system includes terminals (display stations), printers, and controllers. As figure 1-3 shows, the terminals access the mainframe through IBM 3x74 controllers, which can provide for anywhere from 32 to 64 terminals. So a large system consists of many of these controllers.

Although 3270 terminals were common in the 1980s and early 1990s, you won't find too many mainframe shops using them any more. Today, *3270 emulation* software makes it possible for personal computers (PCs) to be used as 3270 terminals. That way, you not only have access to CICS applications but also to PC applications like word processing and spreadsheet software.

In this figure, you can see two ways to connect PCs with 3270 emulation software to CICS. One way is through a *Local Area Network* (*LAN*) or a *Wide Area Network* (*WAN*). The other way is through the Internet or an intranet. Either way, the 3270 emulation software interprets data streams sent from CICS and constructs a 3270 screen display on a PC's monitor, usually in a window. IBM includes 3270 emulation software as part of its CICS software package, but you can also purchase 3270 emulators from other companies. To test the examples in this book, I used an emulator called *EXTRA! for Windows 98* that accessed CICS at a remote site over the Internet.

## The 3270 information display system




## Description

- The 3270 family of terminal devices has been the standard for IBM mainframe computers since 1971. They can be connected to a mainframe directly or remotely via a 3x74 controller.
- Today, the typical user connects to CICS via a PC running some form of *3270 emulation* software. This software interprets 3270 data streams sent through a network and constructs a 3270 display screen, usually in a window on the PC's desktop.
- When you use 3270 emulation on a PC, the PC can connect to CICS through a *Local Area Network* (*LAN*) or a *Wide Area Network* (*WAN*) or through the Internet or an intranet.

Figure 1-3 The 3270 information display system

## Alternate user interfaces

Today, the vast majority of CICS applications are written for 3270 display terminals or PCs that are emulating 3270 terminals. But as figure 1-4 shows, CICS can also be used with *front-end programs* that are written in other languages. In this case, the front-end program provides the user interface, or *presentation logic*, while the *back-end program* (the CICS application) provides the *business logic*.

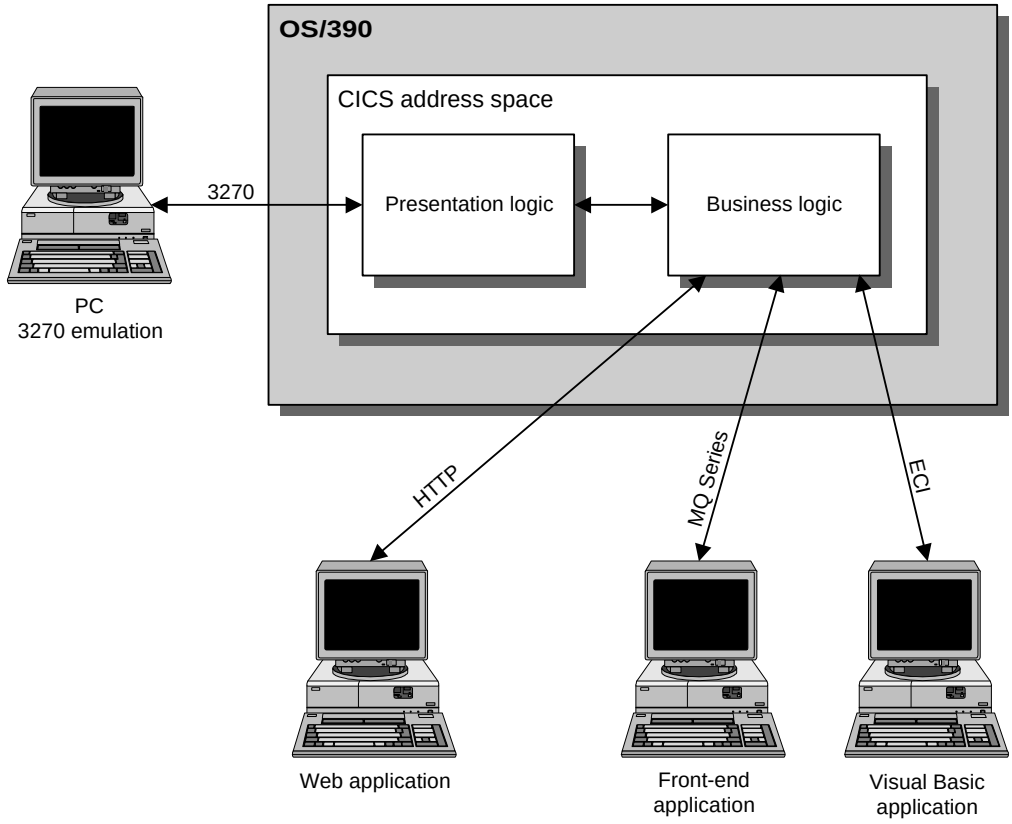
As this figure shows, if you use 3270 terminals or 3270 emulation, the CICS application provides both the presentation logic and the business logic. In this case, the user interface is text only, not graphical. Also, all of the processing is done by the CICS application because 3270 terminals are "dumb" terminals, which means they can't do any processing.

If you want to provide a graphical user interface for an application or if you want the front-end program to do some of the processing, you can use one of the three alternate user interfaces shown in this figure. In this case, the front-end program provides the GUI and perhaps some processing, while the CICS application provides the business logic.

If you write the front-end for an application in a language like Visual Basic or Java, the front-end can use the CICS *External Call Interface*, or *ECI calls*, to send data to and from CICS. If you want to pass data to and from a program on another platform, you can use IBM's *MQSeries*, which allows disparate systems to exchange information through *message queuing*. If you want to access a CICS application from a Web application (an application that runs in a Web browser), you can do that in several different ways, including using standard *HTTP* (the *Hypertext Transfer Protocol*).

Note in all three cases that CICS is used for the business logic of the application. This insures that the same business rules are applied to the user requests for data no matter how the user accesses the system. In chapter 20, you'll learn how to write CICS programs with separate presentation and business logic so you can take advantage of these alternatives. And in chapter 21, you'll learn how CICS can be used for developing Web applications.

## Alternate user interfaces




## Description

- When you use 3270 terminals or PCs running 3270 emulation software, the CICS application provides the *presentation logic* for the terminals or PCs. That means that CICS determines what screens will be displayed and how they will work. In this case, CICS also provides the *business logic* that determines how the user requests will be processed.
- CICS can also be used with other types of user interfaces. In this case, the *front-end program* provides the presentation logic, but the CICS application is still the *back-end program* that provides the business logic.
- Visual Basic is one product that can be used to develop a front-end program with a graphical user interface (GUI) that accesses a back-end program written in CICS. To access CICS programs, the Visual Basic program issues *ECI calls*.
- A front-end program written for another platform can retrieve data managed by CICS by issuing message requests through an IBM product called *MQSeries*. This *message queuing* product lets you send data between two disparate computer platforms.
- Web applications run in a Web browser and can access CICS programs and services through an Internet protocol like *HTTP*.

Figure 1-4 Alternate user interfaces

# How CICS manages multiple users

Now that you've been introduced to CICS, you're ready to understand how CICS is able to manage the processing for hundreds or thousands of users at the same time. To do that efficiently, CICS uses features called multitasking and multithreading.

## How multitasking and multithreading work

In CICS, a *task* is the execution of an application program for a specific user. For every execution of a program, a task is started. For example, if User 1 is running an application program under CICS, then User 1 has created a task.

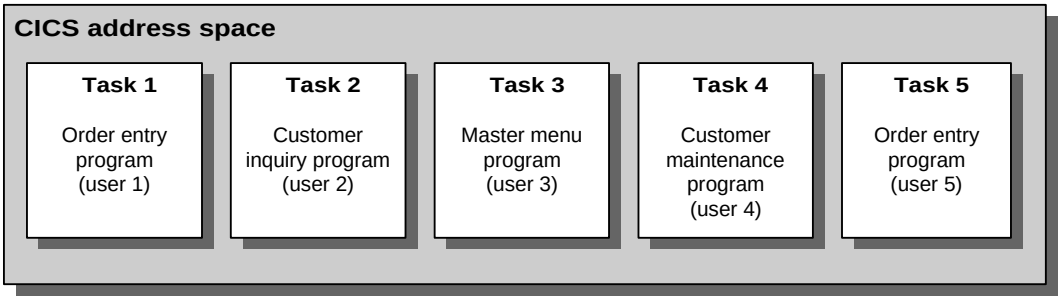
One of the basic features, and strengths, of CICS is its ability to multitask. *Multitasking* means that CICS allows more than one task to be executed at the same time, which is essential in an environment where hundreds of users can be logged into CICS at the same time. In figure 1-5, for example, you can see a CICS address space that has five tasks running, one for each of the five users currently logged on to the system.

Although all of the operating systems that support CICS are capable of multitasking on their own, CICS provides for multitasking within the single address space provided by the operating system. In other words, it ignores the multitasking capabilities of the operating system. As a result, multitasking works the same under CICS no matter which operating system you use.

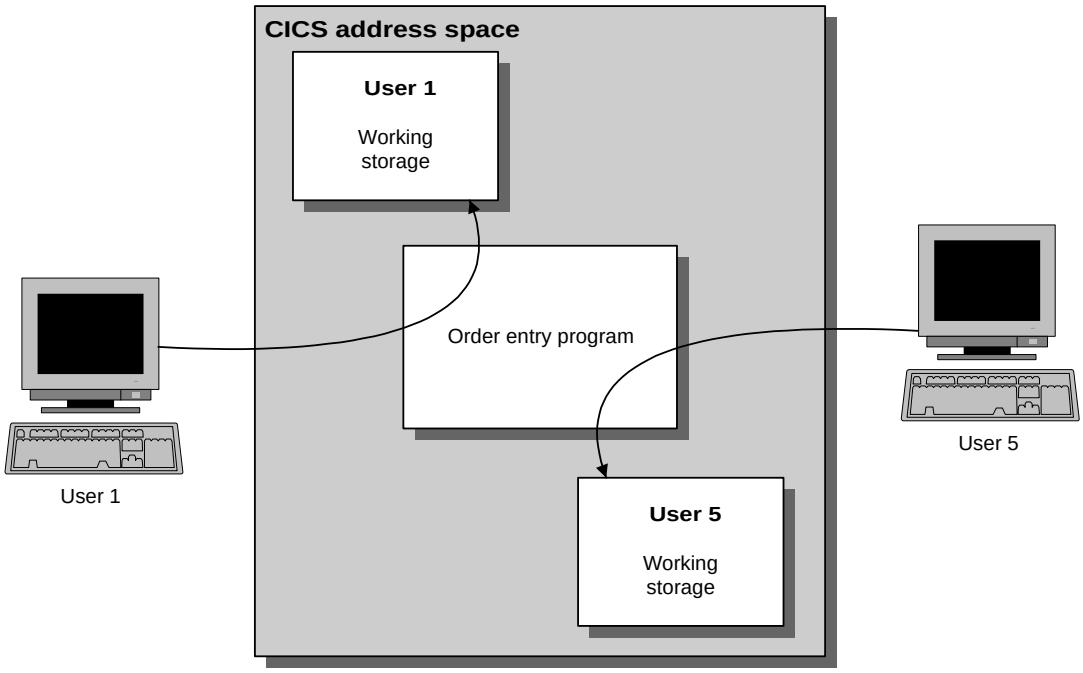
In contrast to multitasking, *multithreading* allows all the users in a CICS region to use the same copy of a program at the same time. In this figure, for example, you can see that both User 1 and User 5 are running the order entry program. In this case, if both users had their own copies of the program, valuable internal storage space would be used unnecessarily. With multithreading, though, only one copy of a program is loaded and shared by all.

For multithreading to work, the program must be *reentrant*. A program that's completely reentrant doesn't change itself in any way. In other words, a truly reentrant program cannot modify data in working storage. Obviously, though, COBOL programs that can't use working storage would be difficult to write. So CICS provides a separate copy of working storage for each user running a program. As you can see in this figure, the users share the same copy of the program's executable code, but each is given a separate working storage area. Once the program finishes executing, the working storage for that user is released and the virtual memory that was used is free to be allocated to another user.

## CICS uses its address space to support multitasking




## Multithreading provides a separate copy of working storage for each user




## Description

- A *task* is the execution of an application program for a specific user. Within CICS, two or more tasks can execute at the same time using a CICS feature called *multitasking*.
- Although all of the operating systems that support CICS provide multitasking of their own, CICS handles multitasking internally. This provides for a stable system that works the same way under any operating system.
- CICS also provides a feature called *multithreading*. With multithreading, two or more users can access the same copy of a program at the same time. CICS accomplishes this by providing a separate copy of working storage for each user running the program.

Figure 1-5 How multitasking and multithreading work

## How transactions work

Under CICS, a user cannot directly invoke a program. Instead, the user invokes a *transaction*, which in turn specifies the application program to be run. When a user invokes a transaction, CICS locates the application program associated with the transaction, loads it into storage (if it isn't in storage already), and starts a task. The difference between a task and a transaction is that while several users may invoke the same transaction, each is given a separate task.

Each transaction is identified by a unique four-character code called a *transaction identifier*, or *trans-id*. A user initiates a transaction by entering the transaction identifier at the terminal. If, for example, the user keys in the characters ORD1 and presses the Enter key, the transaction named ORD1 is invoked.

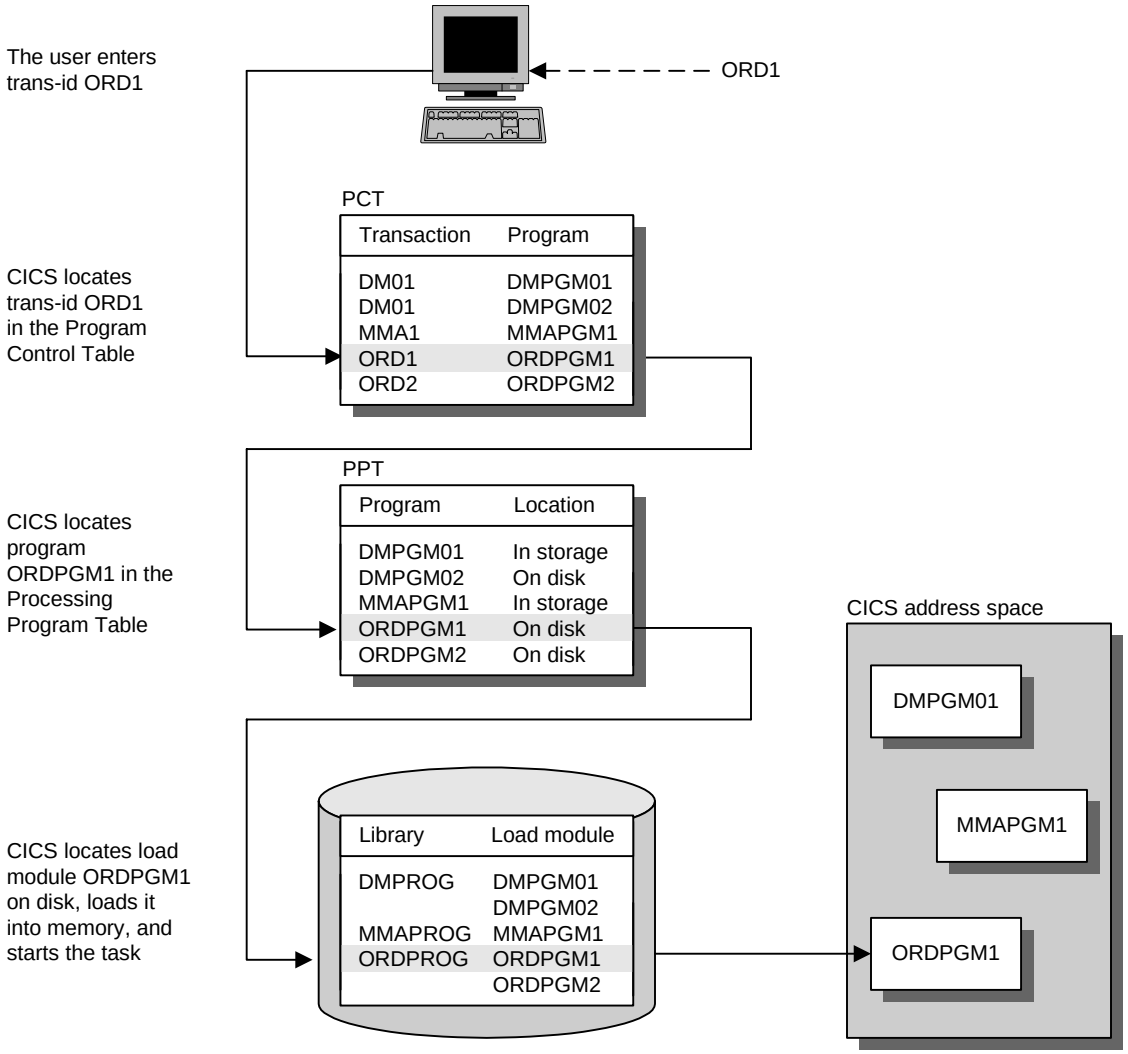
Every transaction must be defined in a special CICS table called the *Program Control Table*, or *PCT*. Basically, the PCT is a list of valid transaction identifiers. Each trans-id in the PCT is paired with the name of the program CICS will load and execute when the transaction is invoked.

Another CICS table called the *Processing Program Table*, or *PPT*, contains a list of all valid program names. This table keeps track of which programs are located in storage. CICS uses it to determine whether a new copy of a program needs to be loaded into storage when a transaction is invoked.

CICS creates these internal control tables based on *resource definitions* created by systems programmers. To create these definitions, a systems programmer can use a batch program for that purpose or an interactive program called *Resource Definition Online* (*RDO*). Because RDO lets the systems programmer define resources such as transactions and programs from a CICS terminal, it is the preferred way to create resource definitions. (Because the trans-id for RDO is CEDA, RDO is sometimes referred to as *CEDA*.)

To review the way transactions work, please refer to figure 1-6, which shows how a task is initiated under CICS. Here, a user enters the trans-id ORD1. Then, CICS searches the Program Control Table to find the program to be executed. As you can see, the program for transaction ORD1 is ORDPGM1. Next, CICS searches the Processing Program Table to determine if the program is currently in main storage. In this case, it isn't, so CICS locates the program on disk, loads it into storage, updates the PPT, and initiates a new task.

## How CICS invokes an application program




## Description

- Each *transaction* is identified by a unique, four-character identifier called a *transaction identifier*, or *trans-id*. To start a transaction, the user enters its transaction identifier at a terminal.
- Transactions are defined in a CICS table called the *Program Control Table* (*PCT*). Each trans-id in the PCT identifies the program CICS should execute when the transaction is invoked.
- Programs are defined in the *Processing Program Table* (*PPT*). This table keeps track of which programs are already loaded into storage. If a program has not been loaded into storage, it's loaded when its associated transaction is invoked.

Figure 1-6 How transactions work

# CICS services

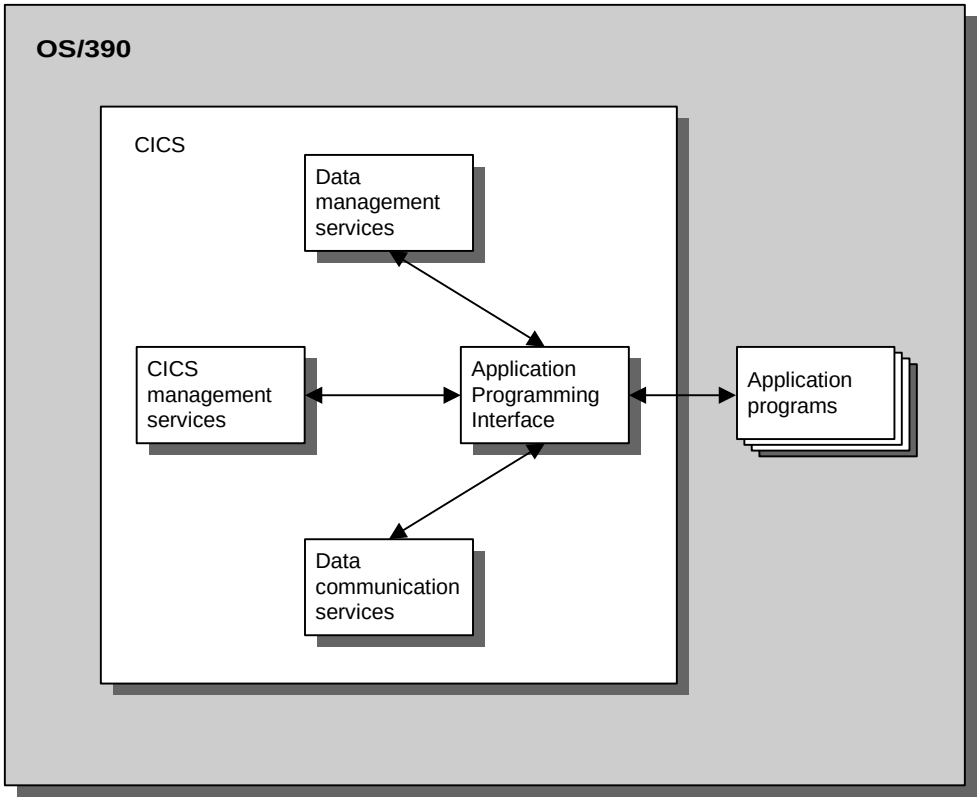
To help the programmer develop application programs, CICS provides many services that are available to application programs. These services let a program communicate with terminals, access data files, use operating system features, and so on. The next four topics introduce you to these services.

## The Application Programming Interface

To access CICS services from an application program, you use the CICS *Application Programming Interface*, or *API*. This is illustrated in figure 1-7. Here, an application program communicates with the API, which in turn communicates with the individual CICS services. In this way, the API insures that all of the CICS services are invoked in a consistent manner.

The primary focus of this book is to teach you how to code CICS commands within application programs to access CICS services through the API. Because CICS programs are written predominantly in COBOL, this book presents the CICS commands in the context of COBOL programs. Keep in mind, though, that you can use these commands in programs written in other languages, too.

## How an application program accesses CICS services




## Description

- CICS provides many services that are available to an application program through its *Application Programming Interface*, or *API*. This API insures that all the services are invoked in a consistent manner.
- The data communication services allow users to communicate with programs through their terminals.
- The data management services let programs access data stored in files or databases.
- The CICS management services provide a variety of services for managing the execution of application programs.

Figure 1-7 The Application Programming Interface

## Data communication services

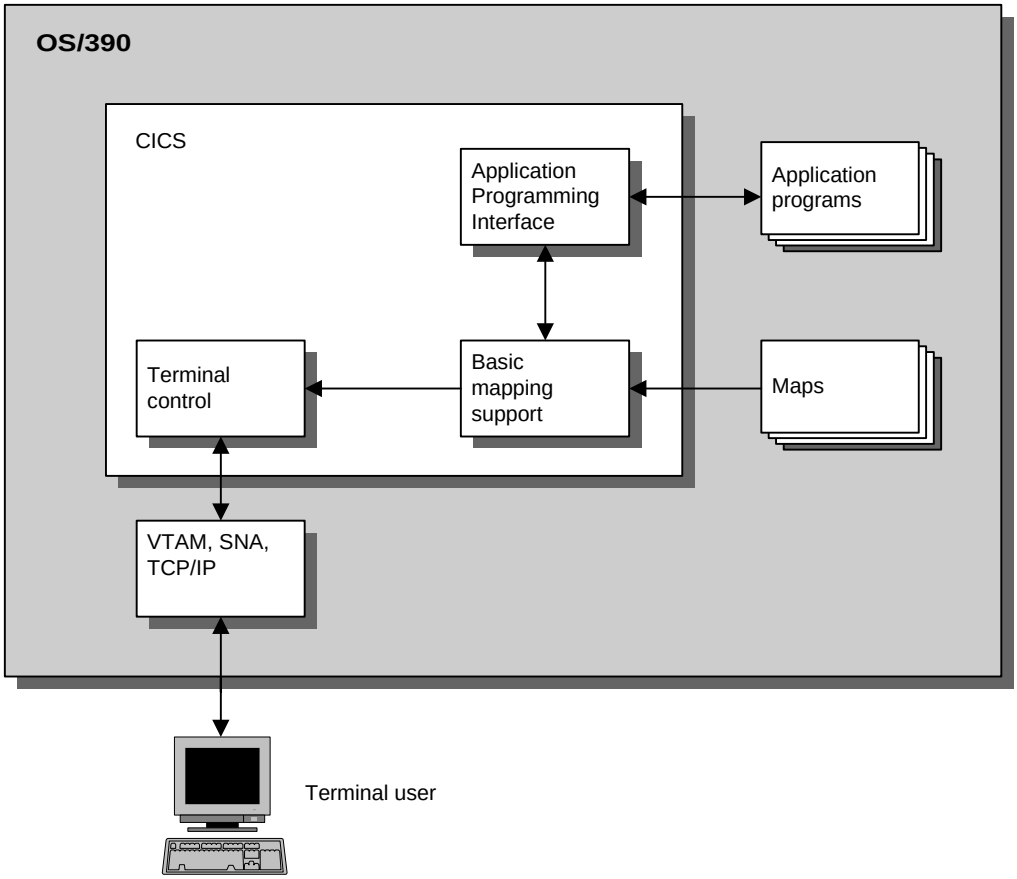
CICS's *data communication services* let a CICS application program communicate with terminal devices. Typically, these services send information to a terminal screen or retrieve user input from it. As you can see in figure 1-8, terminal control and basic mapping support are the two CICS services in this group.

*Terminal control* is CICS's interface with the operating system's telecommunication access method. Terminal control lets you send text to or receive text from the terminal that initiated the task. Although terminal control handles most of the details of working with the access method, it is still difficult for your application program to use it directly. For instance, an application program that uses terminal control directly must process complicated strings of control characters and data sent to and received from the terminal.

To relieve you of the task of building and decoding complicated strings of control characters and data, *basic mapping support*, or *BMS*, was developed. As this figure shows, BMS is an interface between application programs and terminal control. BMS lets you create a *map* that specifies the format of data as it appears on the terminal display. To receive data from or send data to a terminal, an application program issues a BMS request. After BMS retrieves the specified map, it creates a terminal control request that will process the data according to the format specified by the map.

A special CICS table called the *Terminal Control Table*, or *TCT*, is used to define each terminal to the CICS system. In the TCT, each terminal is given a unique one to four-character *terminal identifier*, or *term-id*. These definitions make it possible for CICS to direct program output to the correct terminal and any input from the terminal to the correct program. Here again, creating and maintaining the TCT is the responsibility of CICS systems programmers.

## How an application program communicates with terminal devices




## Description

- *Terminal control* provides the interface between CICS and the operating system's telecommunication access method. It lets you send text or receive text from the terminal that initiated the task.
- The most common telecommunication access methods are *VTAM* (the *Virtual Telecommunication Access Method*), *SNA* (*Systems Network Architecture*), and *TCP/IP* (*Transmission Control Protocol/Internet Protocol*). Some of these access methods can also be used in combination with each other.
- *Basic mapping support* (*BMS*) provides the interface between application programs and terminal control. It lets you create *maps* that specify the position and the characteristics of the individual display elements on the terminal screen. As a result, you can create interfaces that are easy for users to work with.
- Each terminal in the CICS system must be defined in the *Terminal Control Table* (*TCT*). Each terminal in this table is assigned a unique one- to four-character *terminal identifier*, or *term-id*.

Figure 1-8 Data communication services

## Data management services

Figure 1-9 shows that CICS's *data management services* consist of file control, SQL, and DL/I. SQL and DL/I are interfaces to IBM's relational and hierarchical database managers (DB2 and IMS respectively), while file control provides the interface for VSAM files.

The data management service you'll use the most will probably be the *file control* service. As you can see in this figure, when an application program issues a file control request, file control passes it on to VSAM, which manages the data stored on direct access devices.

One of the major responsibilities of the file control service is to manage shared access to files so two or more users can't update the same record at the same time and thus corrupt the data. To provide for this, the file control service locks a record when a user accesses it for updating. As a result, other users can't access the same record until the update is complete.

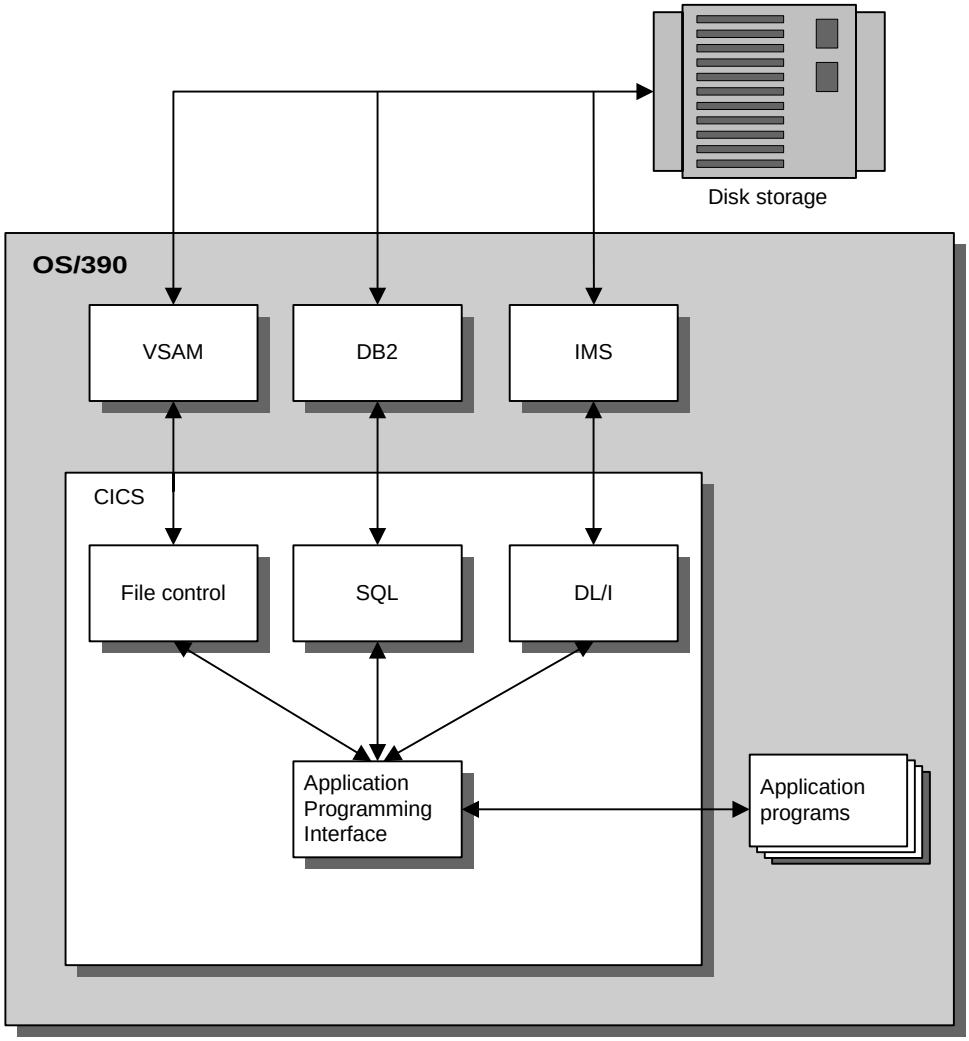
To keep track of which files are available to application programs, CICS maintains a table called the *File Control Table*, or *FCT*. In addition to the name and type of each file, the FCT lists the file control operations that are valid for each file. Specifically, it lists whether existing records can be read sequentially, read randomly, deleted, or modified, and whether new records can be added. Like the TCT, entries in the FCT are created and maintained by systems programmers.

As you will see in the next chapter, CICS file control simplifies the file processing code in your COBOL programs. Because the FCT keeps track of each file's characteristics, you don't use Select and FD statements to identify the files. And instead of COBOL I/O statements like the Open, Read, and Write statements, you issue CICS commands to the API.

In contrast to the file control service, CICS's *SQL* service is used to access data in a *DB2* database. SQL, which stands for *Structured Query Language*, is a standard language that's used to access the data within a database. If you have DB2 experience, chapter 17 will teach you how to use SQL statements within CICS programs to get the data that your programs need. (If you don't have DB2 experience, you can gain the background you need in our book, *DB2 for the COBOL Programmer, Part 1*.)

The third data management service in this figure is the *DL/I* service. It is used to access data in an *IMS* database. Because DL/I isn't used much any more, it isn't presented in this book. If you need to learn how to use it, though, we recommend a book that we published a number of years ago but that is still useful today, *IMS for the COBOL Programmer, Part 1*.

## Three ways an application program can access data from disk storage




## Description

- CICS provides three services to manage data: file control, SQL, and DL/I.
- *File control* passes data requests made by a program to VSAM. VSAM, in turn, manages the data stored in files on direct access devices.
- Each file accessed by CICS must be defined in the *File Control Table*, or *FCT*. The FCT specifies the file name, along with other information such as the file type and the operations that can be performed on the file.
- *SQL* passes data requests made by a program to IBM's relational database system, *DB2*.
- *DL/I* passes data requests made by a program to IBM's hierarchical database system, *IMS*.

Figure 1-9 Data management services

## CICS management services

Figure 1-10 summarizes the nine *CICS management services*. These services allow your application program to take full advantage of the features of CICS. Although you may never need some of these services, you should at least be familiar with them.

The *program control* service manages the programs that are executing within CICS. More precisely, it manages the flow of control from one program to the next. In chapter 2, you'll be introduced to some of the program control commands, and you'll master these commands in chapter 5.

*Temporary storage control* provides a convenient way to store information outside of the working storage area. In chapter 7, you'll learn how to use this service.

*Interval control*, *task control*, and *storage control* are services used to control how programs are executed in a CICS environment. With these services, you can control when programs start, how they allocate main storage, and what CICS resources they can have exclusive control over. You'll learn how to use these services in chapter 18.

*Dump control* can be used to help determine what the problem is when a program fails. With dump control, you can generate a report called a *transaction dump* that shows the contents of main storage used by the program. You'll be introduced to the command you use to create a transaction dump in chapter 2.

You can also use the report that's generated by *trace control* to help determine the problem with a program. Because you're not likely to use this report, though, this book won't show you how to create it. If you want more information on this service, you can refer to the CICS desk reference we publish.

You can also refer to the desk reference for information on the CICS commands for working with *transient data control* and *journal control*. Like trace control, you're not likely to use the commands for working with these services. However, you may use the features of journal control that provide for the automatic recording of file updates. You'll learn about these features in chapter 18.

## CICS services that help manage application programs

| Management service | Description |
|---|---|
| Program control | Manages the programs executing within a task. With this service, programs can call or transfer control to other programs. |
| Temporary storage control | Provides a simple way to store data outside the working storage area of a program. The data is stored in simple files called *temporary storage queues*. Because they're temporary, any data they contain is lost if CICS is shut down. |
| Interval control | Provides time-related services, including services for getting the current date and time and scheduling tasks for execution. |
| Storage control | Provides for allocating main storage space outside of the working storage area of an application program. |
| Task control | Can be used to control the execution of a task. Provides for temporarily suspending a task that is monopolizing CICS resources or gaining exclusive control of a CICS resource. |
| Dump control | Produces a *transaction dump* when a fatal error is encountered during the execution of an application program. The dump can then be used to help debug the program. |
| Trace control | Maintains a trace table that indicates the sequence of CICS operations performed. This table can be used to help debug a program. |
| Journal control | Creates a record or *journal* that can be used to restore files in the event of a task or system failure. |
| Transient data control | Provides a convenient way to use sequential files called *destinations* to store data. Records written to a destination are added to the end of the destination. Records are read sequentially from a destination and then deleted. |

## Notes

- Although you may never use some of the CICS management services, you should be familiar with the features they provide so you can use them if you ever need to.
- The CICS commands for working with all of these services except for trace control, journal control, and transient data control are presented in this book.

Figure 1-10 CICS management services

# Perspective

Now that you've completed this chapter, you should be familiar with the concepts and terms that you need to know as you develop CICS programs. The concepts will help you understand what is actually happening as you test your programs. The terms will help you converse with your colleagues.

That doesn't mean that you should remember exactly how every concept presented in this chapter works and what every term means. But you should at least be familiar with the concepts and terms so you can refer back to this chapter whenever necessary.

## Terms you should be familiar with

CICS (Customer Information Control System)
online transaction processing (OLTP)
region
graphical user interface (GUI)
3270 display station
terminal
3270 emulation
Local Area Network (LAN)
Wide Area Network (WAN)
front-end program
presentation logic
back-end program
business logic
External Call Interface
ECI call
MQSeries
message queuing
HTTP
task
multitasking
multithreading
reentrant program
transaction
transaction identifier
trans-id
Program Control Table (PCT)
Processing Program Table (PPT)
resource definition
Resource Definition Online (RDO)
CEDA
Application Programming Interface (API)
data communication services
terminal control
VTAM
SNA (Systems Network Architecture)
TCP/IP
basic mapping support (BMS)
map
Terminal Control Table (TCT)
terminal identifier
term-id
data management services
file control
File Control Table (FCT)
Structured Query Language (SQL)
DB2
DL/I
IMS
CICS management services
program control
temporary storage control
temporary storage queue
interval control
task control
storage control
dump control
transaction dump
trace control
transient data control
destination
journal control