CICS WEB SERIES

Mainframe Web Applications with CICS

Longchamps Informatique Inc.
95 rue d'Édimbourg
Laval, Qc, Canada
H7M 1N2

Tel: (450) 967-3160
Fax: (450) 967-0155
Cell: (514) 231-8163

E-mail: longlac@attglobal.net
www.longchamps-info.com

CICS/TS WEB Interface   November 26, 2003

MODULE 1

INTRODUCTION

Objectives

- To describe various ways to connect CICS to the WWW.
- To review the terminology used in that environment.
- To describe IBM's Websphere and list its components.
- To introduce HTTP Server for OS/390 and focus on the required directive statements.

Once upon a time...

➔ Traditional CICS applications consisted of :

- A transaction, attached to a terminal
- Business and application logic, that were part of the same program.
- A CICS program, which controlled the flow of information and program execution.
- A “closed” environment where CICS guaranteed data integrity by way of its command-level interface.

➔ A programmer could choose between a conversational and a pseudo-conversational environment. Either way, he was responsible for maintaining the state of the conversation by saving data in a **communication area** or in a **T**emporary **S**torage **Q**ueue.

➔ Networks were simple to maintain and considered safe since their communication lines were not shared with the public.

- BTAM and, later, VTAM were used as communication servers.
- CICS programmers didn't really have to know much about networks : just mainly about 3270 protocol and what BMS did with it.

Once upon a time...

From a programming standpoint, programs were built following a specific pattern.

```
PROCEDURE DIVISION.

   IF EIBCALEN=LENGTH OF DFHCOMMAREA
      PERFORM PROGRAM-BODY
   ELSE
      IF EIBCALEN=0
         PERFORM FIRST-TIME
      ELSE
         EXEC CICS RETURN END-EXEC
      END-IF
   END-IF

PROGRAM-BODY.
   EXEC CICS RECEIVE MAP(   ) MAPSET (   ) ...

   PERFORM GET-DATA-FROM-SCREEN

   PERFORM VALIDATE-DATA

   PERFORM PROCESS-DATA

   EXEC CICS SEND MAP (   ) MAPSET (   ) ...

   EXEC CICS RETURN
        TRANSID (   )
        COMMAREA (   )
   END-EXEC

GET-DATA-FROM-SCREEN.

   IF FIELD1L>0
      MOVE FIELD1I TO SAVE-FIELD1
  ELSE
      IF FIELD1F=DFHBMEOF
         MOVE SPACES TO SAVE-FIELD1
      END-IF
   END-IF
```

This was done for each field on the map. Once all data was gathered, the map I/O was cleared with low-values and map attributes were reset to their original values.

VALIDATE-DATA.

This routine validated the data that had been merged with data from a previous screen iteration.

PROCESS-DATA.

This was the routine acting on the data.

ERROR-ROUTINES.

These routines handled various exceptional conditions returned by failing CICS commands.

Once upon a time...

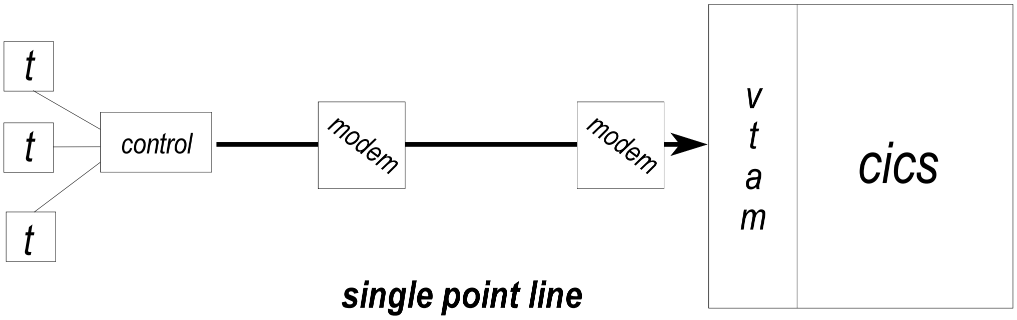
➔ If a change was made to a map, programs needed to be recompiled.

➔ In the late 70's, IBM introduced LU-6, an **A**dvanced **P**rogram-to-**P**rogram **C**ommunication protocol.

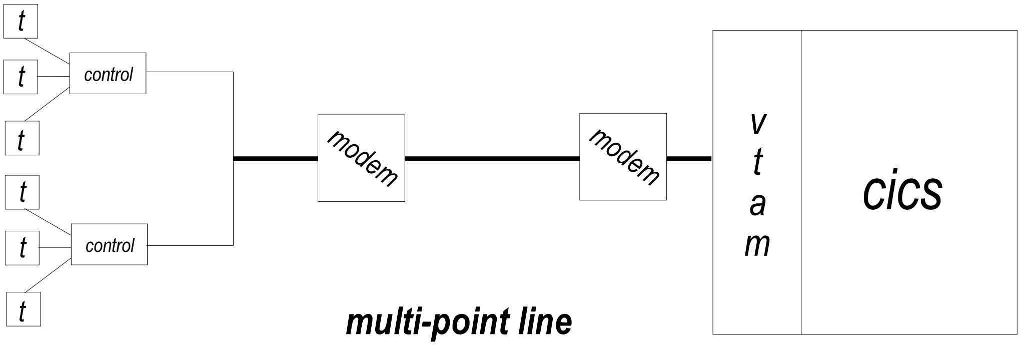
- It was now possible to exchange information between programs.
- Although maintaining the state of the conversation was a little more complicated, it was still the responsibility of the programmer.
- Basic program design remained unchanged.

Once upon a time...

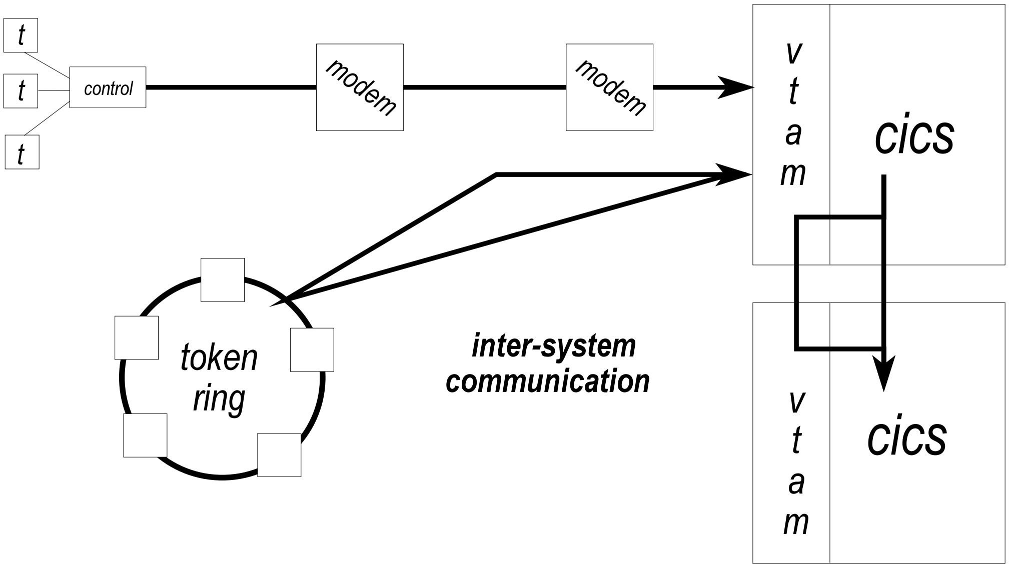
➔ Meanwhile, networks also evolved in terms of reliability and speed.



single point line



multi-point line



inter-system communication

Once upon a time...

In the late 80's, IBM concluded that the then-current design, which had carried CICS for two decades, would not work very well with new technology. The product was redesigned as CICS/ESA.

Application programs would continue to work and traditional design was supported. It was now easier for IBM to provide new functions.

➔ The **F**ront-**E**nd **P**rogramming **I**nterface was the first API implemented in CICS that allowed non-3270 devices to use and interact with CICS programs.

➔ Support for **T**ransmission **C**ontrol **P**rogram / **I**nternet **P**rotocol became available through VTAM.

➔ **D**istributed **P**rogram **L**ink provided “server” capabilities to CICS.

- By buying add-on products such as CICS/CLIENTS, it became possible to introduce client/server support in CICS.
- DPL led to EXCI in CICS/ESA 4.1. **EXCI** is a door to UNIX services. Programs capable of issuing **R**emote **P**rocedure **C**alls requests were now able to RPC to a CICS program.
- Using the TCP/IP socket interface made it possible to execute transactions that were not aimed at 3270. The original WEB interface used such capabilities.

➔ CICS/TS, introduced in 1996, changed the way data was accessed. CICS/TS 1.3 provides full support for

✓ 3270-based applications

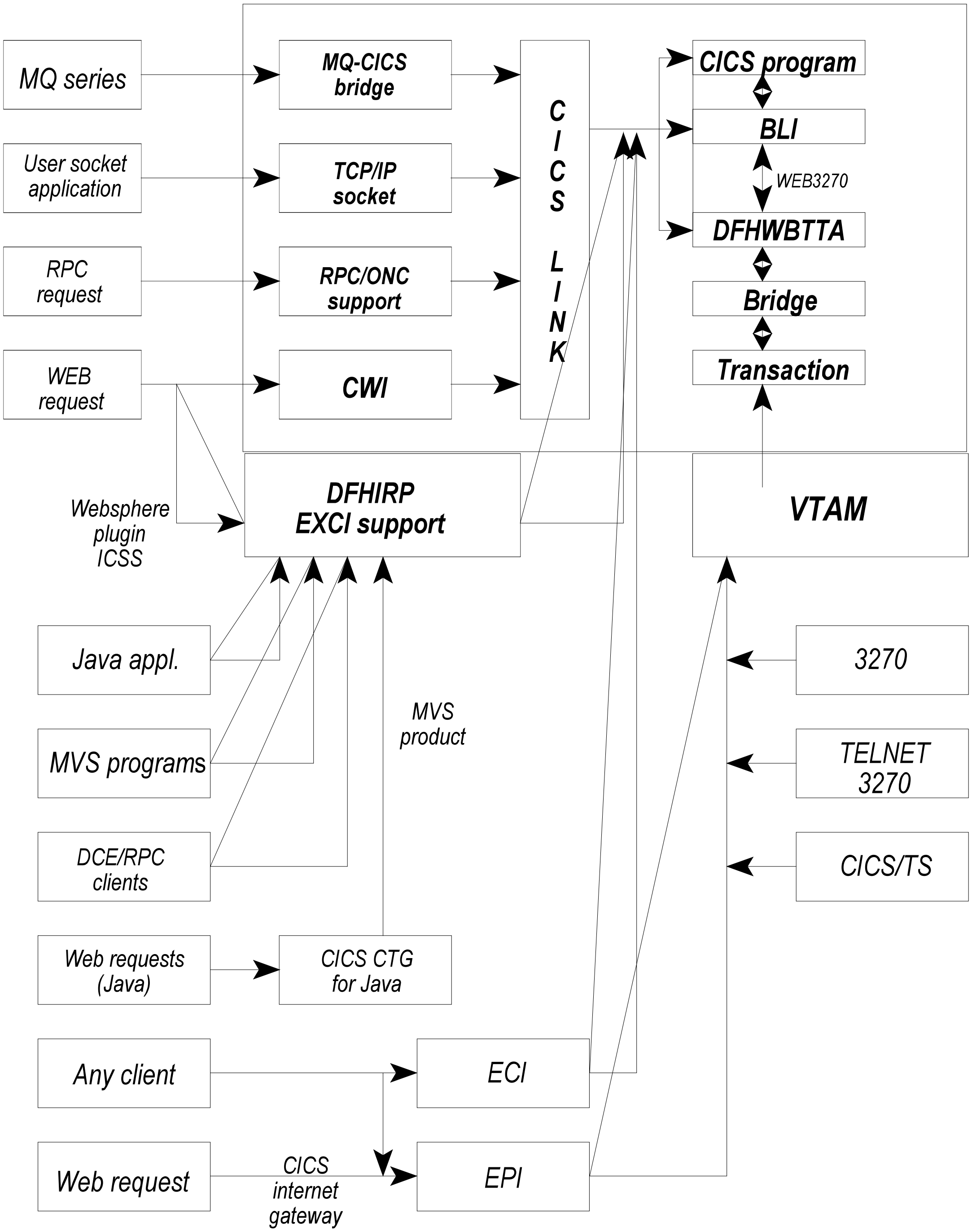
✓ Client/server-based applications

✓ Web-based applications

✓ In theory, with **CICS/BTS**, an application could receive input from multiple presentation interfaces, return output to multiple types of terminals and use the same business logic program for all cases.

➔ **Java**, with a very fast-growing user base, is now an official programming language in CICS.

Today's CICS Network



Today's CICS Network

➔ As illustrated on the previous page, CICS is now accessible from just about anything that runs on a computer. By linking to the **B**usiness **L**ogic **I**nterface, CICS can interact with

- ✓ MQ-Series
- ✓ Its own TCP/IP socket application, for example, E-MAIL
- ✓ TCP/IP Remote Procedure Calls
- ✓ Web URLs pointing at CICS programs

● EXCI can be used to support

- ✓ Requests from OS/390 web servers
- ✓ MVS programs wanting access to CICS data
- ✓ DCE clients
- ✓ Java clients

● Using CICS/CLIENTS ECI interface, any client program can gain access to CICS/BLI.

● CICS for INTERNET GATEWAY, combined with the **E**xternal **P**resentation **I**nterface, can provide access to CICS server programs.

NOTE
EPI access does not include attachment compatibility with CICS/BLI. EPI uses virtual terminals to transfer data while CICS/BLI expects a commarea.

Web-enabled Applications

➔ Web-enabled applications generally consist of three elements : presentation, business logic and data access logic.

- Business and data access logics are usually part of an application.
- Presentation logic could be
  - ✓ HTML pages and associated scripting
  - ✓ Java servlets driving dynamic HTML
  - ✓ Java applets communicating directly with the server
- These above-mentioned facilities always involve
  - ✓ a web browser on the client's machine,
  - ✓ a web server.

➔ CICS server programs can be accessed by

- ✓ enabling web browsers to invoke CICS application programs;
- ✓ using standard HTTP and IIOP protocols;
- ✓ using gateways and/or native access.

➔ Several design considerations must be looked at before making a final decision in choosing a CICS web support.

- **Is this a brand new application ? Is it strictly web-based ?**
  - ✓ A strictly web-based application means that business logic does not need to be separate from presentation logic.
  - ✓ It also means writing web-aware applications. Personnel should be familiar with this kind of design.
  - ✓ It will be easier to debug.
- **Is it going to be accessed from other sources ?**
  - ✓ In this case, business logic will have to be separate from presentation logic.
  - ✓ A mechanism, such as CICS/BLI, will be needed.
  - ✓ An integration logic has to be designed.
  - ✓ It is very likely to be harder to test and debug.

Web-enabled Applications

- **Will it be using existing code ?**

  ✓ Existing code usually includes handling a 3270 screen. Customization of that screen may be required.
  ✓ Detecting and flagging errors may have to be changed.
  ✓ Using existing code may speed up the web-enablement of the application.
  ✓ Implementation should be easier since the status of existing applications is known. The only new utility is the web browser.
  ✓ Existing maps must be converted to HTML templates.

- **Is an MVS web server available ?**

  ✓ An MVS-based web browser means flexibility because direct connections, HTTP and EXCI interfaces are supported.

- **Can a new programming language such as Java be handled ?**

  ✓ Java is relatively new to the mainframe market.
  ✓ IBM made it clear that it will be a changing environment, e.g. CICS/TS 2.1 announcement.
  ✓ There are very few debugging tools around.
  ✓ Talent is not easy to get and very expensive. Even if Java talent is available, it still means training on mainframes. Training means “learning curve”. How quickly does a web presence need to be provided ?

- **If access to existing applications is required, are they easy to change ?**

  ✓ How easy will it be to modify that code ? Is the code available ?
  ✓ Which compiler is used ? VS/COBOL is no longer supported and these programs will not work in the next generation of O/S.
  ✓ Which database manager is used ?

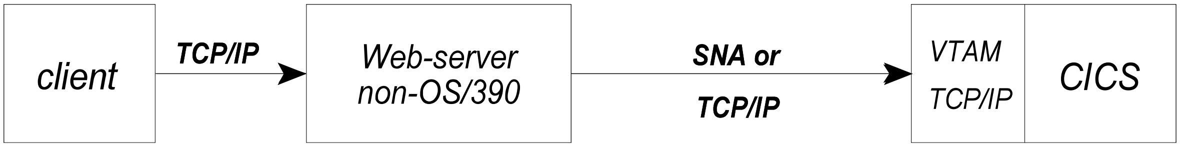
- **Which new product is being considered ?**

  ✓ CICS Transaction Gateway for Java
  ✓ Host-on-demand
  ✓ CICS native IIOP interface (Java)
  ✓ **C**ICS **W**eb **I**nterface (native HTTP)
  ✓ WEB3270 (CWI with 3270 bridge)
  ✓ NETCICS
  ✓ A customized TCP/IP socket interface
  ✓ Native Java
  ✓ CICS/WEB Interface and Business Transaction Services

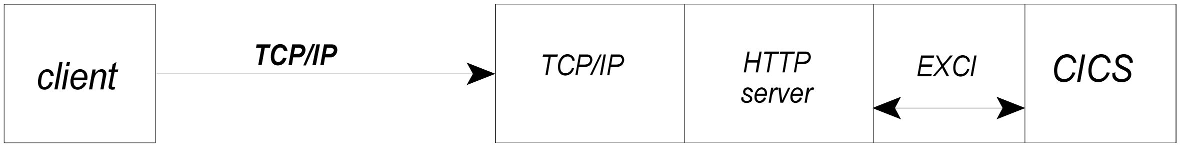
Connection Architecture

➔ There are three types of connection :

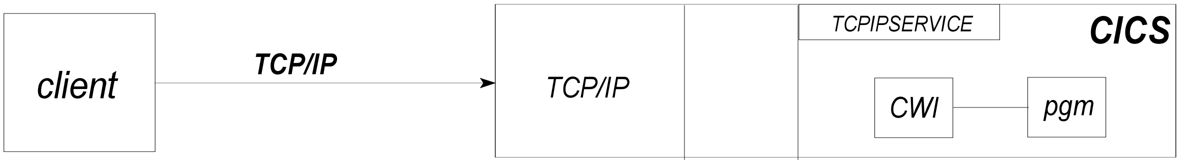
- Non-OS/390 servers that connect to CICS using a network.



- OS/390 servers that connect to CICS using EXCI.



- Requests sent directly to CICS in HTTP format.



Selection Criteria

➔ Corporate Internet strategy

- Solutions should be in line with one's corporation Internet strategy.
- Corporations should be committed to their strategy and have a long-term plan in terms of software, hardware and training.

➔ Budget

- It may be difficult to fully assess the costs and benefits of supporting e-business.
- If a site is well received, there may be additional costs related to acquiring more harware.
- If errors were made, there may be a cost associated with the loss of business.

➔ Capacity and scalability

- Historically, most Internet failures were due to capacity problems.
- Response problems will send customers away.

➔ Availability

- The intent is a 24/7 type of operation.
- The server application should also be 24/7.

➔ Security

- Solutions should provide iron-clad security.
- Authentication of users may be necessary.
- Data may require encryption.

Selection Criteria

- ➔ Expertise
  - Using consultants for this kind of project might resolve it faster but, after completion, support will be left with the site's personnel.
  - A lot of tools are available. Using new tools requires training and, therefore, time.
  - Current technology has evolved to a point where it may be simpler to use products currently in use in one's shop. For example, when strong CICS programmers are available, it may be easier to learn about CWI instead of retraining everyone in Java.
- ➔ Audiences
  - An Internet site aimed at the general public may require a more sophisticated presentation.
  - Corporate users will probably benefit more from a 3270-type approach.
- ➔ Administration & Support
  - The support infrastructure must be in place on the first day of production.
  - Avoid “Under Construction” web pages. Generally speaking, anyone seeing this sign is not likely to come back.
  - How will the code running on the client be supported, if any ? How will the code be distributed ? How will problems be answered ? These are all issues a solution must consider.

Selection Criteria

➔ The above points are key to success when implementing an e-business. A good solution must be chosen by taking into account as many criteria as possible. E-business cannot be implemented based on

- ✓ Politics
- ✓ Magazine advertisement
- ✓ A solution tailored for another enterprise

- If CICS is to be part of the scheme, remember that the Internet environment is different from traditional transaction processing.

- ✓ Transactions are not connected to a terminal.
- ✓ Protocols are different.
- ✓ Web browsers react differently than 3270.

- As a result, CICS designs are more complicated. Aside from supporting the Internet, a number of other enhancements were made to CICS and they may be required in a solution :

- ✓ Parallel sysplex
- ✓ VSAM/RLS
- ✓ Coupling facility services for TS and Global ENQ
- ✓ Workload distribution

Selection Criteria

➔ A number of interface products are available. These products can help in various ways.

- Some really are a “turn key” solution, and allow reusing existing applications.

  ✓ CICS Transaction Gateway
  ✓ The WEB/3270 part of the CICS/WEB interface
  ✓ NETCICS, using a FEPI interface
  ✓ Host-On-Demand

- Other products give more flexibility when designing new e-business applications but in most cases, existing applications wont be reusable.

  ✓ CICS Transaction Gateway
  ✓ CICS/TS WEB interface
  ✓ CICS Socket interface (part of TCP/IP)
  ✓ CICS Universal Client
  ✓ CICS/MQ interface
  ✓ Websphere
  ✓ CICS/TS Java and native IIOP support.

- A bit of information on some of these products will be given in the next pages.

➔ In conclusion, when making a final decision, keep in mind the following :

- Cost
- Expertise
- Time at disposal
- Audience
- Data location

CICS Transaction Gateway

➔ This Java-based web server product is used to access CICS applications. The facilities include

- Java classes and Java beans, for writing servlets that allow bundling requests that require CICS services.

- Java classes and Java beans for writing Java applets that can be downloaded to a browser.

- For non-OS/390 web servers only, terminal servlets are available that provide access to 3270-based applications. This facility uses Java to emulate a dumb terminal. It also allows customization of the screen sent to the browser.

- CORBA client is supported when communicating with Websphere.

- CICS Transaction Gateway has five components :

  - ✓ The Java Gateway application, communicates with CICS using CICS Universal Client with non-OS/390 servers. OS/390 servers use EXCI.
  - ✓ CICS Universal Client, provides **E**xternal **C**all **I**nterface or **E**xternal **P**resentation **I**nterface (ECI or EPI).
  - ✓ A CICS/Java class library API, for communication between the Gateway and user applications.
  - ✓ Java EPI beans, for creating front-ends to existing CICS 3270 applications.
  - ✓ Terminal servlets, for emulating 3270 terminals.

- Connections to CICS are made via TCP/IP or SNA except for Sun Solaris which only supports TCP/IP connections.

- When emulating 3270 terminals, no programming is required other than applying “cosmetics”. For any other type of implementation, Java code will be required.

CICS Transaction Gateway

➔ This product has some limitations :

- CICS Universal Client really is the implementation of **D**istributed **P**rogram **L**ink, performed using CICS commareas that have a maximum of 32K. If more than 32K of data needs to be exchanged, multiple invocations of the server program will be required.

- EPI does not support EXEC CICS RETURN IMMEDIATE, nor does it support BMS paging commands.

NOTE
Terminal servlets also use EPI to communicate with CICS.

CICS WEB Interface

➔ A full set of services supporting direct access to CICS over a TCP/IP connection. CICS will monitor specific TCP/IP ports for input by using a new system task, **CSOL**, the CICS listener transaction. Facilities include :

- A new listener transaction that will monitor TCP/IP ports and start CICS work based on the contents of a new resource definition called **TCPIPSERVICE**.
- An analyser program, to pass incoming URLs and determine which CICS program or service is required.
- A WEB/3270 environment, to provide an emulation interface for 3270-based applications.
- Data converters, to translate ASCII to EBCDIC and vice versa.
- ENCODE/DECODE functions that use a commarea interface, allowing the use of CICS/BTS-type facilities.
- A **B**usiness **L**ogic **I**nterface that allows multiple WEB interfaces to share a common application.

➔ CICS/TS 1.3 is shipped with the following components :

- Three new application programming interfaces called TCP/IP, DOCUMENT and WEB.
- New domains, to provide access to web resources :

✓	WB	CICS/WEB interface
✓	DH	Document domain
✓	SO	Socket
✓	S8	SSL interface

- A sample analyser and Web-3270 programs that could be used on an “as is “ basis.
- New traces
- New dumps
- A new Java API

CICS WEB Interface

There are no specific hardware/software requirements other than the CICS/TS 1.3 installation requirements.

CICS/TS now supports HTTP (version 1 only), thus enabling the use of traditional programming languages to develop web-based applications.

The CICS/WEB interface does not require a web server to be in place. However, any CICS/WEB application should be realized in conjunction with a web server in order to provide web facilities such as FTP, e-mail, news, Gopher Deamon, etc...

➔ This product's limitations :

- WEB/3270 uses a 3270-bridge type interface and is therefore governed by the same rules and restrictions in place for any 3270-bridge application.

✓ EXEC CICS SIGNON, SIGNOFF commands cannot be used.

✓ The CLSDST (PASS) command cannot be used.

✓ BMS commands with ACCUM or PAGING will not work.

✓ WEB/3270 also requires that BMS maps be converted to HTML templates, entailing some customization work.

NETCICS

Using CICS/FEPI, it allows the web-enablement of current CICS applications without the need to migrate these applications to a newer version of CICS. It also provides facilities for implementing a new user interface to an existing application.

- The following facilities are included :

✓ A CICS server region that sits between the browsers and the applications.

✓ FEPI pass-through transactions.

✓ Customization routines written in REXX/CICS.

- Because the server region uses the 3270-bridge environment, CICS/TS 1.2 is the minimum requirement. The user CICS region can be at any CICS level.

- Because of FEPI, there must be a VTAM connection between NETCICS and the target CICS region.

- There are no specific program requirements other than REXX/CICS for customizing screens.

➔ The limitations :

- Because of FEPI, restrictions that apply to WEB/3270 do not apply here.

- VTAM CLSDST PASS cannot be used.

- Some 3270 features are not supported :

✓ Inbound cursor location
✓ Numeric-only
✓ Must-fill fields audit
✓ Color
✓ Alternate screen size
✓ Structured fields

- This IBM offering will make the transition to the web easy.

- Indispensable for web-enabling CICS applications that cannot be migrated to CICS/TS 1.3 or higher.

CICS/IIOP Interface

__I__nternet __I__nter __O__RB __P__rotocol is a facility supported since CICS/TS 1.3 ; it provides inbound access to CICS applications.

It defines message formats and protocols used for communication between object-oriented programs executing on different processors supporting __C__ommon __O__bject __R__equest __B__roker __A__rchitecture (CORBA).

➔ Facilities include :

- A CICS listener transaction which monitors a TCP/IP port for incoming IIOP requests.
- CICS will determine which program to run based on two resource definitions, **TCPIPSERVICE** and **REQUESTMODEL**.
- A set of CICS programs that will

✓ provide a userid;
✓ instantiate the target object;
✓ provide input parameters;
✓ invoke the user Java application program;
✓ format a reply and send it back to the requester.

➔ All facilities are part of the base product in CICS/TS 1.3

➔ A Java compiler must be available. Both Java objects and CORBA-compliant clients must be available.

➔ Java programs can invoke non-Java CICS programs, which could then be used to provide access to existing code.

CICS/IIOP Interface

➔ Limitations :

- Inbound request only are supported.
- The following CORBA specifications are not supported :
  - ✓ Name Server support
  - ✓ Externalization service
  - ✓ Persistence service
  - ✓ Interface repository framework service
- Transaction and security contexts are managed by CICS, not CORBA.
- Application data is managed by CICS ; outbound requests are not supported.
- Location service is not used. All object references refer to a specific server or, if WLM is in use, to a server group.

Websphere

Websphere is a software product that implements a three-tier architecture.

- The client platform consists of a web browser, applets and an application.
- The middle tier is composed of a web server, a servlet redirector and administration servers.
- The third tier includes interfaces to mainframe products such as MQ and CICS.

➔ Websphere is a suite of products including :

✓ Servlet API
✓ JSP
✓ HTTP session and user profile support
✓ Administration database
✓ Connection pooling
✓ DB/2 data administration tools
✓ Remote servlet support
✓ Integrated security
✓ CORBA support
✓ Support for C++ and Java CORBA Business objects

➔ The Performance Pack provides a set of highly scalable functions like

✓ Caching
✓ Non-disruption servers
✓ Data mirroring and replication
✓ Dynamic load balancing

➔ Websphere Studio offers

✓ a workbench, for creating, editing and copying files;
✓ a wizard, which shortens the amount of time required to become proficient with the product;
✓ a page designer, to help create HTML documents;
✓ an applet designer, to assist when coding in Java.

➔ A test environment and a Java beans development environment are also included.

Websphere

Websphere is a product that can most definitely help to come up with a “unique” implementation scenario.It works on most computing platforms, can provide a single interface to users and it supports connections to CICS and DB/2.

The HTTP Server

➔ Web browsers format requests by building a **U**niform **R**esource **L**ocator. The purpose of the URL is to uniquely identify the request.

```
HTTP://WWW.LONGCHAMPS-INFO.COM/SERVICE
```

➔ A URL has to contain the following information :

✓ Basic URL format
✓ Analyser transaction
✓ Name of the program or transaction code
✓ Any other data required by the CICS user program.

➔ The format of a basic URL is

protocol://hostname<:port>/identifiers

- Protocol can be HTTP, HTTPS, FTP, NNTP or EMAIL.
- <:port> specifies the port on which the server is listening.
- Identifiers identify the request. This is the part used by CICS.

Websphere

The HTTP Server

➔ CICS uses a request/response protocol that works as follows :

- The client sends a request.
- The server sends a response.
- The stateless environment works contrary to APPC (which is connection-oriented) and operates over a TCP/IP link.

➔ Requests are made up of <u>method</u>, <u>headers</u> and <u>body</u>. Sometimes, a footer will be added. Common methods are :

✓ GET	--> Ask for data.
✓ POST	--> Transmit data and get response. **This is where CICS comes in.**
✓ PUT	--> Store information. Mostly used with FTP.
✓ DELETE	--> Delete information.

➔ A request is always followed by a response that has the following format :

✓ A status line
✓ Headers
✓ Body

➔ The basic function of a web server is to map a request to a file or service on the CICS server. IBM's HTTP server is powered by “Apache”, which is the server of choice in the business today. In MVS, multiple servers can be started but each server can service one port only. To configure a server :

- Start Open Edition in TSO. On the command line, enter

```
====> TSO OMVS
```

- In Unix services, edit the configuration file HTTPD.CONF

```
OEDIT HTTPD.CONF
```

MODULE 2

TCP/IP

Objectives

- To review TCP/IP concepts and protocols.
- To discuss TCP/IP implementation in CICS/TS.
- To discuss TCP/IP CICS application programming interfaces.

CICS Implementation

Support for TCP/IP was formally introduced in CICS with CICS/TS 1.3.

➔ CICS systems have supported TCP/IP connections for a long time, using the TCP/IP socket interface available with TCP/IP. A socket program running in the CICS address space analyzed incoming requests. After analysis was completed, control was passed to the user application.

➔ CICS now supports two specific TCP/IP services, **HTTP** and **IIOP**, with or without SSL support.

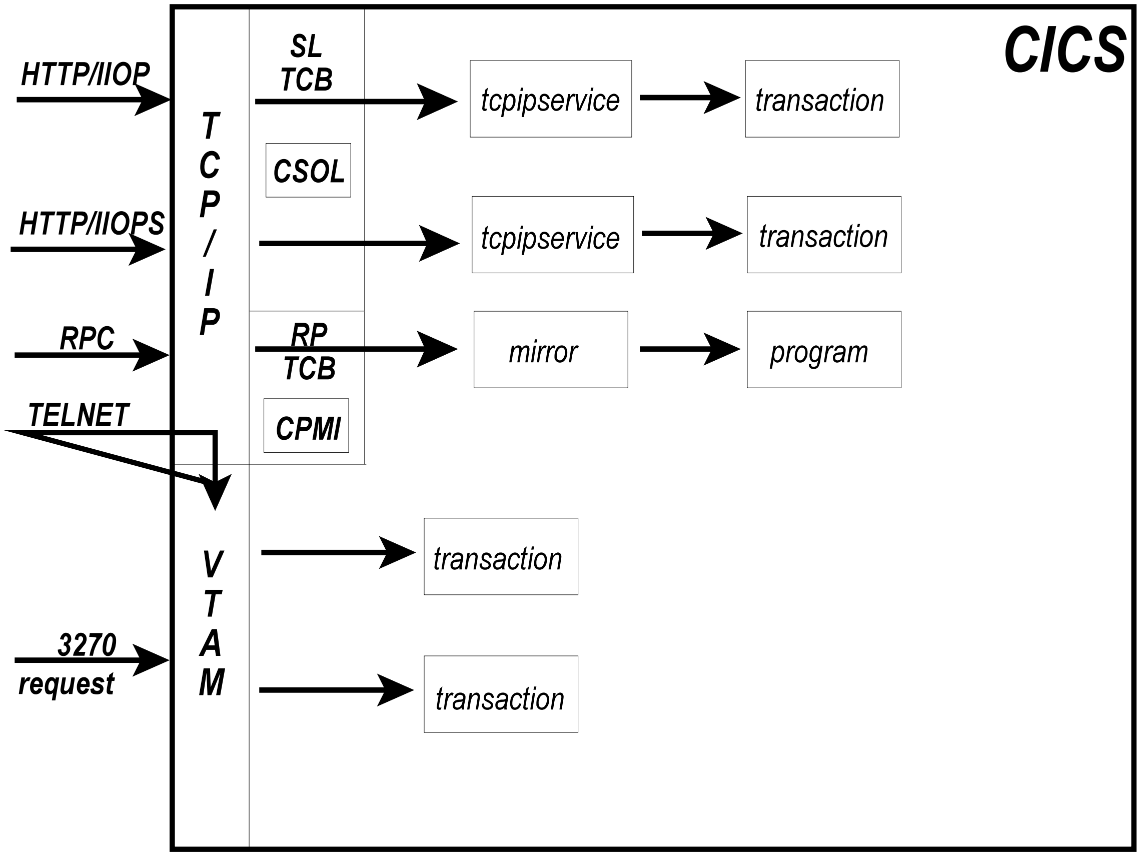
NOTE
TELNET-3270, which provides 3270 emulation across TCP/IP networks, is not supported by CICS, but by VTAM.
Therefore, a terminal connected to CICS via TCP/IP is seen by VTAM first. As a result, CICS will see this activity as if it was coming from SNA network.

➔ Port mapping (service identification) is accomplished by providing a new resource definition called **TCPIPSERVICE**.

➔ The TCP/IP configuration file should be used to reserve ports for specific CICS address spaces, as will be discussed later on...

CICS Implementation

The following diagram illustrates the various ways of connecting to CICS.



- Note : Socket programs are invoked by TCP/IP, which then invokes **CSKL**, the socket listener, then the user application.

CICS Implementation

TCP/IP configuration

➔ As many ports as needed should be reserved for CICS web support.

- Any port number from 256 to 65535 can be used. For MVS, port numbers below 1024 must be defined to UNIX System Services.

- To reserve a port for CICS, specify the **PORT** option and the CICS jobname in the TCPIP.PROFILE.TCPIP dataset.

➔ Use the **SOMAXCONN** parameter to set the maximum length of any queue of requests for a TCP/IP port. This value must be coordinated with the value chosen for the **BACKLOG** parameter of the TCIPIPSERVICE resource definition.

➔ For full function CICS/WEB interface, CICS needs to access a name server during its operation. Programs **DFH$WBSN** and **DFHWBENV** use this service.

- The default name server can be set by providing its address in a file allocated to the **SYSTCPD DD** statement in the CICS JCL. This will set the RESOLVER CONFIG environment variable to the MVS dataset. Example :

NSINTERADDR n.n.n.n where n.n.n.n is the IP address of the name server.

CICS Implementation

To activate TCP/IP support in CICS/TS :

DFHSIT TCPIP = YES

- Support is provided via two MVS TCBs :

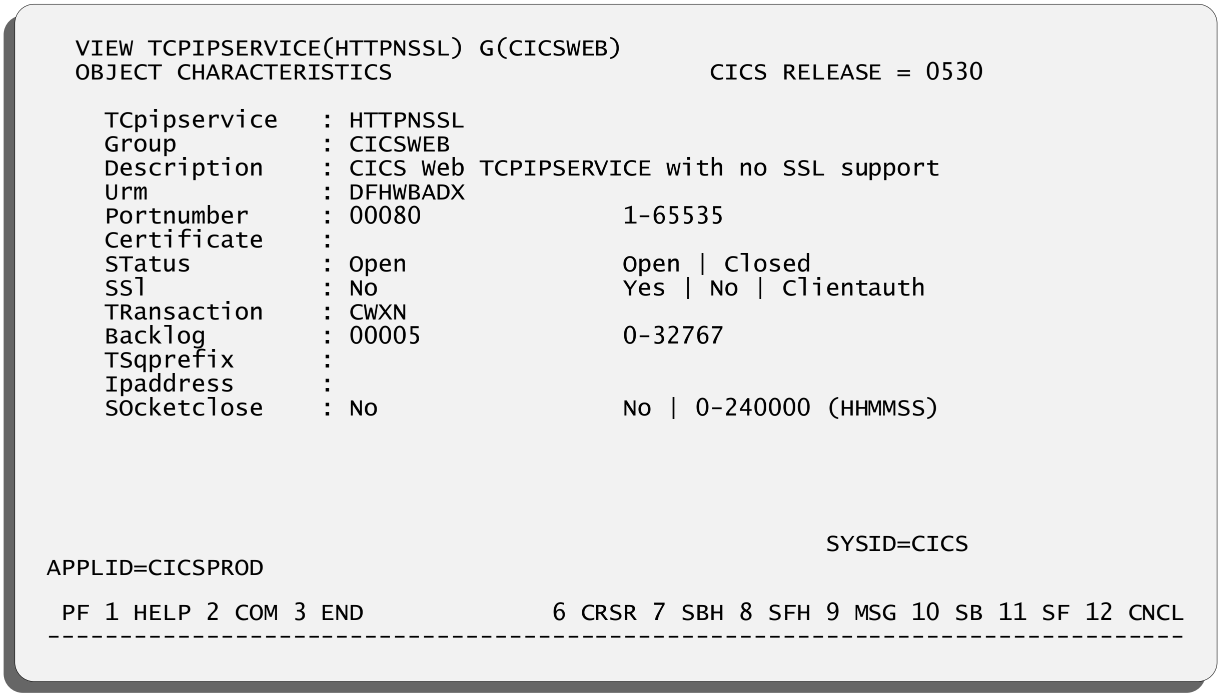
✓	SL-TCB, the socket listener	Use it to listen for incoming traffic.
✓	SO-TCB, the socket interface	Use it to process all other socket activities like SEND and RECEIVE.

- In addition, there must be a TCPIPSERVICE resource definition for each port CSOL is going to listen to.

TCPIPSERVICE

- The TCPIPSERVICE resource definition is used to configure the port CSOL will be listening to.

Example



```
  VIEW TCPIPSERVICE(HTTPNSSL) G(CICSWEB)
  OBJECT CHARACTERISTICS                       CICS RELEASE = 0530

    TCpipservice   : HTTPNSSL
    Group          : CICSWEB
    Description    : CICS Web TCPIPSERVICE with no SSL support
    Urm            : DFHWBADX
    Portnumber     : 00080             1-65535
    Certificate    :
    STatus         : Open              Open | Closed
    SSl            : No                Yes | No | Clientauth
    TRansaction    : CWXN
    Backlog        : 00005             0-32767
    TSqprefix      :
    Ipaddress      :
    SOcketclose    : No                No | 0-240000 (HHMMSS)



                                                          SYSID=CICS
APPLID=CICSPROD

 PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
------------------------------------------------------------------------------
```

CICS Implementation

TCPIPSERVICE

- **TCPIPSERVICE** specifies a 8-character name for this service.
- **URM** is the name of the analyzer user-replaceable module.
  - ✓ For HTTP requests, this module scans the URL and determines which program to execute. Module **DFHWBADX** is supplied as a default URM.
  - ✓ For IIOP, module **DFHXOPUS** allows the specification of a userid.
- **Portnumber** identifies which port Socket Domain is to register. It can be any number from 1 to 65535.
- **Certificate** identifies which digital certificate is to be used as the server certificate. It only applies when SSL is **not** set to NO.
- **Status**
  - ✓ Set to **OPEN** and when the definition is installed, the service will be registered with the Socket Listener and socket monitoring will start.
  - ✓ Specify **CLOSE** to prevent SL-TCB from monitoring TCPIPSERVICE when the definition is installed.
- **SSL** specifies whether **S**ecure **S**ocket **L**ayer support is to be activated and the type of support required.
- **Transaction**
  - ✓ Indicates the name of the transaction to be attached by the socket listener. **CWXN** should be used for HTTP requests.
  - ✓ A user-defined alias transaction could also be specified, but the first program executing **must** be **DFHWBXN**.
  - ✓ For IIOP, the transaction must be **CIOR** or an alias and the first program to execute must be **DFHIIOPA**.

CICS Implementation

TCPIPSERVICE

- **Backlog** represents the maximum number of requests queued by TCP/IP and waiting for processing by CICS. Once this number is reached, further incoming connection requests will be rejected. It must match **SOMAXCONN** in the TCP/IP configuration file.

- **TSQPrefix**

  - ✓ Specifies a 6-byte character string used to generate the names of temporary storage queues used by the interface.
  - ✓ A **TSMODEL** must exists.
  - ✓ **DFHWEB** will be used as default if no TSQ prefix was defined.

- **IPAddress**

  - ✓ Specifies the IP address this service is to communicate with. Use it when multiple servers access a given CICS system.
  - ✓ If no IP address is supplied, CICS will accept requests from all addresses defined to the system.

- **Socketclose=** *0* | *timevalue* | *no* determines the termination of the CWNX transaction.

  - ✓ **When set to *0***, the second web RECEIVE performed by CWXN is immediately posted with a closed socket. CWXN will then terminate and new incoming data will cause CWXN to be reattached.
  - ✓ **When set to *timevalue***, the socket remains open until *timevalue* is reached; during that time, incoming data is handled by the existing CWXN transaction. When *timevalue* is reached,
    - ✧ the socket is closed;
    - ✧ CWXN terminates;
    - ✧ subsequent incoming data will cause CWXN to be reattached.
  - ✓ **When set to *no***, CWXN never terminates.
    - ✧ All work is handled by the existing CWXN.
    - ✧ After processing a request, it issues another RECEIVE SOCKET and waits for more work from the RECEIVE command.
  - ✓ **You are recommended to set this parameter to *no* as it is the operand that allows achieving a given workload using the least amount of CPU time.**

MODULE 3

WEB3270

Objectives

- To describe the 3270 bridge environment.
- To analyze the CICS/WEB implementation of the 3270 bridge facility.
- To review the requirements needed to activate 3270 support in a web environment.
- To discuss implementation and customization strategies.

Overview

Implementing WEB3270 is a likely choice when using existing CICS applications that accept data from 3270 terminals. It can be an old application that cannot be modified or customized or it could be a brand new application that accepts data from both 3270 and the Web.

WEB3270 also is a logical choice when a site's expertise is mostly in the 3270 area; it is then faster to develop an application for 3270 and implement it on the Web.

Finally, WEB3270 is a good choice when one's corporate strategy plans to eliminate all 3270 hardware and replace it with web browsers.

➔ CICS provides the following components :

- A bridge monitor program, **DFHWBTTA**, that sends and receives requests to/from a HTTP port and starts a user transaction under the 3270-bridge facility.
- A bridge exit program, **DFHWBLT**, which converts data as appropriate.

➔ In most cases, it will require little effort to implement; there are a few things not supported by the bridge that can be worked around. Also, some customization of the maps could be needed, mostly for cosmetic reasons.

➔ It might be desirable to review the programming techniques used in the old applications. Detecting errors has not changed, but the way of telling the user about them might have.

- For example, highlighting a field to show an error works well on 3270 but not on the Web.
- When colors are used, a white attribute shows well on the black background of a 3270 terminal but will not show at all on the defaulted-white background of a web page.
- Attention identifiers are 3270 keys and must be simulated on the Web, as explained later in this module.
- By default, CICS generates a button for each of the 24 PF keys. It is doubtful that a program makes use or tests them all. Plus, the cluttered display does not look very sharp on a page.

3270 Bridge

This facility allows the execution of 3270-based transactions on non-3270 hardware.

➔ CICS supports the following transport environments. A bridge exit is supplied for all.

✓ Web
✓ MQ series
✓ CICS/BTS
✓ TS
✓ TD

➔ Typically, when the bridge is used, a transaction is started.

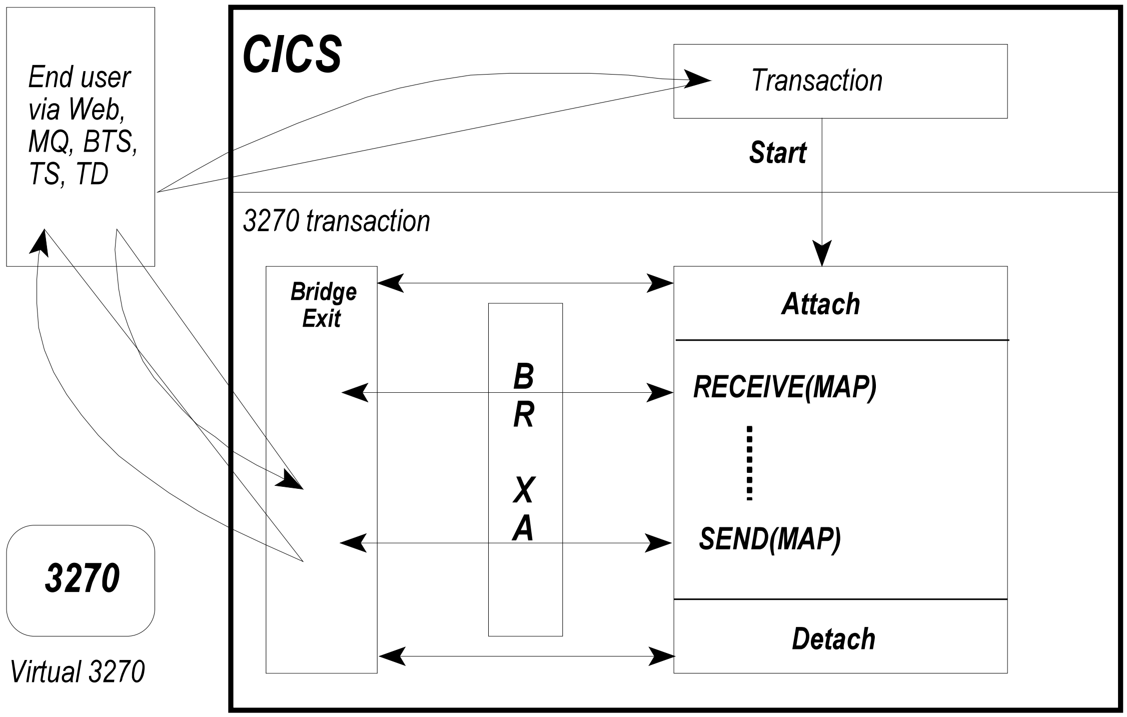
- The transaction definition indicates that this transaction is to execute under the bridge.
- A monitor program analyzes the input stream and starts the transaction using a CICS command.

```
EXEC CICS START
            TRANSID( )
            BREXIT( )
            BRDATA( )
```

- The first call to the exit sets up a virtual terminal seen only by that task.
- SEND and RECEIVE commands are interrupted by the bridge exit.
- At transaction detach time, the bridge exit is activated to commit the data.

3270 Bridge

Graphical Representation



3270 Bridge

Bridge Components

➔ The 3270-bridge mechanism is a CICS facility that allows the support of 3270 transactions in a non-VTAM environment. It includes

- The **user** transaction
- The **client** application; in our case, a web-based application
- The **transport** mechanism
- **Messages**, to provide the data necessary to run the 3270 application
- The bridge **monitor**, a long-running task associated with the transport mechanism, which monitors incoming messages and starts the transaction under the bridge when requested.
- The bridge **environment**
- The bridge **exit**

✓ Responsible for formatting data

✓ A separate program can also be used to format the data. The formatter program or the exit will have access to 3270 data in an ADS descriptor, an area that describe the map and its fields.

✓ The exit is called at the following points :

- ✧ Transaction initialization
- ✧ Transaction bind
- ✧ Transaction termination
- ✧ Transaction abnormal termination
- ✧ EXEC CICS SEND and RECEIVE
- ✧ SYNCPOINT

✓ The formatter will be called when the following functions are used :

- ✧ SEND and RECEIVE
- ✧ CONVERSE
- ✧ FREE
- ✧ ISSUE DISCONNECT and ERASAEUP
- ✧ RETRIEVE

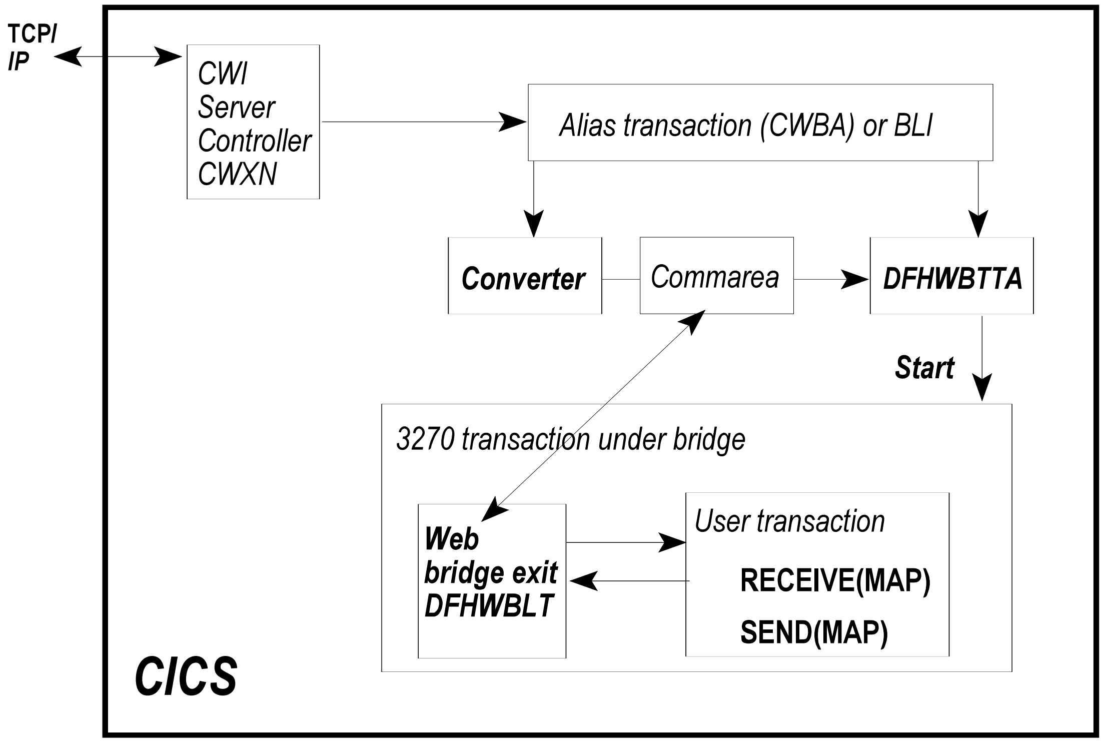
3270 Bridge

Bridge Components

- The **formatter**, as described above
- The **bridge exit area**, a commarea passed to the exit, contains the following sections :
  - ✓ A header containing version information
  - ✓ A transaction area containing information about the user
  - ✓ A command area providing details about the command being intercepted
  - ✓ A user area, to store data between invocations of the exit
  - ✓ An ADS-descriptor detailing the BMS map being intercepted
- The **bridge facility**, a virtual terminal that exists only for the duration of a task.
- The **FACILITYLIKE** definition to identify the type of terminal being emulated.
- CICS-supplied **programs** that can be used as bridge exit routines or as templates to ease the development of a bridge exit.
- The CICS/WEB interface includes **DFHWBTTA**, a bridge monitor program, and **DFHWBLT**, the bridge exit.

3270 Bridge

CICS/WEB Implementation



3270 Bridge

CICS/WEB Implementation

➔ The analyzer program prepares a request in the commarea and passes it to **DFHWBTTA**, the monitor program. DFHWBTTA accepts two types of requests :

- An initial request to start a transaction; data can be passed to the transaction using a "+" sign instead of a blank to separate the transaction id from the data.

- A continuation of a pseudo-conversation. The initial path can be any format as long as the transid follows the last "/" of the URL. DFHWBTTA has stored state information on the previous HTML page as hidden fields.

➔ To activate the process, the URL specified in the action field of the form or the URL used as a target in a reference must be in one of the following formats :

```
http://IPaddress or hostname :
                [port]/converter/alias/program/transid
                        /converter/alias/program/trans?token
                        /converter/alias/program/keyword/transid
                        /converter/alias/program/keyword/transid?token
```

➔ **DFHWBTTA** ignores the values for the converter program, the alias transaction id and the program name.

- ***keyword***, if present, should say "UNFORMAT" to indicate that input is coming from an unformatted screen.

- ***transid*** is the transaction to be started under the bridge. It is ignored when part of a continuation request.

- ***token*** will also be ignored by the monitor program.

For example, to get transaction "LM00" to start, the URL could be :

```
http://www.longchamps-info.com/CICS/cwba/dfhwbtta/lm00
```

To issue a CEMT command on port 89 :

```
http://38.275.598:89/CICS/cwba/dfhwbtta/cemt+inq+tas
```

3270 Bridge

CICS/WEB Implementation

➔ A continuation request may contain data; it will be URL-encoded data in the form of ***variable=value***. Elements should be separated by ampersands.

➔ To maintain status information about the execution of a task, BMS generates special fields when converting maps to templates.

- Data is saved in a TSQ whose name is found in field DFL_STATE_TOKEN.
- **EIBCPOSN** is kept in field DFH_CURSOR.
- Buttons are associated with PFkey settings as discussed earlier.
- The name of the next transaction to execute is found in field **DFH_NEXTTRANSID**.*n* where *n* is a number representing the variable that will be searched.

MODULE 4

OS/390 WEB SERVERS

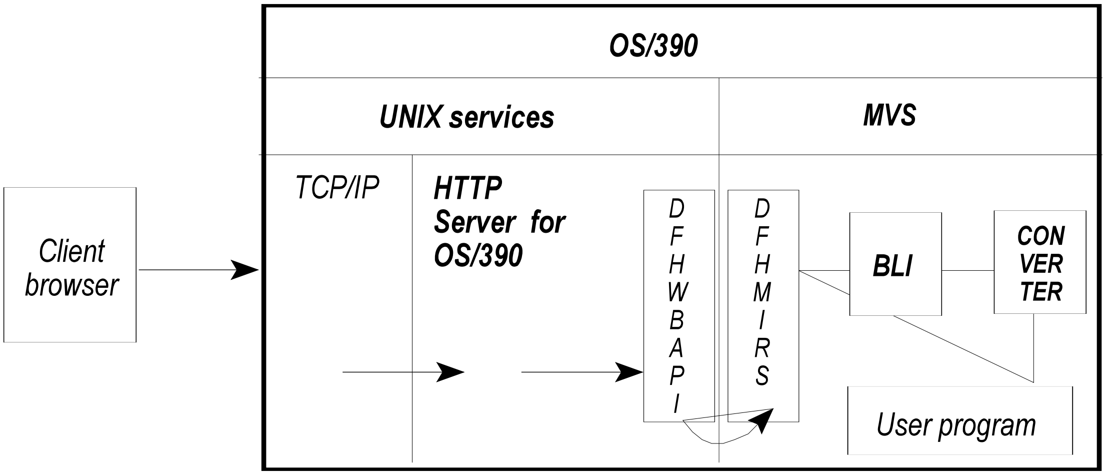
Objectives

- Describe how OS/390 Web servers communicate with CICS.
- Establish the restrictions relative to their use.
- Describe the External CICS Interface protocol used to connect non-CICS clients to CICS servers.

Overview

CICS provides a plugin for servers executing on OS/390 such as the Websphere application suite.

- It enables passthrough mechanisms from the web server through the EXCI and into CICS.
- It uses the Business Logic Interface.
- It provides more or less the same functions as a TCP/IP connection.



➔ The **Ex**ternal **C**ICS **I**nterface provides a communication interface for non-CICS clients that require access to a server program executing in the CICS address space.

- This type of communication was first used with TCP/IP **R**emote **P**rocedure **C**alls.
- Batch programs also have access to the facility.

➔ To use EXCI with a web server, the environment must be customized.

HTTP Server Configuration

MVS and CICS

- Check the following SIT parameters :

  ✓ ISC = YES
  ✓ IRCSTRT = YES

- Install RDO group **DFHWEB**.

- A generic EXCI connection must be installed : **EXCG** from group DFH$EXCI can be used along with the sessions for the EXCG connection.

- Define the following datasets to RACF program control (RACF will note the serial number of the volume containing the library and prevent the use of a different volume) :

  ✓ **CICSTS13.CICS.SDFHEXCI**
  ✓ **CICSTS13.CICS.SDFHDLL1**

- Modify the web server start-up procedure to concatenate the two above-mentioned datasets to the steplib.

HTTP Server Configuration

UNIX SERVICES

➔ In the directory containing the **httpd.cnf** file, issue the following command :

```
ln - e  DFHWBAPI  dfhwbapi.so
```

This command establishes a link from the server to DLL DFHWBAPI.SO, found in member DFHWBAPI in the SDFHDLL1 library.

➔ At least one directive must be added to the server's configuration field :

```
SERVICE/CICS/EXCI / *    / ETC / DFHWBAPI.SO:DFHSERVICE/CICSPROD/CICS/CSM3
```

- /**CICS/EXCI**/* is found in the source URL and will route the request to CICS using the plugin.
- **CICSPROD** is the VTAM applid of the targeted CICS region.
- **CICS** indicates that the converters are not to be used. This could be replaced by the name of a converter program.
- **CSM3** is the mirror transaction. If group **DFH$EXCI** was installed, the EXCI transaction could be used instead of CSM3.

➔ Some CICS-supplied templates contain references to graphic files. The following service statement should be added to the configuration to make sure graphics will be displayed properly.

```
SERVICE/CICS/EXCI / *   /home /dfhwbapi.so: DFHSERVICE /applid /DFHWBIMG /CSM3 / *
```

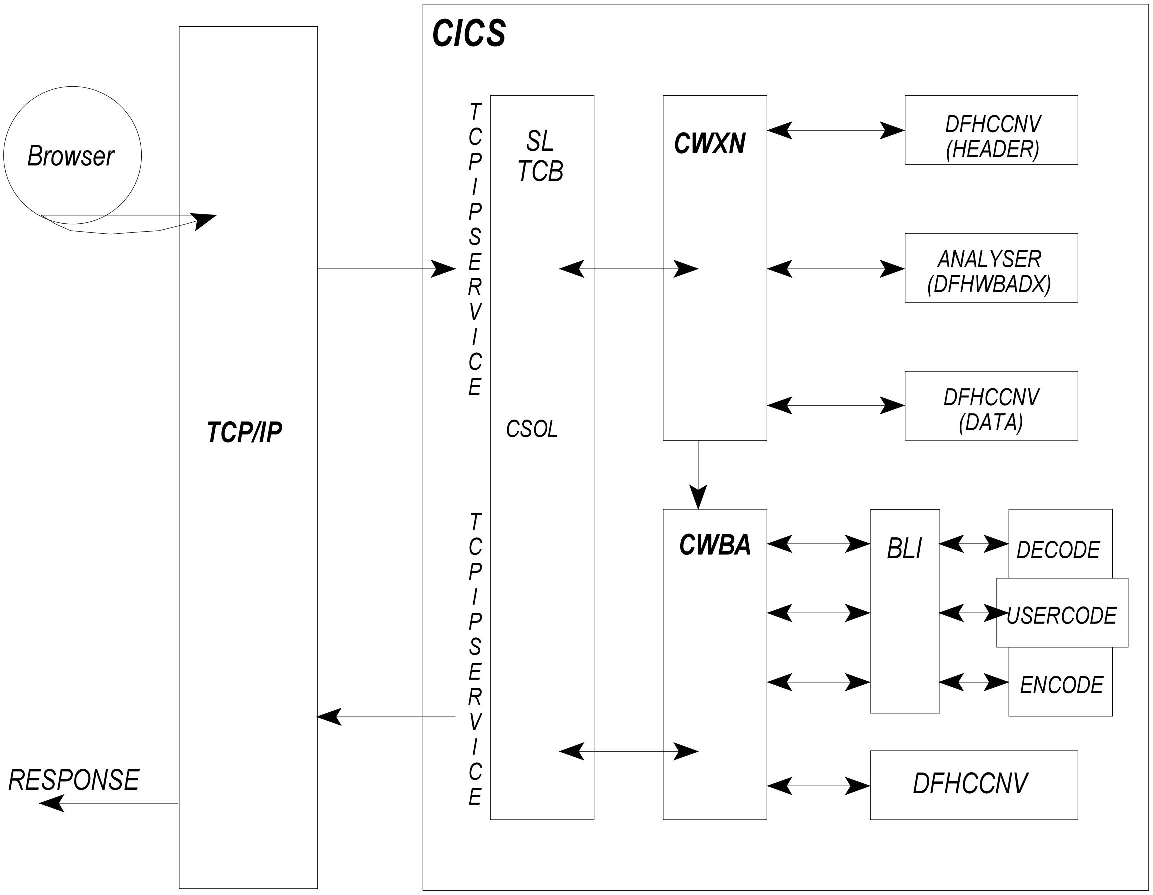
MODULE 5

CICS WEB SUPPORT

Objectives

- To describe the Web Support environment and its major components.
- To review the CICS/WEB application programming interface.
- To look at the DOCUMENT application programming interface.
- To provide design strategies that simplify the programming functions of CICS.

Overview


CICS
Browser
TCP/IP
RESPONSE
TCPIPSERVICE
SL
TCB
CSOL
TCPIPSERVICE
CWXN
DFHCCNV
(HEADER)
ANALYSER
(DFHWBADX)
DFHCCNV
(DATA)
CWBA
BLI
DECODE
USERCODE
ENCODE
DFHCCNV

Overview

CICS provides an environment aimed at supporting multiple flavours of web connections and design styles.

➔ Major components are :

- **CSOL** DFHSOL is a socket listener transaction executing on its own TCB (SL-TCB).
- **CWXN** This web-monitoring program receives data from the Web.
- **DFHWBADX** A user-replaceable module, that analyzes incoming URLS to decide which programs need to be started. CICS provides such an analyzer routine.
- **DFHWBBLI** A business logic interface module. This program either links directly to the user program (if the URL does not provide the name of a converter program) or to the converter program. The converter program will provide data to the user program via a commarea and vice versa.

  When BLI is used, the user program is strictly a server program and can be called from sources other than the Web.

  When no converter program was specified, BLI will automatically link to the user program, which has to be web aware.

NOTE
BLI, converter programs and user programs will be covered in the next module.

- **DFHCCNV** A data conversion routine needed to convert data from ASCII to EBCDIC or the reverse. DFHCCNV can be called upon on multiple occasions by DFHWBXN, DFHWBBLI or by an alias transaction.

Overview

➔ HTTP requests always consist of two components : a **header** and **user data.**

- The header is separated from user data by two consecutive CRLF characters.
- When requests are received, **CSOL** attaches **CWXN** which in turn receives the data and puts it into a TSQ.
- The header portion is translated to EBCDIC.
- The analyzer is given control. It looks at the header to determine the names of the converter and user programs to execute. If the converter program's name is **CICS**, it is assumed by the analyzer that no converter services are required and, consequently, no ENCODE/DECODE functions will be provided. After the analyzer is done, it returns control to the monitoring program.

➔ User data may need translation and **DFHCCNV** will be called if so.

- The alias transaction is started (the default is **CWBA**). The first program to execute must be **DFHWBA.** DFHWBA puts the information received from the analyzer in a commarea and links to the business logic interface.
- BLI invokes the converter program's DECODE function, if requested.
- BLI invokes the user program with a commarea. If no DECODE function executed, the commarea contains the request in its original form.
- After completion of the user program, the converter program's ENCODE function may be invoked, if necessary. At this point, BLI has a formatted response which is passed to the alias transaction. The response was formatted either by the converter program or by the user program.
- The alias transaction then invokes **DFHCCNV** to convert the response back to ASCII, if data was found in the commarea.
- The response is sent back to the client's browser by issuing **WEB WRITE** and **WEB SEND** commands.
- CICS' Socket Domain issues the necessary interface call to TCP/IP.

Customizing CICS for CWS

➔ **L**anguage **E**nvironment / 370 is required.

- SCEERUN is concatenated to STEPLIB.
- SCEERUN and SCEECICS are concatenated to DFHRPL.
- Member **CEECCSD**, from the SCEESAMP library, should be used to apply the necessary definitions in the CICS CSD.
- Add the **CEE** group to the list used to start CICS.
- Add the **CESE** and **CESO** TDQs to the system.
- Add the **CEEMSG** and **CEEOUT** DD statements to CICS' startup JCL.

➔ CWS definitions

- Group DFHWEB includes :
  - ✓ CWBA
  - ✓ CWXN
  - ✓ CWBG, for garbage collection
  - ✓ Programs used by the environment
  - ✓ TDQ CWBO with corresponding DD statement
  - ✓ TDQ CSDH and CSOO for the Document and Socket domains respectively.

➔ One or many TCPIPSERVICE resource definitions

- CICS group **DFH$SOT** contains sample definitions which can be used as a guide.
- Some **DOCTEMPLATE** resource definitions, to tell CICS where to load the templates from :
  - ✓ PDS dataset
  - ✓ TSQ One template per queue
  - ✓ TDQ One template per queue
  - ✓ LOAD module
  - ✓ CICS file
  - ✓ Exit program

Customizing CICS for CWS

- Sequence numbers, if used in PDS members, must be found in

✓ position 1 - 8 for RECFM = VB
✓ position 73 - 80 for RECFM = FB

✓ **They cannot be used in any other cases.**

✓ When changes are made to an existing template, its associated definitions must be re-installed.

- An exit program can be used to obtain HTML templates from another source, for example, a DB2 table. CICS will give control to the program and pass the following information via a commarea :

NOTE
You are not obligated to use pre-defined templates.

Fieldname	Description
DHTX_BUFFER_PTR	An area of memory where the template must be returned.
DHTX_BUFFER_LEN	The length of that buffer
DHTX_TEMPLATE_LEN	The actual length of the template

- The following copybooks will map the commarea :

Copybook	Compiler
DFHDHTXD	ASM
DFHDHTXH	C
DFHDHTXL	PL/1
DFHDHTXO	COBOL

Customizing CICS for CWS

- A conversion table may also be required if WEB API is not used. This will be discussed further in the next module.

- A number of tools can be used when developing web-aware CICS applications :

  ✓ The CICS WEB API

  ✓ The CICS DOCUMENT API

  ✓ Programs that can be LINKed to perform special functions, such as :

✧ **DFHWBUN**	the web unescaping program
✧ **DFHWBPA**	the parser program
✧ **DFH$WBST**	the state management program
✧ **DFHWBEP**	the web error program

  ✓ We will also mention two additional programs that could be used (but will not discuss them in this course) :

    ✧ **DFHWBENV**, a CICS/WEB program; the information provided by this program is now available using CICS' API.

    ✧ **DFHWBTL**, the CICS template manager, but the new DOCUMENT interface now provides the same functions.

The CICS/WEB Interface

The CICS/WEB application programming interface allows the retrieval of various elements from incoming HTTP requests.

- For older CICS/WEB applications, it means eliminating either the complex parsing logic required to get to the information or the LINK to DFHWBENV.
- The application program no longer needs to manipulate pointers.
- Documents created using DOCUMENT API can be sent using WEB API.

➔ All commands will start with the following string :

```
EXEC CICS WEB ...
```

➔ All commands will end using a command delimiter which varies according to which programming language was used :

Compiler	Delimiter
COBOL	END-EXEC
PL/1	; (semi-colon)
C	; (semi-colon)
Assembler	space

➔ The CICS/WEB API allows

- extracting HTTP information;
- getting information from a single record part of the header or browse through all header records embedded in the message;
- receiving the body of a document;
- writing headers;
- sending documents created by DOCUMENT API;
- retrieving a token associated with a document.

The CICS/WEB Interface

The application program never has “direct” access to a web client or its browser.

➔ Incoming requests are stored in TSQs by **DFHWBXN**. There are separate TSQs for input header and document body. WEB API commands will provide information via these queues.

- Three more TSQs could be created, mostly as a result of sending a response to the browser :

  ✓ Response line
  ✓ Output HTTP header
  ✓ Output document body or user data

➔ The output stream is created by the user program or **DFHWBA**. DFHWBA checks for TSQs created when executing EXEC CICS WEB SEND commands.

- If none are found, DFHWBA prepares a response and sends it out.
- When TSQs are found, DFHWBA assumes that a reply has already been prepared.

➔ TSQs for outgoing data are always rewritten, so only the data of the last CICS WEB SEND command is available.

➔ Web commands are:

- EXTRACT
- RECEIVE
- RETRIEVE
- READ HTTPHEADERS
- READ FORMFIELD
- WRITE
- STARTBROWSE / READNEXT / ENDBROWSE through HTTPHEADER and FORMFIELD

The CICS/DOCUMENT Interface

In traditional CICS programming, communication with the user is performed by sending information in 3270 format.

- Data is sent/received in a format called a **map**.
- BMS, in conjunction with **T**erminal **C**ontrol **P**rogram, manages the physical aspect of the communication.
- The CICS transaction is “connected” to the terminal so the programmer never worries about where the data needs to be sent.

➔ On the Web, the situation is quite different.

- Programs are independent from transactions and not “connected” to any terminal. The CICS/WEB interface simulates the environment.
- 3270 data streams are not supported; data is part of HTML documents, which can be created in many ways :

NOTE
Design issues will be discussed a little later in this module.

✓ Self-contained HTML templates stored in CICS

✓ A mixture of HTML templates and program instructions resulting in one document being transmitted

✓ Any of the above plus a mixture of graphic files stored on the web server.

➔ Document commands are handled by DH domain in CICS/TS.

- Documents created by DH domain can be sent to the client browser without having to manipulate the data.
- The EXEC CICS WEB SEND command accepts the DOCTOKEN returned by DH domain when a document is created.
- A document is 'alive' only for the duration of that one CICS task. If a document is needed for future communications with a client, it must be saved using known or traditional CICS techniques.

The CICS/DOCUMENT Interface

➔ Document commands are:

- CREATE
- INSERT
- SET
- RETRIEVE

MODULE 6

SECURITY, JAVA & IIOP

Objectives

- To review security in the CICS/WEB environment.
- To introduce Java and IIOP support.

Security

Overview

➔ In a traditional CICS environment :

- A transactions is submitted from a terminal.

- The transactions is internally attached to the terminal. This terminal is known as the primary facility.

- The sign-on process provides security attributes. These attributes are stored in control blocks attached to the primary facility. As a result, for most transactions, security is only available if

  ✓ they were started from a terminal;
  ✓ they were started by a program and the userid was either passed on the CICS command itself or provided by the resource definition.

- When a user does **not** sign on, work coming from this terminal executes under the default userid in CICS.

> *NOTE*
>
> ***DFHSIT DFLTUSER=CICSUSER***

➔ In a CICS/WEB environment :

- The traditional pseudo-conversational environment does not exist.

- Transaction codes are used to provide an environment familiar to CICS programs.

  ✓ An alias transaction must start **DFHWBA**.
  ✓ DFHWBA links to the user program.
  ✓ Because of this, it would be difficult to ensure security based on transaction code.

- The biggest problem comes from the absence of a primary facility.

  ✓ CESN cannot be used.
  ✓ EXEC CICS SIGNON cannot be used.
  ✓ The userid is obtained
    ✧ from a SSL certificate, if one exists;
    ✧ from a prompt, sent to the user via the CICS/BLI interface.

- If no userid was provided, a transaction will execute from the default userid of the CICS address space.

Security

Overview...

➔ When setting up a security environment, the following objectives should be targeted.

- **Authentication :** This is where the user will confirm he is who he claims to be. Several techniques can be used : userid/password or certificates.

- **Privacy** : Data exchanged between a user and CICS is not visible to anyone else. This can be accomplished with encryption.

- **Integrity** : Making sure that data has not been altered during transmission. This is done using hashing techniques at both the sender and receiver ends.

- **Authorization** : An authenticated user is allowed to use a particular service.

- **Accountability** : To provide assurances that a piece of work took place. This is accomplished by creating user journals.

➔ Therefore, on most CICS/WEB systems, security is a very important part of the design.

➔ Security implementation will depend on

- the nature of the data ?

- users : internals (company employees) or external (clients) ?

- the network : private or public ?

- the will to put some effort into it ?

➔ Two types of security are available and they are <u>not</u> mutually exclusive.

- **S**ecured **S**ocket **L**ayer, which provides encryption/decryption services

- non-SSL, which is a little easier and cheaper but does not provide encryption.

CORBA/IIOP

*The **I**nternet **I**ntro-**O**rb **P**rotocol is an industry standard that defines formats and protocols to provide client/server semantics for distributed object-oriented applications in a TCP/IP network.*

➔ Support for inbound requests is included in CICS/TS 1.3. CICS/TS 2.2 includes support for both inbound and outbound requests.

➔ A subset of CORBA services is provided. It is suitable for distributed objects that have evolved from existing CICS applications and therefore have the following characteristics :

- State by virtue of their explicit use of CICS resources rather than state that is managed by the object's request broker.
- Transaction and security managed by CICS facilities, as opposed to managed by CORBA facilities.
- Applications and their interface are pre-defined.

➔ The information required to communicate, such as operations available to the client or operation arguments, is provided by an interface defined using the **O**bject **M**anagement **G**roup / **I**nterface **D**efinition **L**anguage,

- This interface is used to code a set of interface definitions.
- Each method call is implemented as a CICS transaction.

IIOP and Java supports are changing quite considerably with CICS/TS 2.1. It is therefore recommended to check with IBM or at least read the Release Guide for CICS/TS 2.1 to make sure your design is accurate by the time it is ready to run under Z/OS.

CORBA-IIOP

Terminology

- **OMG** — **O**bject **M**anagement **G**roup is a consortium of software organizations who defined the CORBA architecture.
- **CORBA** — **C**ommon **O**bject **R**equest **B**roker **A**rchitecture is the specification for object-oriented computing.
- **ORB** — **O**bject **R**equest **B**roker is a CORBA component that acts as an intermediary between client and server applications.
- **IIOP** — **I**nternet **I**ntro-**O**rb **P**rotocol is an industry standard that defines formats and protocols to provide client/server semantics for distributed object-oriented applications in a TCP/IP network. It is also part of CORBA.
- **IDL** — **I**nterface **D**efinition **L**anguage is a language used in CORBA to describe the characteristics and behavior of an object.
- **Module** — Map to a Java package
- **Interface** — Map to a class a client requested. In IDL, an interface that defines the server object.
- **Operation** — An action that can be performed on an object. It is a map to a method.
- **IOR** — **I**nteroperable **O**ject **R**eference is used to provide enough information about the location of a server or an object.
- **Stub** or **proxy** — Generated by a client IDL compiler, it is used by the ORB to convert a local object reference to an IOR and invoke the translation of object datatypes from/to IIOP message syntax.
- **Skeleton** — Generated by a server IDL compiler, it is used by the ORB to parse a message into a method call on a local object.

CORBA-IIOP

Processing Flow

➔ The CICS TCP/IP Listener monitors ports for inbound requests.

- IIOP ports are identified by the TCPIPSERVICE resource definition.
- The request is received and a CICS transaction is started. For IIOP, this transaction is **CIOR** or an alias. The program started must be **DFHIIOP**.

➔ DFHIIOP retrieves the incoming request and matches its interface and operation (class and method) against templates.

- These templates were defined in CICS using a **REQUESTMODEL** resource definition.
- When a match is found, CICS starts the transaction associated with that REQUESTMODEL. The program associated with the transaction must be **DFHIIOPA.**
- If no match is found, CICS starts transaction **CIOD**.

➔ DFHIIOP then gives control to a user module in order to acquire a userid. The default module is **DFHXOPUS**.

➔ DFHIIOP then attaches the CICS transaction (CIOD or an alias).

➔ DFHIIOPA links to DFHJIIOP to handle the request.

- It analyzes the content of the request, which was passed in a commarea.
- It instantiates the target object.
- It demarshals the input parameters.
- It invokes the request method on the target object. This can access CICS resources and LINK to other CICS programs.
- It marshals the reply and returns it to DFHIIOPA for transmission back to the sender.

CORBA-IIOP

CORBA Services Support

➔ Name Services Support

- This service is not implemented in CICS. A stringified reference to the CosLifeCycle::GenericFactory implemented in the server can be written to a file using the GenFacIOR utility class. Make sure this reference is available to clients.

➔ Security

- Security is provided by CICS. All IIOP requests to CICS will run under a default userid unless you provide a program to generate a userid for each request.

➔ Lifecycle Support

- Only the CORBA GenericFactory is supported. It is implemented in program DFJGFAC.

➔ Location Service

- Not used in CICS. All object references refer either to a specific server or to a server group.

➔ The following services are not supported :

- Externalization
- Persistence
- Concurrency
- Interface Repository Framework

CORBA-IIOP

Building Java Programs for Servers Requirements

- OS/390 UNIX System Services
- A Java compiler
- VisualAge for Java, Entreprise Toolkit for OS/390
- CICS/TS 1.3 or later with Language Environment

➔ SIT Parameters

- Same as for WEB and SSL support
- Pay particular attention to EDSALIM. Because of the size of Java programs and the number of links to the various system facilities used, EDSALIM should be set to a very large number.
- Since CICS requires two transactions per request, MXT value should also be increased.

➔ .JAR files

- The CLASSPATH operand of the Java environment module (DFHJVM) should contain the following files :

✓	DFJCIDL.JAR	CICS IDC compiler
✓	DFJCORB.JAR	CICS ORB classes
✓	DFJCICS.JAR	JCICS API classes

➔ CICS Libraries

- CICSTS13.CICS.SDFJLOAD
- A PDSE for the JCICS program

➔ TCPIPSERVICE and REQUESTMODEL **resource definitions**

Java Virtual Machine

CICS provides support for running Java transactions in a CICS region under the control of a OS/390 JVM. Two types of programs are supported :

➔ Java program objects produced by a standard Java compiler are

- restricted to JCICS API (will be discussed later);
- run as standard CICS programs;
- loaded from a PDSE concatenated to a DFHRPL library.

➔ JVM programs are compiled to the bytecode, then stored and loaded from HFS datasets. The HFS directory is identified in the CLASSPATH statement, in the DFHJVM member of the CICS530.SDFHENV dataset.

➔ In addition to JCICS/API classes, the following packages are available :

●	java_io	to access HFS files,
●	java_net	to access OE sockets,
●	java_rmi	for bytecode interpretation,
●	java_lang	for application-level threading,
●	java_awt	for windowing support.

Java Virtual Machine

- PG Domain looks for the program definition and

  ✓ collects information about the program attributes;
  ✓ is responsible for executing LINK, XCTL, etc...
  ✓ does not load actual Java bytecode; it loads DFHCJVM, the CICS Java machine, which will then load the code from HFS.

- JVM programs can be used as CICS programs in the following situations :

  ✓ As initial programs
  ✓ As programs linked using EXEC CICS LINK
  ✓ As PCT programs
  ✓ As the target of a XCTL command
  ✓ As the target of a EXEC CICS HANDLE ABEND
  ✓ As user-replaceable modules.

- The following commands cannot be used with Java programs :

  ✓ EXEC CICS LOAD
  ✓ EXEC CICS RELEASE

- COBOL dynamic CALL to Java programs is not supported .

- The following DD statements must be added to the CICS startup deck :

```
//DFHJVM DD DSN=CICS530.SDFHENV(DFHJVM),DISP=SHR
//DFHCJVM DD DUMMY
```

JCICS

A new programming interface for use by CICS application programs written in Java.

➔ CICS Java classes (JCICS) provided Java access to CICS services. Access was traditionally available through CICS procedural API.

➔ Programs can be developed on a workstation or in a OS/390 UNIX shell. Two execution environments are available :

- VisualAge for Java Enterprise ToolKit for OS/390 brings Java bytecode into OS/390 executable files which are stored in a PDSE and then executed by CICS in a LE-run unit.
- MVS Java Virtual Machine executes on a J8-TCB supervised by CICS.

➔ Some JCICS classes can be used as JavaBeans. They are

✓ Program
✓ ESDS
✓ KSDS
✓ RRDS
✓ TDQ
✓ TSQ
✓ ATTACHINITIATION
✓ ENTERREQUEST

- These beans consist of properties and methods. They can be initiated in one of three ways :

✓ By calling **new( )** for the class itself (recommended)
✓ By calling **Beans.instantiate ( )** for a .ser file with property values set at design time
✓ By calling **Beans.instantiate ( )** for the name of the class with property values set manually.

➔ The following .jar files are stored in HFS, in the $CICS_HOME/CLASSES directory :

✓ dfgcics.jar	Required for compiling
✓ dfgwrap.jar	Used internally by CICS

- Sample programs are available in SDFHSAMP and HFS libraries.
- Documentation can be found in a JAVADOC HTML document located in the HFS file $CICS_HOME/DOCS/DFJCICS_DOCS.ZIP.