

# **CICS**

## **Customer Information Control System**

**Der am weitesten verbreitete, IBM proprietäre, Transaktionsmonitor.**

**Verfügbar unter den /390-Betriebssystemen MVS und VSE, sowie in modifizierter Form (als Encina Erweiterung) unter NT, OS/2, AIX, HPUX, Sinix, Solaris sowie Digital Unix.**

**Anwendungs-API besteht aus „CICS EXEC .....“ Kommandos, die in Standardprogrammiersprachen eingefügt werden.**

**Der CICS „Basic Mapping Support“ (BMS) stellt Benutzeroberflächen für *a/n* Bildschirme (Screens) zur Verfügung.**

**Alle Anwendungen und Dienste laufen als im Problemstatus ungeschützt voneinander innerhalb eines einzigen Adressenraums. Anwendungen und Ressource Managers laufen als Threads innerhalb dieses Adressenraums.**

**Literatur: R. K. Lamb: „Cooperative Processing using CICS“. McGraw-Hill 1993  
W. G. Bruno: „CICS, Mastering Command Level Coding“. Prentice Hall 1986**

# **Beispiel für ein CICS-Statement innerhalb eines COBOL-Programms**

```
EXEC CICS  
WRITEQ TS QUEUE('ACCTLOG') FROM(ACCTDTLO)  
LENGTH(DTL-LNG)  
END EXEC
```

**Ein existierender Datensatz „ACCTDTLO“ wird in eine temporäre Warteschlange ACCTLOG geschrieben, die als Log zur Datensicherung dient**

# **CICS Interprocess Communication (IPC)**

## **Transaction Routing**

**Transaktionen können zwecks Ausführung von einem CICS-System einem anderen CICS-System unverändert übergeben werden**

## **Function Shipping**

**Eine Anwendung in einem CICS-System kann auf Daten eines anderen CICS-Systems zugreifen**

## **Distributed Transaction Processing**

**Eine Transaktion in einem CICS-System kann eine Transaktion in einem anderen CICS-System aufrufen und mit ihr synchron kommunizieren**

## **Distributed Program Link**

**Ein Programm eines CICS-Systems kann mit Hilfe des Kommandos EXEC CICS LINK eine Verbindung mit einem Programm eines anderen CICS-Systems aufnehmen**

# **3270 Protokoll**

**Ursprünglich für nicht-intelligente Terminals eingesetzt**

**Arbeitet mit einem zeichenorientierten Screen, bestehend aus 24 Zeilen mit je 80 a/n Zeichen**

**Jede der  $24 * 80 = 1920$  Positionen ist vom Anwendungsprogramm im Hostrechner individuell adressierbar (gewisse Ähnlichkeit mit dem X-Window-Protokoll)**

**Normalerweise werden „Felder“ angesprochen. Ein Feld ist eine Folge von Zeichenpositionen. Felder können gelesen und geschrieben werden**

**Eine CICS-Utility, Basic Mapping Support (BMS) erleichtert dem Anwendungsprogrammierer die Entwicklung von Screens**

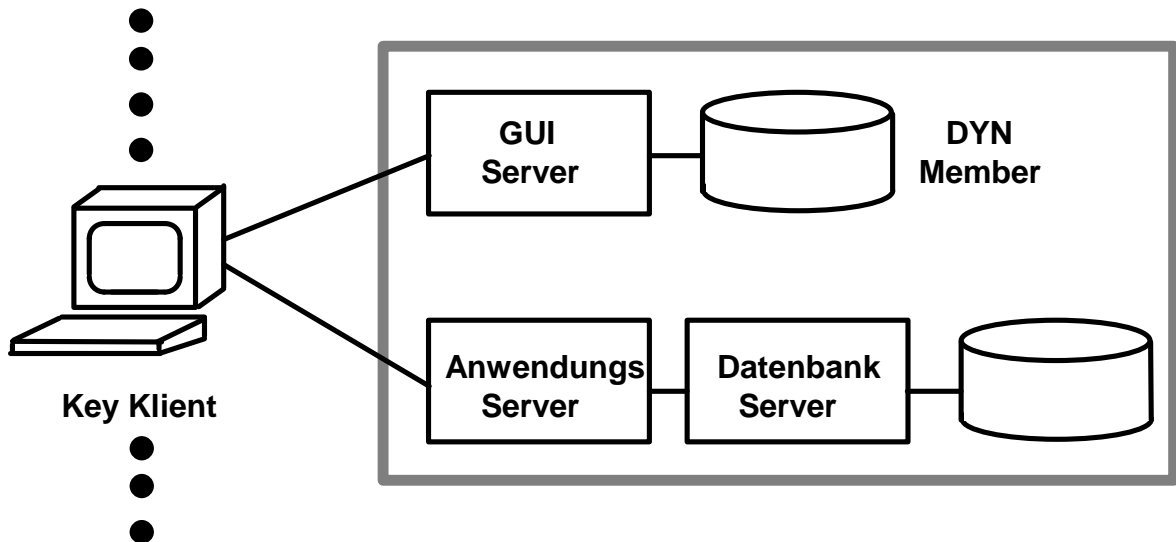
**Auf heutigen Arbeitsplatzrechnern mit Hilfe der 3270 Emulation implementiert**

```

Col: 1234567890123456789012345678901234567890123456789012345678901234567890123
Row:
1  +ACCOUNT FILE:+RECORD DISPLAY
2
3  +ACCOUNT NO:+_____ +SURNAME: +_____
4  +FIRST: +_____ + MI:+_+ TITLE:+_____|
5  +TELEPHONE:+_____ +ADDRESS: +_____
6  +_____
7  +_____
8  +OTHERS WHO MAY CHARGE:
9  +_____
10 +_____
11
12 +NO. CARDS ISSUED:+_+ DATE ISSUED:+_+ +_+ REASON:+_|
13 +CARD CODE:+_ | +APPROVED BY:+_ | +SPECIAL CODES:+_+_+_ |
14
15 +ACCOUNT STATUS:+_+ CHARGE LIMIT:+_____
16
17 +HISTORY: BALANCE BILLED AMOUNT PAID AMOUNT
18 + / / / / /
19 + / / / / /
20 + / / / / /
21
22 +_____ (message area) _____

```

## Beispiel eines CICS Basic Mapping Support (BMS) a/n Bildschirms

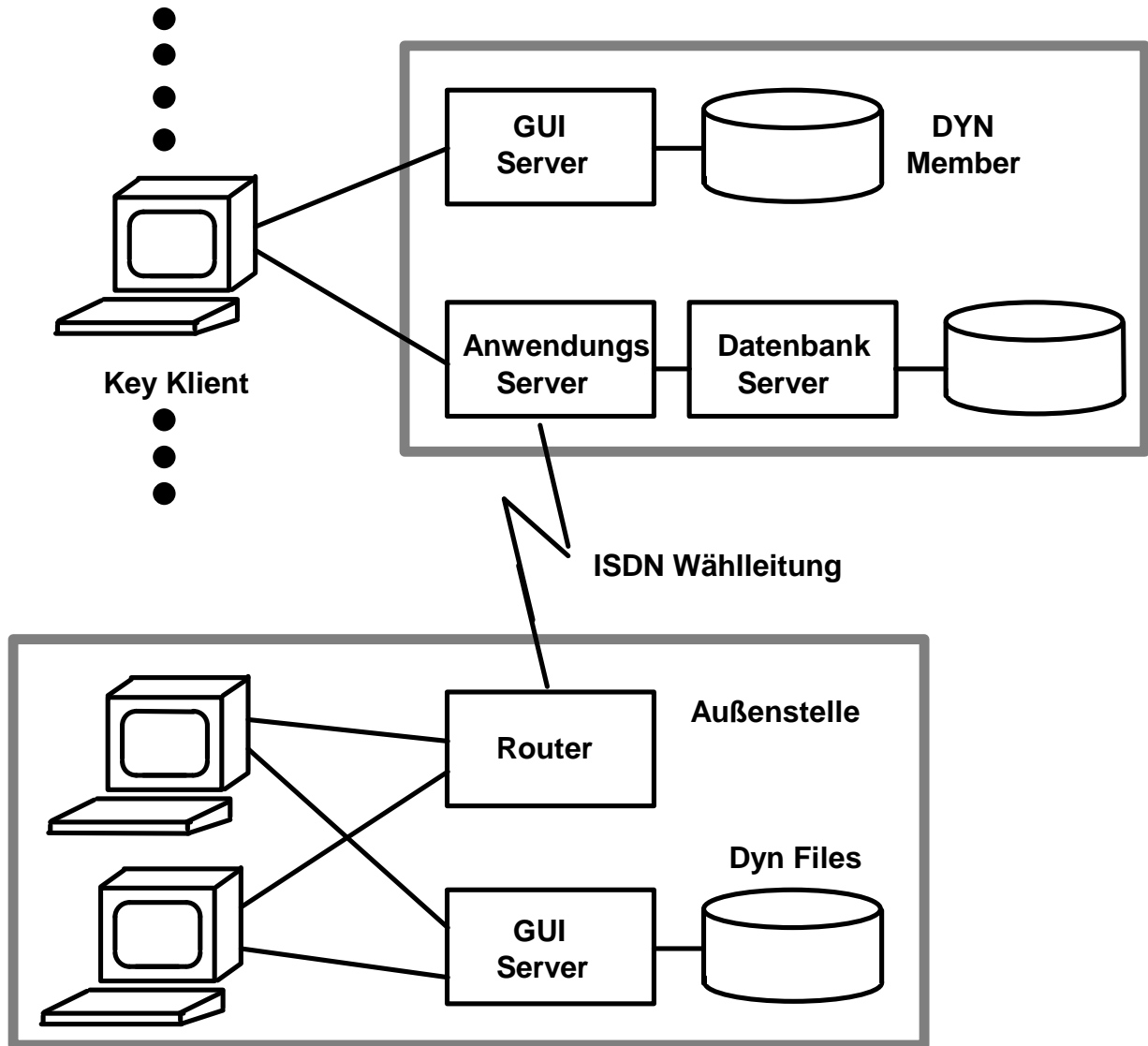


## Screen Management

Die Bildschirmdarstellung besteht aus 2 Komponenten: Screen und Inhalt.

Zusätzlich zum Anwendungsserver und dem Datenbankserver (z.B. Oracle, DB2) existiert ein GUI Server. Der GUI Server enthält „Dyn“ files. Jede Dyn file beschreibt einen „Screen“, das Aussehen eines Bildschirm-Fensters des Klienten. Das Fenster wird mit Ausgabedaten gefüllt, die das Anwendungsprogramm erstellt. Je nach Transaktionsart wird die dazugehörige Dyn File in den Klienten geladen. Jeder Klient enthält ein GUI Programm, welches die Dyn File in einen Screen übersetzt. Der Screen bestimmt, wie und wo die Ausgabedaten in dem Fenster dargestellt werden.

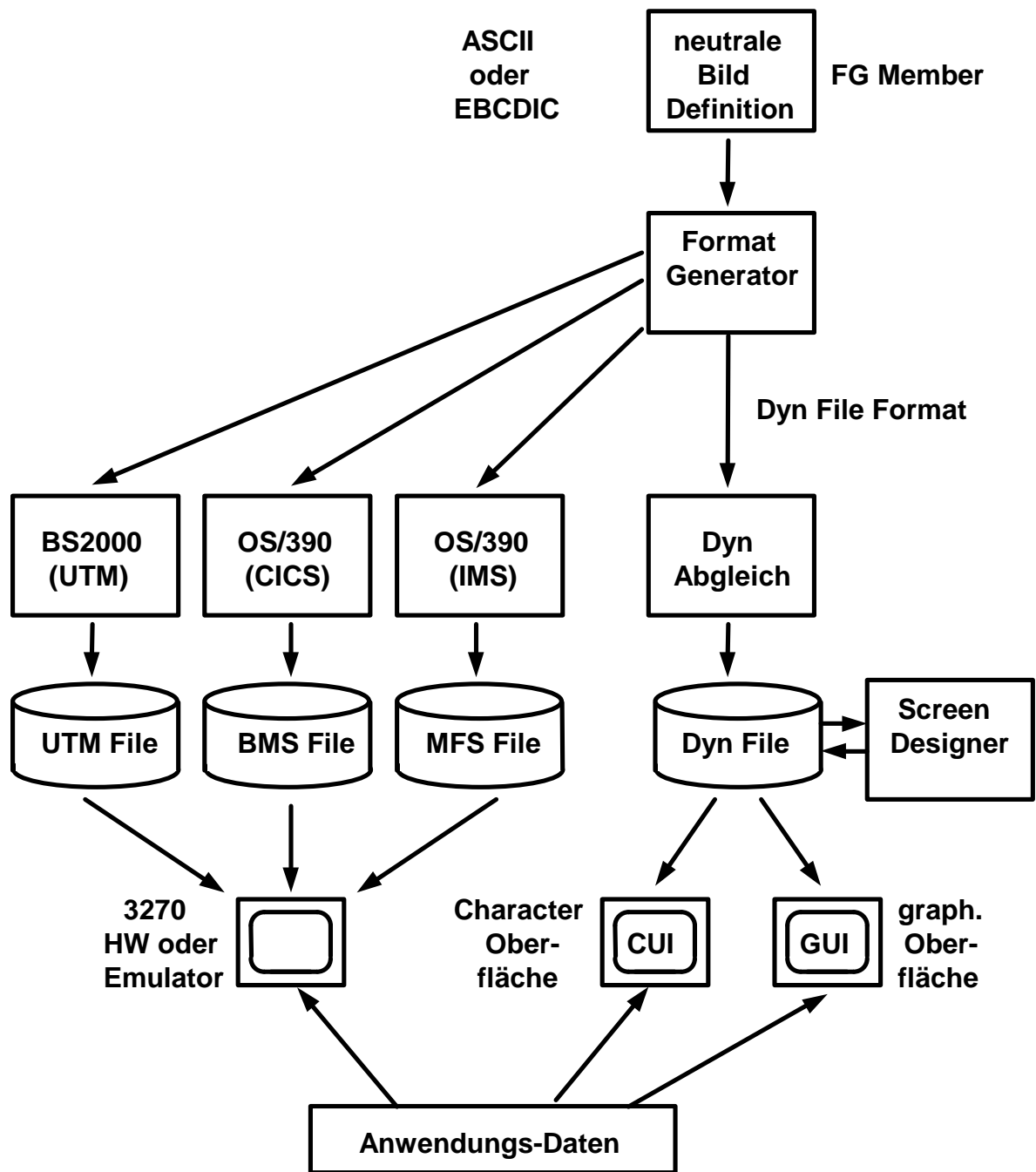
Prinzipiell ist es auch möglich, die Dyn files auf jedem PC zu installieren. Ein wesentlicher Vorteil des zentralen GUI Servers ist die Vereinfachung bei Änderungen und Updates. Ein mittleres System kann mehrere hundert Screens enthalten.



### Dezentrale Konfiguration mit verteilten GUI Servern.

Bei Außenstellen, die mit der Zentrale über eine WAN Leitung verbunden sind, kann zur Verringerung der Netzwerkbelastung ein entfernter GUI Server aufgestellt werden, welcher die Dyn Files lokal vorhält.

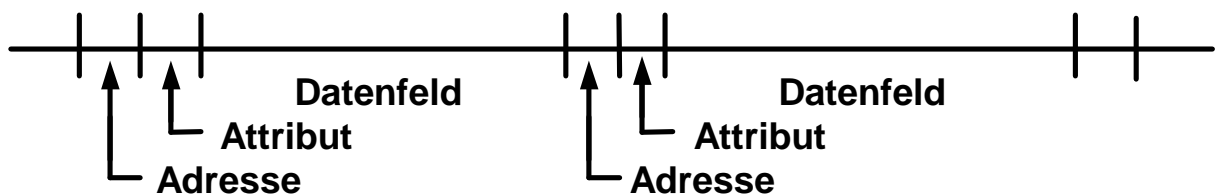




## Generierung der GUI Oberfläche

## 3270 Bildschirm Datenausgabe

Das Anwendungsprogramm erzeugt die Datenausgabe an den Bildschirm im Format



Das Datenfeld enthält eine variable Anzahl von  $a/n$  Zeichen, welche auf dem Bildschirm in einem Feld wiedergegeben werden.

Das Attributfeld (3 Bytes bei BMS/CICS) enthält Steuerzeichen, welche Information über die Art der Wiedergabe des folgenden Datenfeldes enthalten, z.B. Darstellung in roter Farbe, blinkender Cursor, Font, andere...

Das 3270 Protokoll verwendet eine Untermenge der 256 Zeichen des ASCII oder EBCDIC Zeichensatzes zur Datenwiedergabe auf dem Bildschirm. Die restlichen Zeichen werden als Steuerzeichen für Steuerungszwecke eingesetzt.

Der 3270 Bildschirm besteht aus 24  $a/n$  Zeilen mit je 80 fixed Font-Width Zeichenpositionen pro Zeile. Das Adressenfeld kennzeichnet eine der  $24 \times 80 = 1920$  Zeichenpositionen auf dem Bildschirm.



<b>CICS API</b>
<b>Encina</b>
<b>Distributed Computing Environment</b>
<b>Betriebssystem</b>

## **Implementierung von CICS für Unix und NT**

# Encina Transactional-C (TRAN-C)

Ergänzung von C und C++. Baut auf DCE auf.

Transactional-C ist ein Satz Macros und Bibliotheksroutinen (kein Precompiler), die mit jedem Standard C oder C++ Compiler arbeiten.

Definiert spezifische Schlüsselwörter, z.B. `transaction`, `onCommit`, `onAbort`.

```
transaction { ...
    debit (salaryExpense, amount);
    credit (accountsPayable, amount);
    enterAuditData(employeeIdentifier, amount, date);
    ... }

onCommit
    printf(„Transaction succeeded.“);
onAbort
    printf(„Transaction failed.“);
```

Die Statements zwischen {...} des „transaction“ Statements stellen eine einzige LUW dar. Ist die Transaktion erfolgreich, erfolgt automatisch ein Aufruf der `onCommit` Anweisung. Im Fehlerfall erfolgt automatisch ein Aufruf der `onAbort` Anweisung.