# Tutorial 7

## MQSeries

In dieser Aufgabe legen wir uns eine Message Queue an, schicken an diese Nachrichten und lesen diese danach wieder aus.

Hinweis: Dieses Tutorial wurde unter Verwendung der Benutzer-ID "PRAKT20" erstellt. In allen Dateinamen müssen Sie "PRAKT20" durch ihre eigene Benutzer-ID ersetzen. Außerdem wird oft in den Erklärungen PRAKTxx verwendet, wobei die xx ihre Praktikumsnummer sein soll.

*Aufgabe: Arbeiten Sie nachfolgendes Tutorial durch.*

## MQSeries

Für kommerzielles Messaging und Queuing stellt MQSeries ein nahezu Plattform-unabhängiges Middleware-Produkt dar. Es wird besonders in Hochgeschwindigkeits-Implementierungen von verteilten Anwendungen benutzt. MQSeries-Applikationen können mit minimalem Aufwand entwickelt und getestet werden.

Da MQSeries auf einer Vielzahl von Plattformen lauffähig ist, können Programme infolgedessen miteinander über ein Netzwerk von unterschiedlichen Komponenten, wie z.B. Prozessoren, Subsystemen, Betriebssystemen und Kommunikations-Protokollen kommunizieren. MQSeries-Programme verwenden ein konsistentes Application Program Interface (API) auf allen Plattformen. In der Abbildung 1 sind die Haupt-Komponenten einer MQSeries-Anwendung dargestellt.

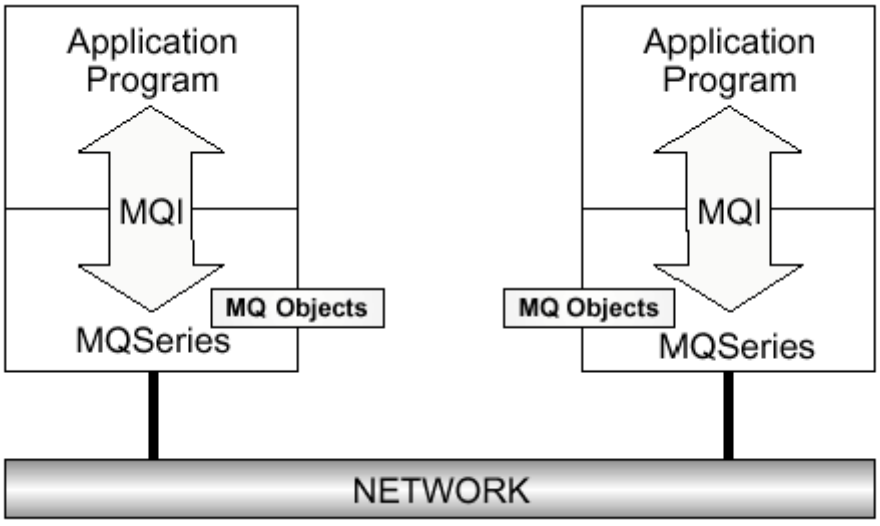


**Abbildung a: MQSeries während der Laufzeit**

MQSeries ermöglicht die Kommunikation zwischen verschiedenen Anwender-Programmen. Ein Programm A stellt eine Nachricht zur Verfügung und schreibt diese in eine Queue. Das Programm B liest diese Nachricht aus der Queue und verarbeitet sie.

Die Programme A und B verwenden jeweils eine besondere Anwendungsprogrammier-Schnittstelle (API, Application Programming Interface), um Nachrichten in die Queue zu schreiben bzw. aus der Queue zu lesen. Das MQSeries-API wird Message Queue Interface (MQI) genannt.

Möglicherweise läuft Programm B gerade nicht, wenn Programm A eine Nachricht in die Queue stellt. In diesem Fall speichert die Queue die Nachricht, bis das Programm B gestartet wird und diese abrufen kann. Andererseits kann Programm A nicht mehr aktiv sein, wenn Programm B die Nachricht aus der Queue abruft. Für MQSeries besteht keine Notwendigkeit, dass beide kommunizierende Programme gleichzeitig aktiv sein müssen, d.h. MQSeries ermöglicht eine asynchrone Verarbeitung.

Bevor wir uns der eigentlichen Aufgabe zuwenden, müssen einige Objekte bzw. Definitionen getroffen werden, mit denen MQSeries arbeitet.

## Message

Eine Message besteht aus zwei Teilen:

- Daten, die vom einem Programm zu einem anderen gesendet werden
- Message-Deskriptor oder Message-Header

Der Message-Deskriptor identifiziert die Message (Message-ID) und enthält Steuerinformationen (Attribute), wie z.B. Message-Type, Zeit-Ablauf, Korrelations-ID, Priorität und Namen der Antwort-Queue.

Eine Message kann bis zu 4 MByte oder 100 MByte lang sein. Die Länge ist von der benutzten MQSeries-Version abhängig. MQSeries-Version 5 (für verteilte Plattformen) unterstützt eine maximale Message-Länge von 100 MByte. Auf unserer Multiprise 2003 an der Universität Leipzig ist MQSeries Vers. 2.1 installiert.

## Queue Manager

Die MQSeries-Software, der die Queues zugeordnet sind und die diese verwaltet, wird als Queue Manager bezeichnet. Der Queue Manager (QMGR) verwaltet die zu MQSeries gehörenden Objekte, z.B. Queues, Channels, Prozesse usw.

Ein Queue Manager stellt zusätzlich das Message Queue Interface (MQI) zur Verfügung. Über das MQI greift eine Anwendung auf ihre Queues und die dort enthaltenen Nachrichten zu. Das MQI implementiert eine einfache Anwendungsprogrammier-Schnittstelle, die für alle unterstützten Plattformen einheitlich ist. Durch das MQI werden Anwendungen vom Queue Manager getrennt.

Eine Anwendung muss zunächst eine Verbindung zu einem Queue Manager herstellen, bevor sie auf seine Ressourcen zugreifen kann. Dafür gibt es den Aufruf MQCONN oder MQCONNX. Wenn die Anwendung keine Verbindung zum Queue Manager mehr benötigt, wird diese mit MQDISC abgebaut.

Um auf eine Queue zuzugreifen, muss eine Anwendung zunächst diese Queue öffnen. Letzteres erfolgt mit Hilfe des Aufrufs MQOPEN. Mit MQCLOSE wird dieser Zugriff beendet und die Queue wieder geschlossen.

Bei geöffneter Queue benutzt die Anwendung den Aufruf MQPUT, um eine Nachricht in eine Queue zu schreiben. Mit dem Aufruf MQGET kann eine Nachricht aus der Queue gelesen werden. Mit Hilfe des Aufrufs MQPUT1 kann eine Anwendung in einem Schritt eine Queue öffnen, eine Nachricht in die Queue schreiben und die Queue wieder schließen.

Queue, Queue Manager und Prozesse sind Beispiele für MQSeries-Objekte.

Bei der Installation von MQSeries auf einem Server muss der Queue Manager vom Systemprogrammierer implementiert werden.

**Der Name des Queue Managers in unserer MQSeries-Installation ist MQA1.**

## Queue

Wenn eine Anwendung eine Nachricht in eine Queue schreibt, sorgt der Queue Manager dafür, dass die Nachricht sicher gespeichert wird, wiederherstellbar (recoverable) ist und nur einmal an die empfangende Anwendung zugestellt wird. Das gilt auch, wenn eine Nachricht an eine Queue eines anderen Queue Managers gesendet werden muss. Dieser Mechanismus im Rahmen von MQSeries wird als "gesichertes Zustellen" bezeichnet.

Es wird zwischen persistenten und nichtpersistenten Messages unterschieden. Persistente Messages werden im MQSeries Log mitgeschrieben und sind nach einem Neustart des QMGRs wieder verfügbar. Nichtpersistente Messages sind nach einem Neustart des QMGRs nicht mehr vorhanden.

Beim Öffnen einer Queue durch eine Anwendung bestimmt der Queue Manager, ob es sich um eine "lokale Queue" oder um eine "remote Queue" handelt. Im Fall der lokalen Queue gehört diese zu dem Queue Manager, mit dem die Anwendung verbunden ist. Eine Queue heißt dagegen "remote", wenn sie zu einem anderen Queue Manager gehört. Eine "remote" Queue ist ein Verweis auf eine Queue eines entfernten Queue Managers.

Wenn eine Anwendung den Aufruf MQPUT ausführt, schreibt der Queue Manager die Nachricht in die lokale Queue. Handelt es sich um eine "remote" Queue, erhält der Queue Manager aus deren Definition die Informationen, die nötig sind, um die Nachricht richtig zuzustellen (Name des entfernten QMGRs, Name der entfernten Queue, Name der zugehörigen Transmission Queue). Die Nachricht wird in eine Zwischen-Queue (Transmission Queue) gestellt, die dem entfernten QMGR zugeordnet ist.

Anschließend ist es die Aufgabe des Message Channel Agent (MCA), die Nachricht von der Transmission Queue zu lesen und über das Netz an einen empfangenden MCA zu senden. Letzterer schreibt die Nachricht in die Ziel-Queue. Danach wird die Nachricht in der Transmission Queue gelöscht.

**In unserer Übung muss zunächst eine lokale Queue angelegt werden.**

## Anwender-Programme

Bei der Kommunikation zweier Anwender-Programme erfolgt der Datenaustausch über das MQI. Letzteres stellt eine einfach strukturierte CALL-Schnittstelle mit einer begrenzten Anzahl möglicher Aufrufe und umfangreichen Optionen für jeden Aufruf dar. Gesetzte Standard- und Anfangswerte garantieren aber einen einfachen und schnellen Start von Anwendungen.

Das MQI verwendet eine Reihe von Strukturen und Konstanten. Mit MQSeries werden Dateien und Copy Books mitgeliefert, die die Definitionen dieser Strukturen und ihrer Felder sowie die Definitionen der aussagekräftigen symbolischen Namen enthalten, die innerhalb der Programmlogik für Konstanten verwendet werden.

Ein Programm redet direkt mit seinem Queue Manager. Er residiert auf demselben Prozessor oder Domäne (für Clients) wie das Programm selbst.

Wenn die Verbindung zwischen einem Client und seinem Server unterbrochen ist, können keine API-Calls ausgeführt werden, da sich alle Objekte auf dem Server befinden.

Insgesamt existieren 13 APIs; diese sind in der Abbildung 20 aufgelistet. Die wichtigsten davon sind:
MQCONN, MQOPEN, MQPUT, MQGET, MQCLOSE und MQDISC.

| | |
|---|---|
| **MQCONN** | Connect to a queue manager |
| **MQDISC** | Disconnect from a queue manager |
| **MQOPEN** | Open a specific queue |
| **MQCLOSE** | Close a queue |
| **MQPUT** | Put a message on a queue |
| **MQGET** | Get a message from a queue |
| **MQPUT1** | MQOPEN + MQPUT + MQCLOSE |
| **MQINQ** | Inquire properties of an object |
| **MQSET** | Set properties of an object |
| **MQCONNX** | Standard or fastpath bindings |
| **MQBEGIN** | Begin a unit of work (database coordination) |
| **MQCMIT** | Commit a unit of work |
| **MQBACK** | Back out |

**Abbildung b: MQSeries APIs**

Es folgen Kommentare zu den verschiedenen APIs:

**MQCONN** stellt eine Verbindung mit einem Queue-Manager mit Hilfe von Standard-Links her.

**MQCONNX** realisiert eine Verbindung mit einem Queue-Manager über schnelle Verbindungswege (Fastpath). Fastpath-PUTs und -GETs sind schneller, die Anwendung muss aber gut ausgetestet werden. Die Applikation und der Queue-Manager laufen in demselben Prozess. Wenn die Anwendung zusammenbricht, fällt auch der Queue-Manager aus. Dieser API-Call ist neu in MQSeries Version 5.

**MQBEGIN** startet eine Arbeitseinheit, die durch den Queue-Manager koordiniert wird und externe XA-kompatible Ressource-Manager enthalten kann. Dieses API ist mit MQSeries Version 5 eingeführt worden. Es wird für die Koordinierung der Transaktionen , die Queues (MQPUT und MQGET unter Syncpoint-Bedingung) und Datenbank-Updates (SQL-Kommandos) verwenden.

**MQPUT1** öffnet eine Queue, legt eine Message darauf ab und schließt die Queue wieder. Dieser API-Call stellt eine Kombination von MQOPEN, MQPUT und MQCLOSE dar.

**MQINQ** fordert Informationen über den Queue-Manager oder über eines seiner Objekte an, wie z.B. die Anzahl der Nachrichten in einer Queue.

**MQSET** verändert einige Attribute eines Objekts.

**MQCMIT** gibt an, dass ein Syncpoint erreicht worden ist. Die als Teil einer Arbeitseinheit abgelegten Messages werden für andere Applikationen verfügbar gemacht. Zurück gekommene Messages werden gelöscht.

**MQBACK** teilt dem Queue-Manager alle zurück gekommenen PUT´s- und GET´s-Nachrichten seit dem letzten Syncpoint mit. Die abgelegten Messages als Teil einer Arbeitseinheit werden gelöscht. Zurück gekommene Messages werden wieder auf der Queue abgelegt.

**MQDISC** schließt die Übergabe einer Arbeitseinheit ein. Das Beenden eines Programms ohne Unterbrechung der Verbindung zum Queue-Manager verursacht ein "Rollback" (MQBACK). MQSeries für AS/400 verwendet nicht MQBEGIN, MQCMIT, MQBACK. Die Commit-Operation-Codes der AS/400 werden dagegen eingesetzt.

**Für die Lösung dieser Übung ist es notwendig, dass der Queue Manager gestartet wurde. Sollte das nicht der Fall sein, dann muss der Queue Manager von einem authorisierten Nutzer im SDSF SYSLOG mit**

**Command Input ===> / ! MQA1 START QMGR**

**gestartet werden.**

Für unsere Übung sind folgende Teilaufgaben notwendig:

- Local Queue anlegen
- Module A (PUT) und B (GET) suchen
- Bibliotheken PRAKTxx.MQA1.USERJCL anlegen,
  die die Member MQPUT und MQGET beinhaltet.
- Die Member MQPUT und MQGET bearbeiten.
- MQPUT und MQGET ausführen
- Ergebnis-Ausgabe überprüfen

Für die Kommunikation der beiden Anwender-Programme ist eine "local Queue" erforderlich. Um letztere zu generieren, wählen wir aus dem Custompac Master Application Menu (CMAM) das "MQSeries for OS/390-Main Menu" aus, d.h. wir geben "m" als Option ein und betätigen die Enter-Taste.

```
CUSTOMPAC MASTER APPLICATION MENU
 OPTION ===> m                                          SCROLL ===> PAGE

   IS  ISMF     - Interactive Storage Management Facility
   P   PDF      - ISPF/Program Development Facility
   ATC ATC      - Application Testing Collection
   ART ARTT     - Automated Regression Testing Tool
   DB2 DB2      - Perform DATABASE 2 interactive functions
   QMF QMF      - QMF Query Management Facility
   C   CPSM     - CICSPlex/SM
   M   MQ       - MQSeries
   IP  IPCS     - Interactive Problem Control Facility
   OS  SUPPORT  - OS/390 ISPF System Support Options
   OU  USER     - OS/390 ISPF User Options
   SM  SMP/E    - SMP/E Dialogs
   SD  SDSF     - System Display and Search Facility
   R   RACF     - Resource Access Control Facility
   DI  DITTO    - Data Interfile Transfer, Testing and Operations
   HC  HCD      - Hardware Configuration Definition
   S   SORT     - DF/SORT Dialogs
   BMR BMR READ - BookManager Read (Read Online Documentation)

  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

**Abbildung 1: Screen von CMAM mit Eingabe "m"**

Als Ergebnis unserer Aktion erscheint das Main Menu von MQSeries.

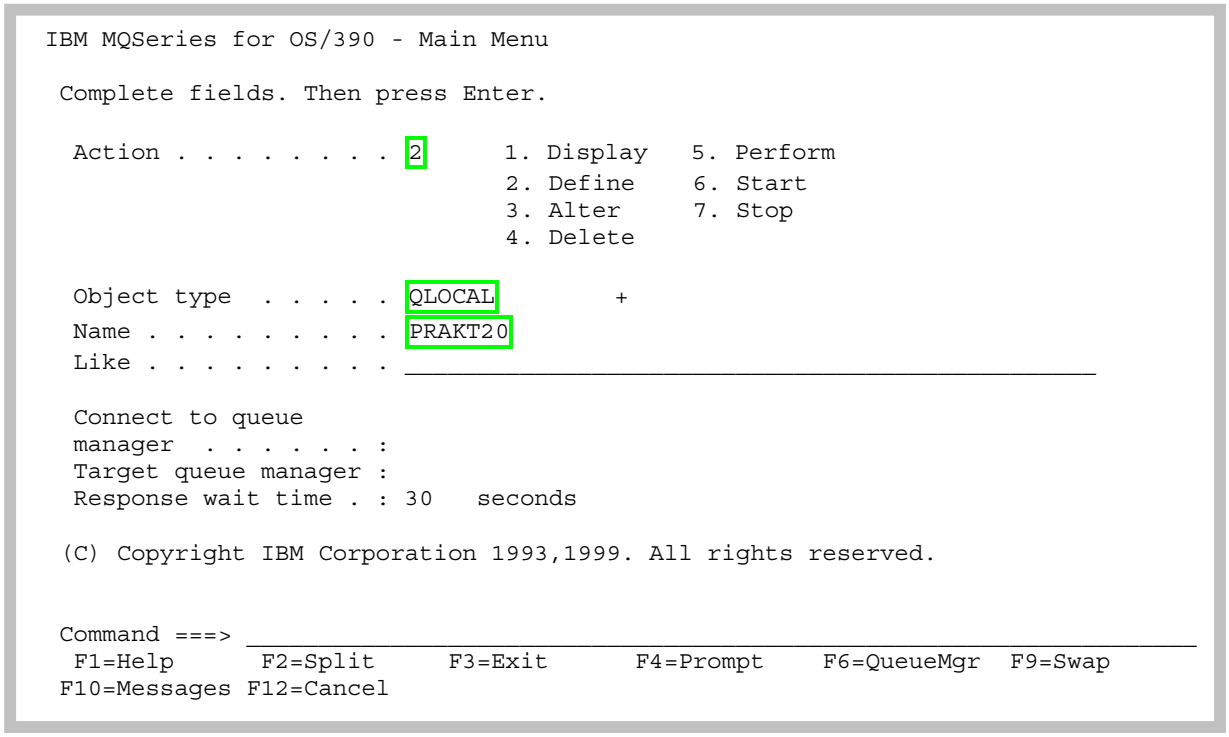


**Abbildung 2: MQSeries for OS/390 - Main Menu**

In dem Main Menu tragen wir in der Zeile "Action" eine 2 (Define) ein. Als "Object type" und "Name" werden QLOCAL bzw. PRAKTxx eingegeben. Alle restlichen Einträge werden ungeändert übernommen. Damit wird eine lokale Queue definiert mit dem Namen "PRAKTxx". Letztere gehört zu dem Queue Manager MQA1.

Hinweis: Falls die Fehlermeldung "CSQO040I Object named PRAKT20 of type QUEUE already exists." erscheint, existiert schon eine lokale Queue mit dem Namen PRAKTxx. Löschen Sie bitte die alte Queue, indem Sie im Panel "IBM MQSeries for OS/390 - Main Menu" in der Zeile "Action" eine 4 eingeben. Als "Object type" und "Name" werden QLOCAL bzw. PRAKTxx eingegeben. Achten Sie darauf, dass in die beiden Zeilen "queue manager" jeweils "MQA1" eingetragen ist. Dreimaliges Betätigen der Eingabetaste löscht die Queue. Die Meldung: "CSQ9022I !MQA1 CSQMUQLC ' DELETE QLOCAL' NORMAL COMPLETION" bestätigt, das der Löschvorgang erfolgreich abgeschlossen wurde.

Um im Main Menu Änderungen der Felder "Connect to QMGR" und "Target QMGR" vornehmen zu können, betätigen wir die Steuertaste F6! Wir tragen 2 mal „MQA1“ als Queue Manager ein. (Siehe Abb. 3)

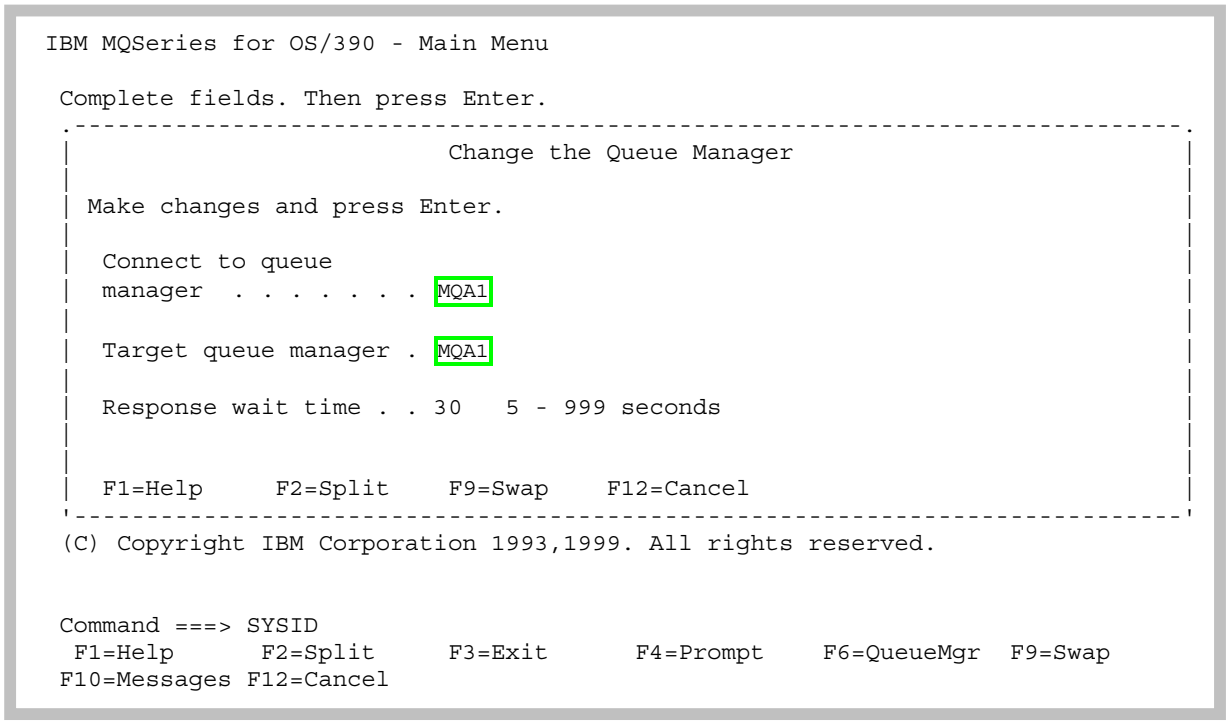


**Abbildung 3: "Change the Queue Manager"**

Durch Enter werden diese Werte übernommen und wir befinden uns bei Abb. 4.

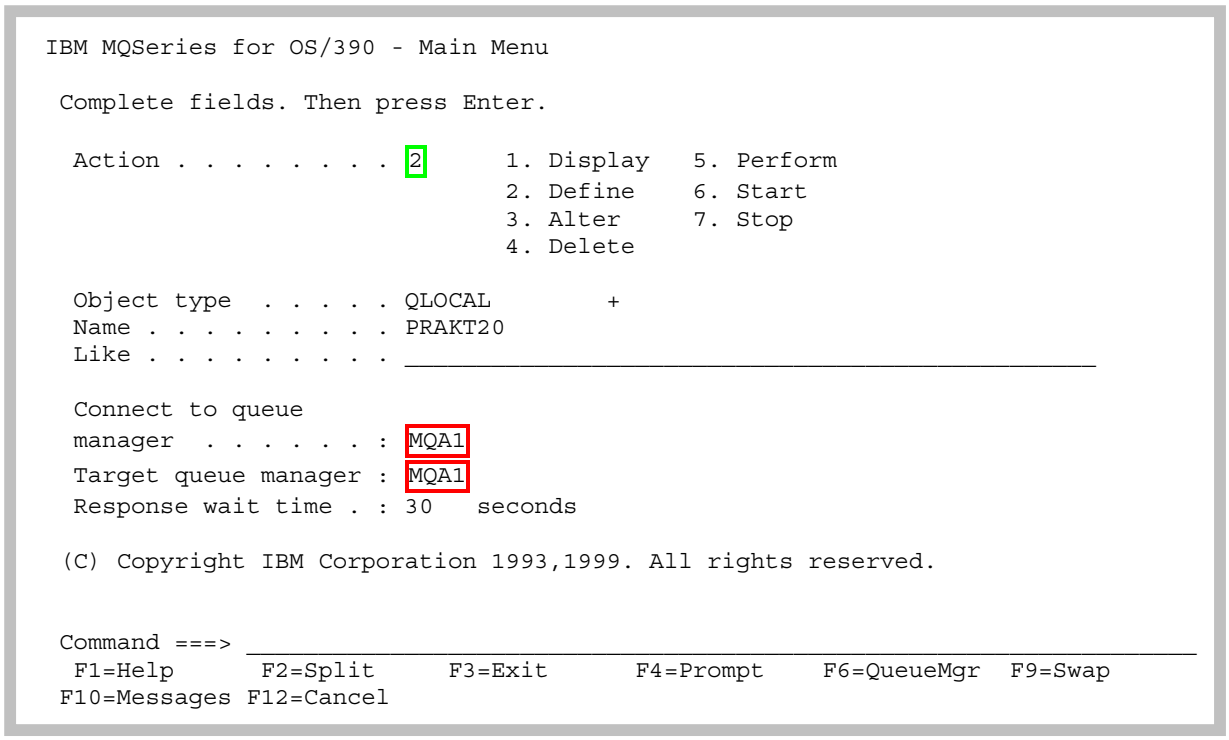


**Abbildung 4: MQSeries Main Menu**

Anschließend wird die Enter-Taste betätigt. Es erscheint als Ergebnis ein weiterer Screen für die Definition der lokalen Queue.

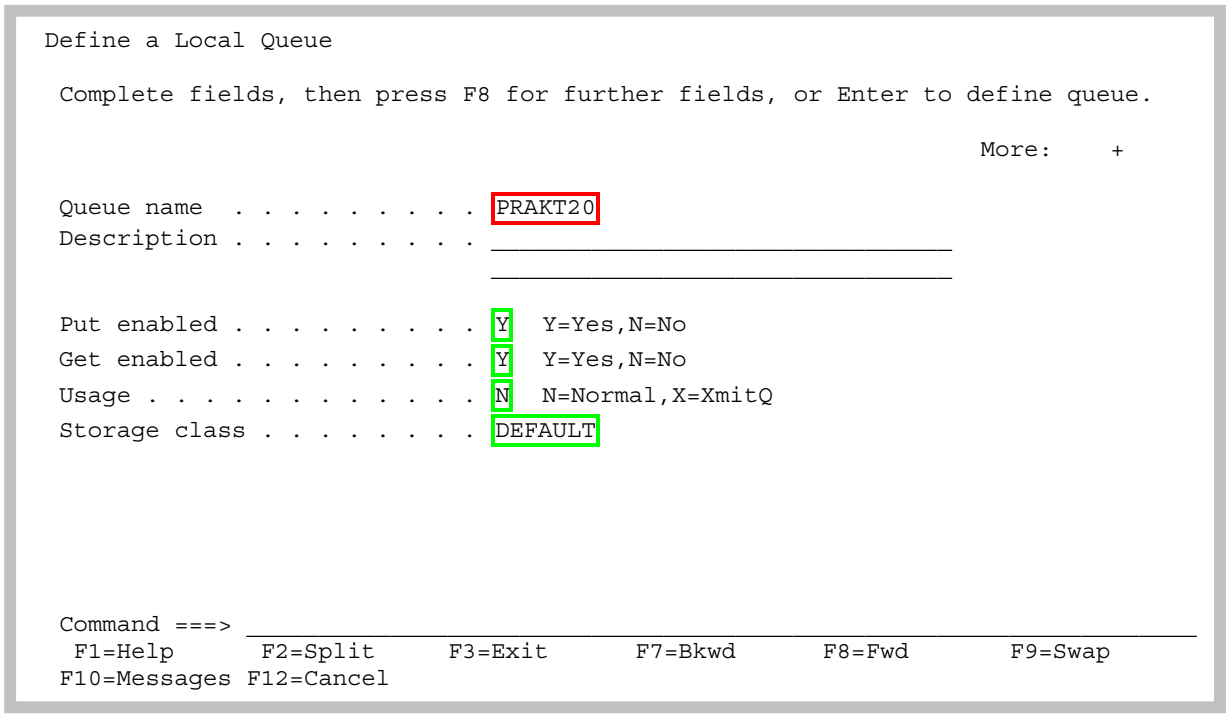


**Abbildung 5: 2. Define Screen**

In diesem Screen muss nochmals der Name der Queue eingetragen werden (PRAKTxx). Für die Freigabe von Put und Get wählen wir jeweils Y (Yes) aus. Die weiteren Parameter werden entsprechend der Abbildung 5 eingegeben.

Anschließend betätigen wir die Steuertaste F8. Das Ergebnis ist ein 3. Define Screen, in dem in der Zeile "Default persistence" "N" (No) eingetragen wird. Eine Nachricht heißt "persistent", wenn sie auch nach einem Neustart des Queue Managers noch vorhanden ist. Dabei ist es gleichgültig, ob der Queue Manager durch einen Bedienerbefehl oder aufgrund eines Systemfehlers gestoppt wurde. Das bedeutet, dass persistente Nachrichten in ein Protokoll geschrieben werden. Wird ein Queue Manager nach einem Fehler erneut gestartet, stellt er diese persistenten Nachrichten aus den protokollierten Daten wieder her.

```
Define a Local Queue

 Press F7 or F8 to see other fields, or Enter to define queue.

                                                                More:  - +

 Default persistence . . . . . N  Y=Yes,N=No
 Default priority  . . . . . . 0  0 - 9
 Message delivery sequence . . P  P=Priority,F=FIFO
 Permit shared access  . . . . N  Y=Yes,N=No
 Default share option  . . . . E  E=Exclusive,S=Shared
 Index type  . . . . . . . . . N  N=None,M=MsgId,C=CorrelId,T=MsgToken
 Maximum queue depth . . . . . 999999999  0 - 999999999
 Maximum message length  . . . 4194304  0 - 4194304
 Retention interval  . . . . . 999999999  0 - 999999999 hours

 Cluster name  . . . . . _____________________________________________
 Cluster namelist name . . . . _____________________________________________
 Default bind  . . . . . . . . O  O=Open,N=Notfixed


 Command ===> ____________________________________________________________
  F1=Help      F2=Split     F3=Exit      F7=Bkwd      F8=Fwd       F9=Swap
 F10=Messages F12=Cancel
```

**Abbildung 6: Eingabe-Parameter im 2. Define Screen**

Eine Nachricht gilt als "nicht-persistent", wenn sie nach einem Neustart des Queue Managers nicht mehr vorhanden ist. Wir entscheiden uns bezüglich unserer lokalen Queue "PRAKTxx" für eine nicht-persistente Nachricht. Die restlichen Parameter im 3. Define Screen werden entsprechend Abbildung 6 übernommen.

```
Define a Local Queue

 Press F7 or F8 to see other fields, or Enter to define queue.

                                                                More:  - +


 Trigger Definition

 Trigger type  . . . . . F  F=First,E=Every,D=Depth,N=None

    Trigger set  . . . . . N  Y=Yes,N=No
    Trigger message priority . 0  0 - 9
    Trigger depth  . . . . . 1          1 - 999999999
    Trigger data . . . . . _____________________________
                           _____________________________

    Process name . . . . . _____________________________________________
    Initiation queue . . . . _____________________________________________


 Command ===> ____________________________________________________________
  F1=Help      F2=Split     F3=Exit      F7=Bkwd      F8=Fwd       F9=Swap
 F10=Messages F12=Cancel
```

**Abbildung 7: 3. Define Screen**

Mittels F8 können weitere drei Define Screens aufgerufen werden. Wir verzichten darauf, indem wir die Eingaben im 3. Define Screen mit dem Betätigen der Enter-Taste abschließen.

Als Resultat dieser Aktion erscheint der Screen "Display messages", der ausgibt, ob die getroffenen Definitionen für unsere lokale Queue "PRAKTxx" erfolgreich waren (Abb. 7)

```
Define a Local Queue

 Complete fields, then press F8 for further fields, or Enter to define queue.

                                                                    More:    +

 Queue name  . . . . . . . . . PRAKT20
 Description . . . . . . . . . ____________________________________
                               ____________________________________

 Put enabled . . . . . . . . . Y  Y=Yes,N=No
 Get enabled . . . . . . . . . Y  Y=Yes,N=No
 Usage . . . . . . . . . . . . N  N=Normal,X=XmitQ
 Storage class . . . . . . . . DEFAULT




 CSQ9022I !MQA1 CSQMMSGP ' DEFINE QLOCAL' NORMAL COMPLETION
 Command ===> ____________________________________________________________
  F1=Help       F2=Split      F3=Exit       F7=Bkwd       F8=Fwd        F9=Swap
 F10=Messages F12=Cancel
```

**Abbildung 8: Display messages**

Mit der Steuertaste F3 kehren wir zurück in das MQSeries Main Menu. In der Zeile "Action" wird eine 1 und in der Zeile "Name" P* eingetragen. Wir betätigen anschließend die Enter-Taste. Damit gelangen wir in den Screen "List Local Queues".

```
IBM MQSeries for OS/390 - Main Menu

 Complete fields. Then press Enter.

  Action . . . . . . . . . 1      1. Display   5. Perform
                                  2. Define    6. Start
                                  3. Alter     7. Stop
                                  4. Delete

  Object type  . . . . . QLOCAL        +
  Name . . . . . . . . . P*
  Like . . . . . . . . . ____________________________________________________

  Connect to queue
  manager  . . . . . . . : MQA1
  Target queue manager : MQA1
  Response wait time . : 30   seconds

 (C) Copyright IBM Corporation 1993,1999. All rights reserved.


 Command ===> ____________________________________________________________
  F1=Help       F2=Split      F3=Exit       F4=Prompt     F6=QueueMgr   F9=Swap
 F10=Messages F12=Cancel
```

**Abbildung 9: MQSeries Main Menu**

Letzterer listet alle lokalen Queues mit ihren Definitionen aus, die mit "P" beginnen. Mit der Steuertaste F11 können wichtige Parameter der lokalen Queues auf der rechten Seite des Screens sichtbar gemacht werden. Das Ergebnis zeigt die Abbildung 10.

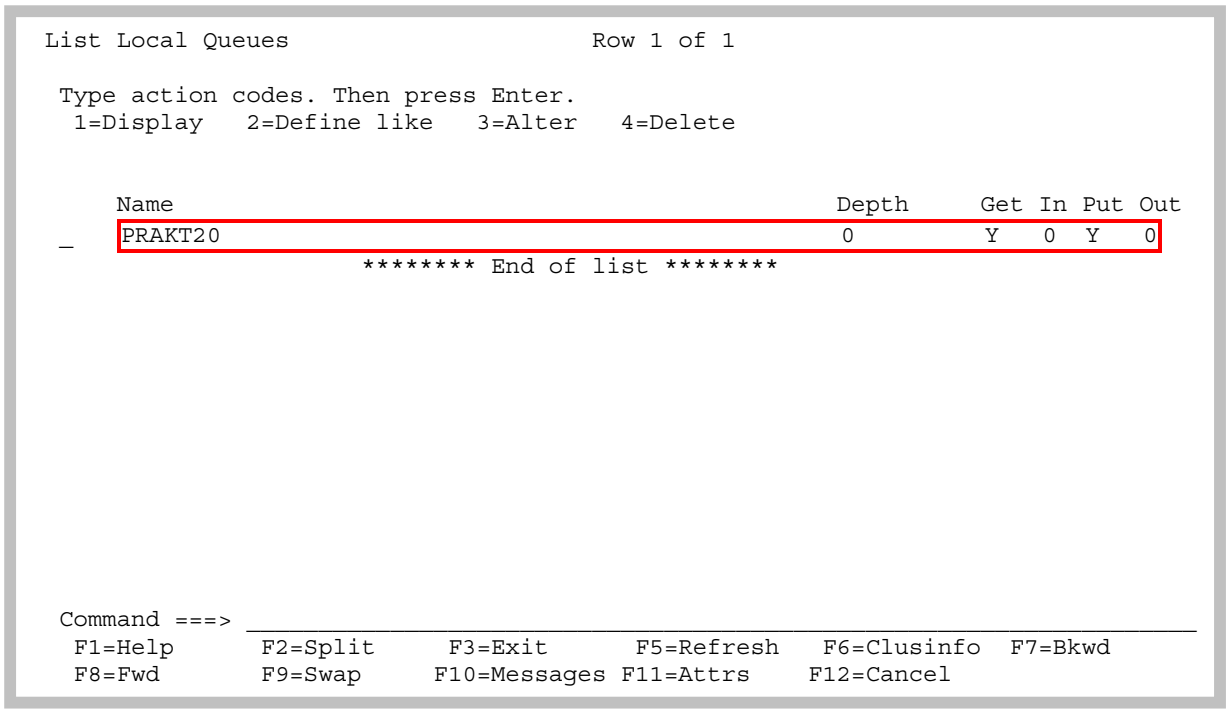


**Abbildung 10: List Local Queues**

*Aufgabe:* *Legen Sie die Local Queue mit dem Namen PRAKTxx (xx für ihre Nummer) an und überprüfen Sie, ob die Queue angelegt wurde.*

Der nächste Schritt beinhaltet die Generierung des Quellcodes für die Anwender-Programme A und B in der Programmiersprache COBOL II.

Da wir nicht voraussetzen, dass der Übungsteilnehmer über COBOL-Kenntnisse verfügt, kopieren wir uns einfach die erforderlichen Quellcodes für die Programme A (PUT) und B (GET) aus der IBM-Bibliothek von MQSeries. Letztere hat den Namen MQM.SCSQCOBS.

Die Bibliotheken liegen schon auf dem Server vorkompiliert vor.
Um eine Übersicht über die beiden COBOL-Programm-Module zu erhalten, gehen wir folgendermaßen vor:
In dem Custompac Master Application Menu wird als Option P.3.4 eingegeben und die Enter-Taste betätigt. Als Ergebnis dieser Aktion erscheint die Data Set List Utility (Abbildung 11)

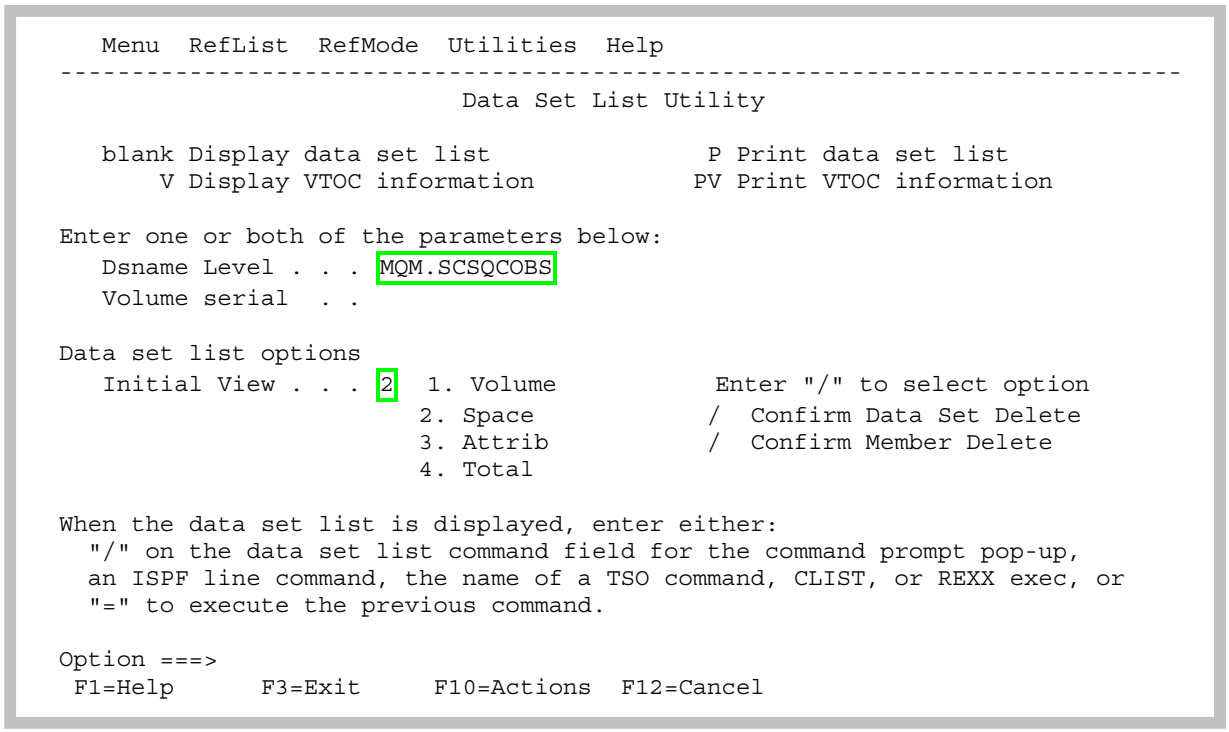


**Abbildung 11: Data Set List Utility**

In diesem Screen wird in der Zeile "Dsname Level" der PDS-Name "MQM.SCSQCOB**S**" eingetragen. Diese Bibliothek zusammen mit MQM.SCSQCOB**C** bilden integrale Bestandteile von MQSeries.
Die erste Bibliothek enthält u.a. die COBOL II-Quellprogramm-Module für PUT und GET (...**S** → **S**ource, während die zweite die Copy Books beinhaltet (...**C** → **C**opy).

In der Zeile Initial View wird eine "2" eingetragen (Space). Anschließend betätigen wir die Enter-Taste.
Der folgende Screen "DSLIST-Data Sets Matching MQM.SCSQCOBS" listet uns den belegten Speicherplatz auf einem 3390-Device aus. (Abbildung 12)

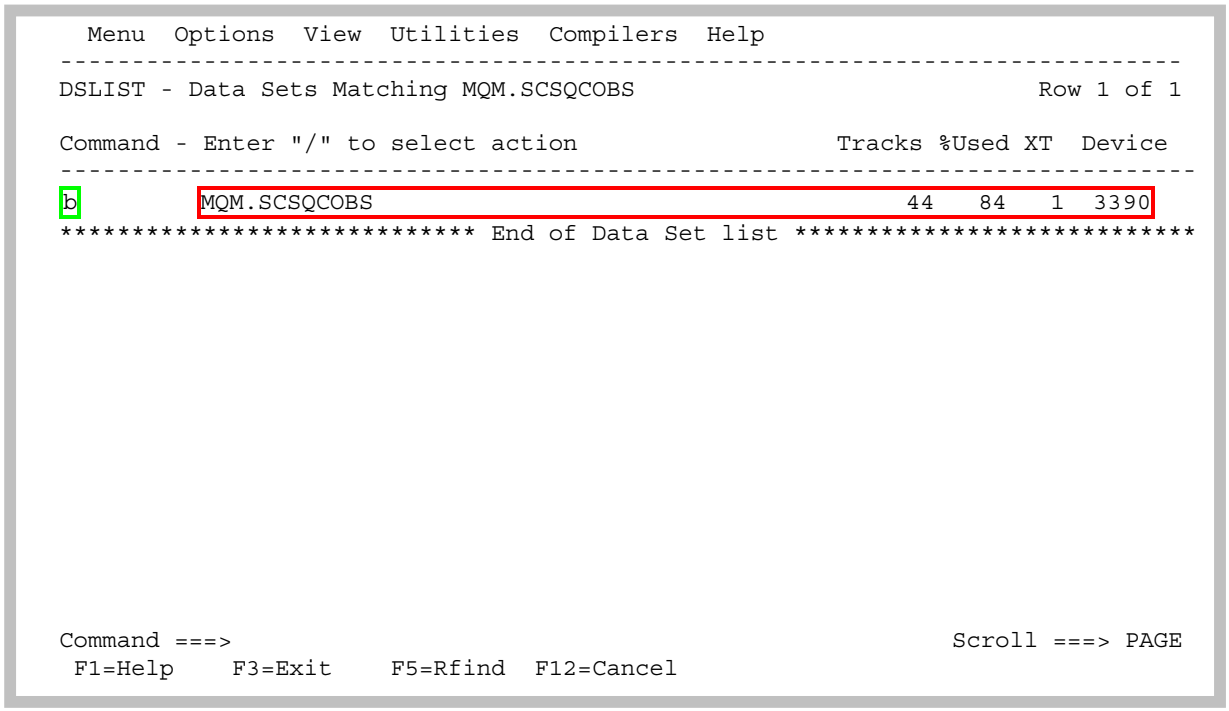


**Abbildung 12: DSLIST**

Wir setzen vor den Dateinamen MQM.SCSQCOBS ein "b" (browse) und betätigen die Enter-Taste. Der Ergebnis-Screen listet uns sämtliche Member dieses PDS aus, diese sind in der Abbildung 11 dargestellt.

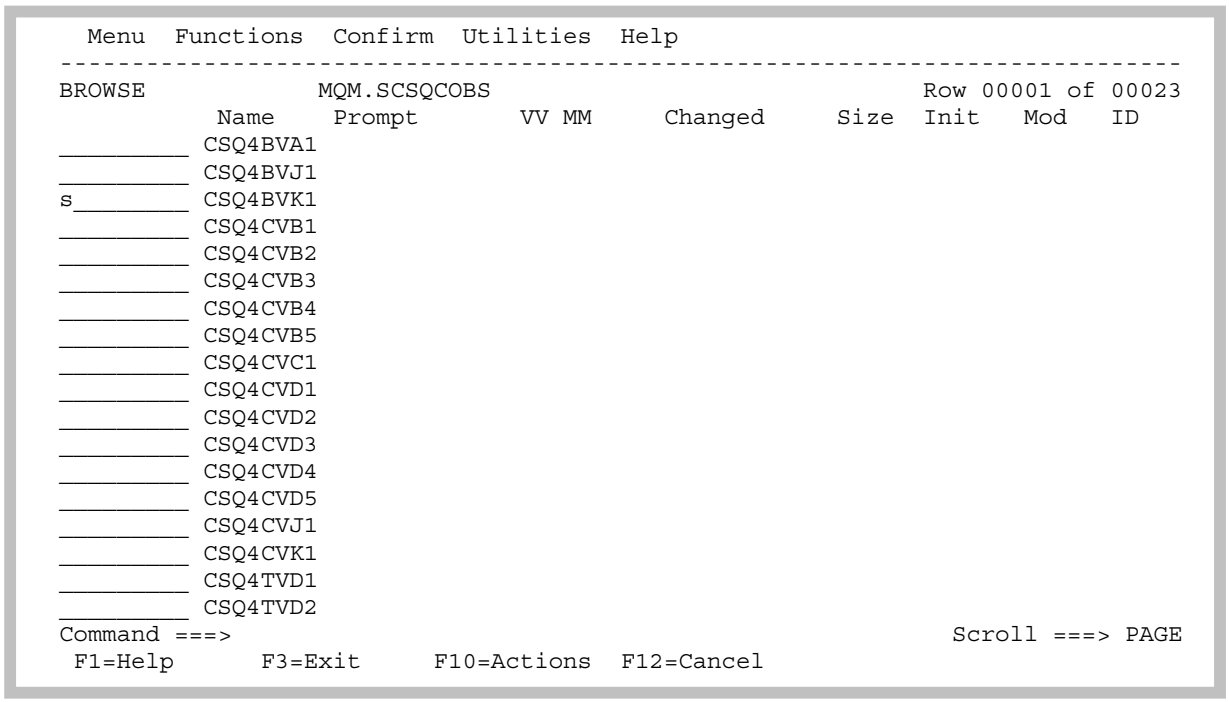


**Abbildung 13: Member des PDS MQM.SCSQCOBS**

Mit dem Kommando "S" (Select) auf der Zeile vor dem Member CSQ4BVK1 kann der Quellcode dieses Modules sichtbar gemacht werden (Abbildung 12). Mit den Steuertasten F8 und F7 kann vor- bzw. rückwärts geblättert werden.

```
   Menu  Utilities  Compilers  Help
 ------------------------------------------------------------------------------
 BROWSE    MQM.SCSQCOBS(CSQ4BVK1)                    Line 00000000 Col 001 080
******************************** Top of Data ********************************
CBL NODYNAM,LIB,OBJECT,RENT,RES,APOST
      *                                                                 *
      * --------------------------------------------------------------- *
       IDENTIFICATION DIVISION.
      * --------------------------------------------------------------- *
       PROGRAM-ID. CSQ4BVK1.
      *REMARKS
      *****************************************************************
      * @START_COPYRIGHT@                                             *
      * Statement:     Licensed Materials - Property of IBM           *
      *                                                               *
      *                5655-A95                                       *
      *                (C) Copyright IBM Corporation. 1993, 1998      *
      *                                                               *
      * Status:        Version 2 Release 1                            *
      * @END_COPYRIGHT@                                              *
      *                                                                *
      *  Module Name      : CSQ4BVK1                                   *
 Command ===>                                            Scroll ===> PAGE
  F1=Help    F3=Exit    F5=Rfind  F12=Cancel
```

**Abbildung 14: 1. Screen des Quellcodes von CSQ4BVK1**

Mit der Steuertaste F3 gelangt man zurück in die Liste der Member von MQM.SCSQCOBS und kann mittels des Select-Kommandos den Quellcode des Members CSQ4BVJ1 auf dem Bildschirm sichtbar machen.

Wir brauchen also nur noch die 2 JCL- Skripte, um eine Nachricht zur Message-Queue MQA1 zu senden und um eine Nachricht zu empfangen.

Bevor wir die beiden Module kopieren, legen wir uns eine private Bibliothek mit dem Namen PRAKTxx.MQA1.USERJCL an. Dazu legen wir wie im 1. Tutorial dieses Dataset an. Bitte dazu die Werte wie im Screen unten dargestellt eingeben.

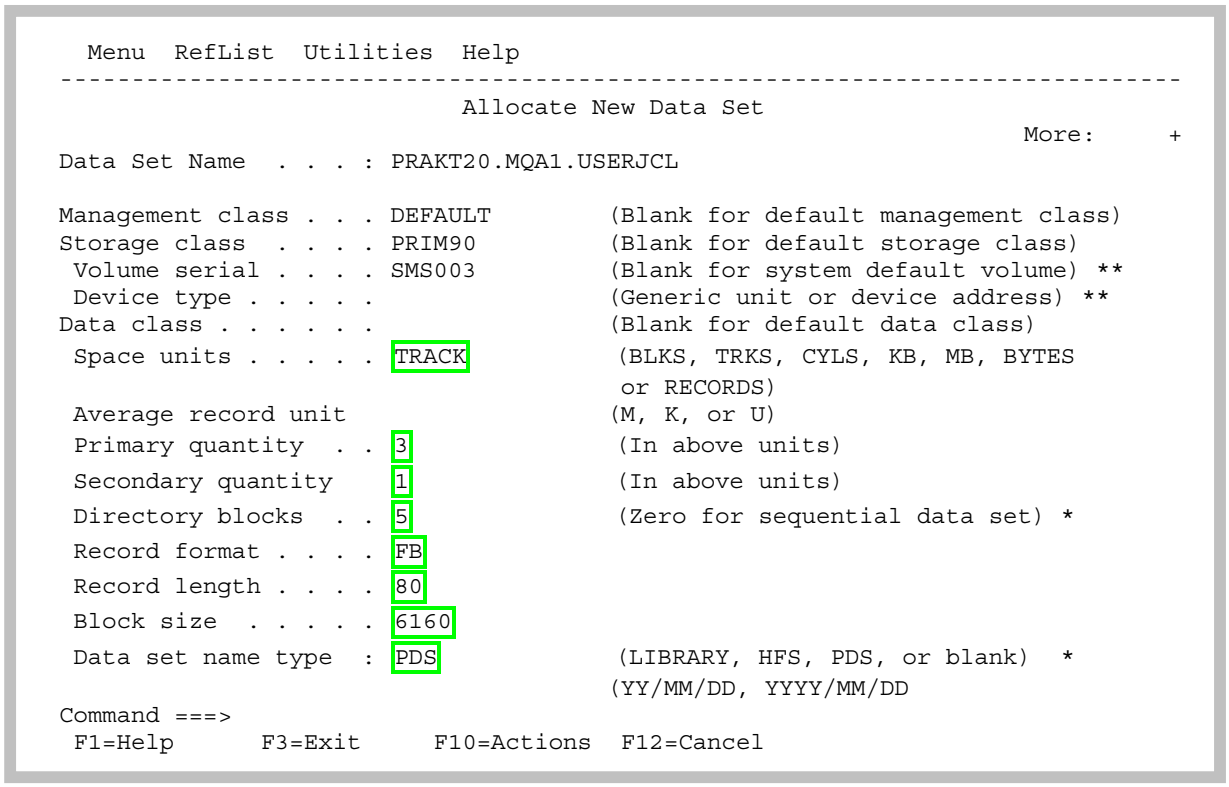


**Abbildung 15: Anlegen eines neuen Data Set**

Dieser PDS soll das entsprechende JCL-Script2 für den PUT- und GET-Job aufnehmen. Wir kopieren uns aus dem PDS MQM.SCSQPROC das Member CSQ4BVJR. Dieses Member sollte zweimal kopiert werden und unter dem Namen MQPUT und MQGET abgelegt werden. Damit verfügen wir über ein JCL-Script mit den Membern MQPUT und MQGET. MQPUT enthält den Job, der durch Aufruf des Moduls CSQ4BVK1 mit den entsprechenden Parametern (QMGR, QUEUE, MSGS, PAD, LEN, PERS) eine Nachricht in eine lokale Queue schreibt und das Ergebnis auf den Bildschirm ausgibt.
MQGET implementiert den Job, der diese Nachricht durch Aufruf des Moduls CSQ4BVJ1 mit identischen Parametern aus der lokalen Queue liest und das Ergebnis ebenfalls auf den Bildschirm legt.

Das Kopieren kann mittels dem Move/Copy Utility erfolgen (Option P.3.3).

```
  Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
                            Move/Copy Utility
                                                                       More:     +
 C  Copy data set or member(s)          CP Copy and print
 M  Move data set or member(s)          MP Move and print
 L  Copy and LMF lock member(s)         LP Copy, LMF lock, and print
 P  LMF Promote data set or member(s)   PP LMF Promote and print

Specify "From" Data Set below, then press Enter key

From ISPF Library:
   Project . . .              (--- Options C, CP, L, and LP only ----)
   Group . . . .          . . .          . . .          . . .
   Type  . . . .
   Member  . . .                 (Blank or pattern for member list,
                                  "*" for all members)

From Other Partitioned or Sequential Data Set:
   Data Set Name . . . 'MQM.SCSQPROC(CSQ4BVJR)'
   Volume Serial . . .           (If not cataloged)

Option ===> C
 F1=Help      F3=Exit     F10=Actions  F12=Cancel
```

**Abbildung 16: Kopieren von CSQ4BVJR**

Wir geben die Werte wie oben dagestellt ein, betätigen die Enter-Taste und gelangen zu folgendem Screen:

```
  Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
COPY     From MQM.SCSQPROC(CSQ4BVJR)
                                                                       More:     +
Specify "To" Data Set Below

To ISPF Library:
   Project  . . PRAKT20          Replace option:
   Group  . . . MQA1                Enter "/" to select option
   Type . . . . USERJCL                Replace like-named members
   Member . . . MQGET      (Blank unless member is to be renamed)

To Other Partitioned or Sequential Data Set:
   Data Set Name . . .
   Volume Serial . . .           (If not cataloged)

Data Set Password  . .           (If password protected)

To Data Set Options:
   Sequential Disposition        Pack Option         SCLM Setting
   1  1. Mod                     3  1. Yes           3  1. SCLM
      2. Old                        2. No              2. Non-SCLM
Command ===>
 F1=Help      F3=Exit     F10=Actions  F12=Cancel
```

**Abbildung 17: Kopieren von CSQ4BVJR nach MQGET**

Wir kopieren den selben Member noch einmal, aber legen ihn nun unter PRAKTxx.MQA1.USERJCL.MQPUT ab.

*Aufgabe: Erstellen Sie den neuen Dataset und kopieren Sie das JCL Skript 2 mal (als MQGET und MQPUT).*

Nun Editieren wir MQPUT und MQGET. Beide Dateien sind JCL-Skripte, die wir für unsere Zwecke anpassen müssen:
Wir öffnen MQPUT mittels des TSO- Editors:

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       PRAKT20.MQA1.USERJCL(MQPUT) - 01.00            Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
==MSG> -CAUTION- Profile is set to STATS ON. Statistics did not exist for
==MSG>           this member, but will be generated if data is saved.
000001 //*********************************************************************
000002 //*                                                                   *
000003 //* @START_COPYRIGHT@                                                 *
000004 //*   Statement:     Licensed Materials - Property of IBM             *
000005 //*                                                                   *
000006 //*                  5655-A95                                         *
000007 //*                  (C) Copyright IBM Corporation. 1993, 1998        *
000008 //*                                                                   *
000009 //*   Status:        Version 2 Release 1                              *
000010 //* @END_COPYRIGHT@                                                   *
000011 //*                                                                   *
000012 //*********************************************************************
000013 //* CUSTOMIZE THIS JCL HERE FOR YOUR INSTALLATION
000014 //* YOU MUST DO GLOBAL CHANGES ON THESE PARAMETERS USING YOUR EDITOR
Command ===>                                                 Scroll ===> PAGE
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

**Abbildung 18: Quelltext MQPUT**

Durch Scrollen mittels F7 bzw. F8 kann der ganz Quelltext sichtbar gemacht werden. Man erkennt leicht, das das Skript sehr gut kommentiert ist. (Das eigentliche Skript kann man auf 8 Zeilen kürzen).

Wir Bewegen uns zu Zeile 56:

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       PRAKT20.MQA1.USERJCL(MQPUT) - 01.00            Columns 00001 00072
000047 //*    - FIRST PARM  (++QMGR++)  QUEUE MANAGER NAME
000048 //*    - SECOND PARM (++QUEUE++) QUEUE NAME
000049 //*    - THIRD PARM  (++MSGS++)  THE NUMBER OF MESSAGES TO GET-(9999)
000050 //*    - FOURTH PARM (++GET++)   GET TYPE-(B)ROWSE/(D)ESTRUCTIVE
000051 //*    - FIFTH PARM  (++SYNC++)  (S)YNCPOINT/(N)O SYNCPOINT
000052 //*  MESSAGES ARE PRINTED TO DD SYSPRINT
000053 //*
000054 //*********************************************************************
000055 //PUTMSGS  EXEC PGM=CSQ4BVK1,REGION=1024K,
000056 //  PARM=('++QMGR++,++QUEUE++,++MSGS++,++PAD++,++LEN++,++PERS++')
000057 //*GETMSGS  EXEC PGM=CSQ4BVJ1,REGION=1024K,
000058 //*  PARM=('++QMGR++,++QUEUE++,++MSGS++,++GET++,++SYNC++')
000059 //STEPLIB  DD   DSN=++USERLIB++,DISP=SHR
000060 //         DD   DSN=++THLQUAL++.SCSQANL++LANGLETTER++,DISP=SHR
000061 //         DD   DSN=++THLQUAL++.SCSQAUTH,DISP=SHR
000062 //SYSDBOUT DD   SYSOUT=*
000063 //SYSABOUT DD   SYSOUT=*
000064 //SYSPRINT DD   SYSOUT=*
000065 //SYSOUT   DD   SYSOUT=*
Command ===>                                                Scroll ===> PAGE
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

**Abbildung 19: 2. Screen Quelltext MQPUT**

Die Parameter ab Zeile 56 sind dabei im Skript sehr gut dokumentiert.
Wir ersetzen diese Zeile durch:

```
  File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       PRAKT20.MQA1.USERJCL(MQPUT) - 01.00            Columns 00001 00072
000053 //*
000054 //*********************************************************************
000055 //PUTMSGS  EXEC PGM=CSQ4BVK1,REGION=1024K,
000056 //  PARM=('MQA1,PRAKT20,3,Z,5,N')
000057 //*PUTMSGS  EXEC PGM=CSQ4BVK1,REGION=1024K,
000058 //*  PARM=('++QMGR++,++QUEUE++,++MSGS++,++PAD++,++LEN++,++PERS++')
000059 //STEPLIB  DD   DSN=MQM.MQA1.USERLIB,DISP=SHR
000060 //         DD   DSN=MQM.SCSQANLE,DISP=SHR
000061 //         DD   DSN=MQM.SCSQAUTH,DISP=SHR
000062 //SYSDBOUT DD   SYSOUT=*
000063 //SYSABOUT DD   SYSOUT=*
000064 //SYSPRINT DD   SYSOUT=*
****** **************************** Bottom of Data ****************************




Command ===> sub                                            Scroll ===> PAGE
 F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

**Abbildung 20: Neuer Quelltext MQPUT**

Das bedeutet: Es werden 3 Messages in die lokale Queue geschrieben, wobei jede Message aus einer Sequenz "ZZZZZ" besteht. Nach dem Editieren der Parameter erfolgt die Eingabe von "SUB" auf der Command-Zeile (s. Abbildung ...).

Wird das Kommando sub mittels Enter bestätigt, erscheint folgender screen:

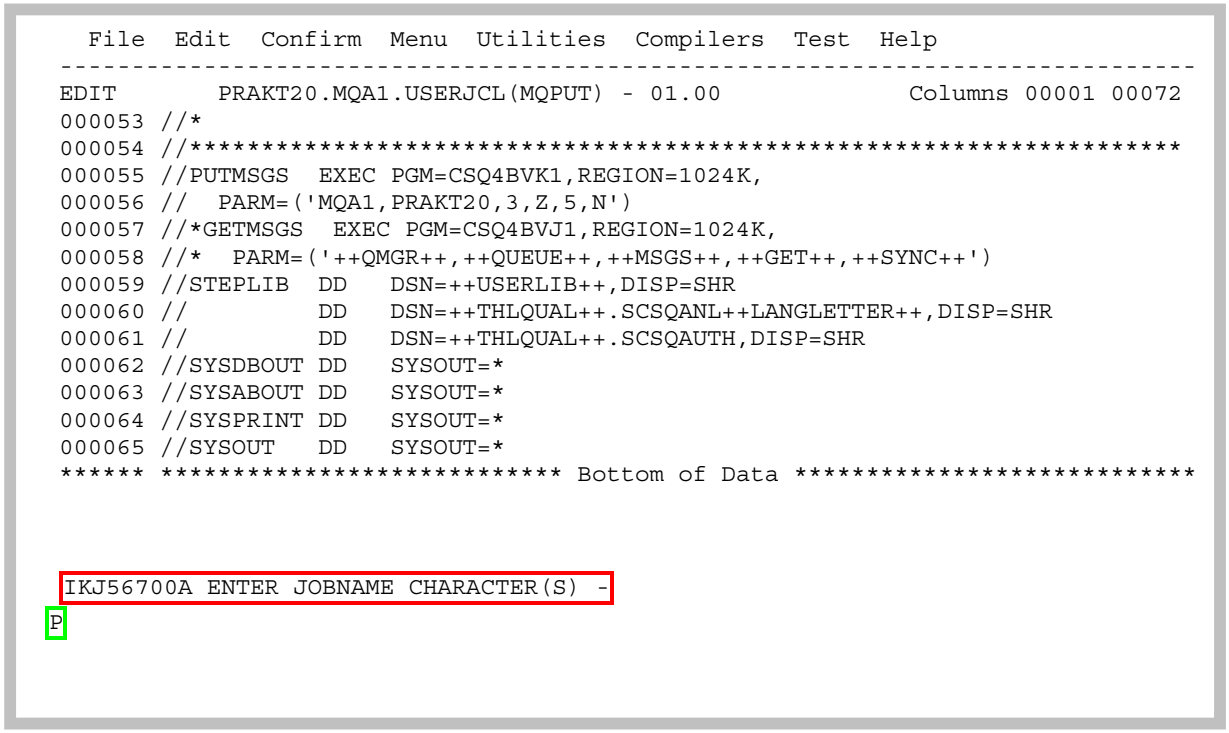


**Abbildung 21: „sub MQPUT“**

Hier geben wir einfach ein P (für Put) ein.
Hintergrund: Der Job bekommt einen Namen, der sich aus dem Login (PRAKTxx) und den oben eingegebnen Zeichen zusammensetzt. (mit maximal 8 Zeichen). Damit heißt unser Job PRAKTxxP

Wir bekommen nun folgende Meldung:

```
   File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
 ------------------------------------------------------------------------------
 EDIT       PRAKT20.MQA1.USERJCL(MQPUT) - 01.00            Columns 00001 00072
 000055 //PUTMSGS  EXEC PGM=CSQ4BVK1,REGION=1024K,
 000056 //  PARM=('MQA1,PRAKT20,3,Z,5,N')
 000057 //*GETMSGS  EXEC PGM=CSQ4BVJ1,REGION=1024K,
 000058 //*  PARM=('++QMGR++,++QUEUE++,++MSGS++,++GET++,++SYNC++')
 000059 //STEPLIB  DD   DSN=++USERLIB++,DISP=SHR
 000060 //         DD   DSN=++THLQUAL++.SCSQANL++LANGLETTER++,DISP=SHR
 000061 //         DD   DSN=++THLQUAL++.SCSQAUTH,DISP=SHR
 000062 //SYSDBOUT DD   SYSOUT=*
 000063 //SYSABOUT DD   SYSOUT=*
 000064 //SYSPRINT DD   SYSOUT=*
 000065 //SYSOUT   DD   SYSOUT=*
 ****** **************************** Bottom of Data ****************************




 IKJ56700A ENTER JOBNAME CHARACTER(S) -
P
 IKJ56250I JOB PRAKT20P(JOB00829) SUBMITTED
 ***
```

**Abbildung 22: 2. Screen „sub MQPUT“**

Wir drücken Enter, bis wir eine Meldung auf dem Bildschirm erhalten:

```
11.51.22 JOB00833 $HASP165 PRAKT20P ENDED AT N1  MAXCC=0 CN(INTERNAL)
 ***
```

**Abbildung 23: Ausgabe von „sub MQPUT“**

Damit haben wir unsere Nachrichten in die lokale Queue geschrieben.

Nun müssen diese aber noch ausgelesen werden. Dazu editieren wir das Member MQGET aus PRAKTxx.MQA1.USERJCL.

Es ist das gleiche Skript wie MQGET, wir Scrollen mit F8 gleich wieder zu Zeile 56 und ersetzen wieder die Zeilen ab 56:

```
   File  Edit  Confirm  Menu  Utilities  Compilers  Test  Help
 ------------------------------------------------------------------------------
 EDIT       PRAKT20.MQA1.USERJCL(MQGET) - 01.00            Columns 00001 00072
 000053 //*
 000054 //*******************************************************************
 000055 //*PUTMSGS  EXEC PGM=CSQ4BVK1,REGION=1024K,
 000056 //*  PARM=('++QMGR++,++QUEUE++,++MSGS++,++PAD++,++LEN++,++PERS++')
 000057 //GETMSGS  EXEC PGM=CSQ4BVJ1,REGION=1024K,
 000058 //  PARM=('MQA1,PRAKT20,3,D,N')
 000059 //STEPLIB  DD   DSN=MQM.MQA1.USERLIB,DISP=SHR
 000060 //         DD   DSN=MQM.SCSQANLE,DISP=SHR
 000061 //         DD   DSN=MQM.SCSQAUTH,DISP=SHR
 000062 //SYSDBOUT DD   SYSOUT=*
 000063 //SYSABOUT DD   SYSOUT=*
 000064 //SYSPRINT DD   SYSOUT=*
 000065 //SYSOUT   DD   SYSOUT=*
 ****** ***************************** Bottom of Data ****************************



 Command ===> sub                                            Scroll ===>
PAGE
  F1=Help      F3=Exit      F5=Rfind     F6=Rchange  F12=Cancel
```

**Abbildung 24: Neuer Quelltext von MQGET**

Durch Eingabe von „sub“ wir das Skript wieder ausgeführt, der Jobname soll nun PRAKTxxG sein, also G eingeben.

*Aufgabe:* *Führen Sie die Skripte MQPUT und danach MQGET je einmal aus.*

Nach erfolgreichem Durchlauf kann man sich fragen, was denn nun eigentlich passiert ist:

Wir kommen mit wiederholtem Betätigen der Steuertaste F3 in das Custompac Master Application Menu. Anschließend geben wir auf der Zeile "Option" SD ein und gelangen damit in das SDSF.

```
  Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
HQX1900-----------------  SDSF PRIMARY OPTION MENU  -------------------------

    DA        - Display active users in the sysplex
    I         - Display jobs in the JES2 input queue
    O         - Display jobs in the JES2 output queue
    H         - Display jobs in the JES2 held output queue
    ST        - Display status of jobs in the JES2 queues
    SE        - Display scheduling environments in the MAS or sysplex
    END       - Exit SDSF

    Licensed Materials - Property of IBM

    5647-A01 (C) Copyright IBM Corp. 1981, 1997. All rights reserved.
    US Government Users Restricted Rights - Use, duplication or
    disclosure restricted by GSA ADP Schedule Contract with IBM Corp.



COMMAND INPUT ===> st                                        SCROLL ===> PAGE
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

**Abbildung 25: SDSF Primary Option Menu**

Dort erfolgt auf der Command-Zeile die Eingabe “ST“ (Status). Enter

```
   Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF STATUS DISPLAY ALL CLASSES                        LINE 1-9 (9)
NP   JOBNAME  JOBID    OWNER    PRTY QUEUE      C  POS  SAFF  ASYS STATUS
     PRAKT20  TSU00826 PRAKT20    15 EXECUTION           SYS1  SYS1
S    PRAKT20P JOB00837 PRAKT20     1 PRINT      A  1106
     PRAKT20G JOB00838 PRAKT20     1 PRINT      A  1107









COMMAND INPUT ===>                                           SCROLL ===> PAGE
 F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
 F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

**Abbildung 26: SDSF Status Display All Classes**

Auf der Zeile der Job-Nummer geben wir ein S (Select) ein und bestätigen mit Enter.
Nun erfolgt die Ausgabe unseres Skriptes, wir scrollen bis ans Ende:

```
   Display  Filter  View  Print  Options  Help
 -------------------------------------------------------------------------------
 SDSF OUTPUT DISPLAY PRAKT20P JOB00839  DSID     4 LINE 20       COLUMNS 02- 81
 COMMAND INPUT ===>                                            SCROLL ===> PAGE
IEF374I STEP/PUTMSGS /STOP  2002093.1214 CPU    0MIN 00.15SEC SRB    0MIN 00.01S
IEF375I  JOB/PRAKT20P/START 2002093.1214
IEF376I  JOB/PRAKT20P/STOP  2002093.1214 CPU    0MIN 00.15SEC SRB    0MIN 00.01S
=========================================
PARAMETERS PASSED :
   QMGR        - MQA1
   QNAME       - PRAKT20
   NUMMSGS     - 000000003
   PADCHAR     - Z
   MSGLENGTH   - 000000005
   PERSISTENCE - N
=========================================
MQCONN SUCCESSFUL
MQOPEN SUCCESSFUL
000000003 MESSAGES PUT TO QUEUE
MQCLOSE SUCCESSFUL
MQDISC SUCCESSFUL
****************************** BOTTOM OF DATA ********************************
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

**Abbildung 27: SDSF Output Display Job von MQPUT**

Hier sehen wir, was das Skript geleistet hat, ein MQCONN, MQOPEN, 3 Nachriten abgelegt, MQCLOSE und MQDISC.
Wir gehen zurück mit F3 und betrachten uns die Ausgabe von PRAKTxxG.

```
   Display  Filter  View  Print  Options  Help
 -------------------------------------------------------------------------------
 SDSF OUTPUT DISPLAY PRAKT20G JOB00838  DSID     4 LINE 24       COLUMNS 02- 81
 COMMAND INPUT ===>                                            SCROLL ===> PAGE
IEF376I  JOB/PRAKT20G/STOP  2002093.1204 CPU    0MIN 00.15SEC SRB    0MIN 00.01S
=========================================
PARAMETERS PASSED :
   QMGR        - MQA1
   QNAME       - PRAKT20
   NUMMSGS     - 000000003
   GET         - D
   SYNCPOINT   - N
=========================================
MQCONN SUCCESSFUL
MQOPEN SUCCESSFUL
000000000 : 000000005 : ZZZZZ
000000001 : 000000005 : ZZZZZ
000000002 : 000000005 : ZZZZZ
000000003 MESSAGES GOT FROM QUEUE
MQCLOSE SUCCESSFUL
MQDISC SUCCESSFUL
****************************** BOTTOM OF DATA ********************************
  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=IFIND     F6=BOOK
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

**Abbildung 28: SDSF Output Display Job von MQGET**

Hier wurden unsere 3 Nachrichten  wieder ausgelesen.

*Aufgabe:* *Bearbeiten Sie die vorher gegebeben Aufgaben und schicken Sie die Print-Screens 27 und 28 im Bitmap- oder JPEG-Format (pro Bild maximal 250 KByte) an die Mailadresse Ihres Betreuers.*