Tutorial MQ 01
Einführung in WebSphere MQ

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 "PRAK222" erstellt. In allen Dateinamen müssen Sie "PRAK222" durch ihre eigene Benutzer-ID ersetzen. Außerdem wird oft in den Erklärungen PRAKxxx verwendet, wobei die xxx ihre Praktikumsnummer sein soll.

Bei Anweisungen, die sich auf Eingaben beziehen, sind die Anführungszeichen nicht mit einzugeben.

Übersicht

Aufgabe: ***Arbeiten Sie nachfolgendes Tutorial durch.***

1. WebSphere MQ

WebSphere MQ, auf das auch heute noch oft unter der alten Bezeichnung MQSeries verwiesen wird, zählt zur sogenannten nachrichtenbasierten Middleware (Message Oriented Middleware; MOM). Bei nachrichtenbasierter Middleware werden, wie der Name schon sagt, Nachrichten zwischen den Systemen ausgetauscht.

WebSphere MQ nutzt eine spezielle Art der Message Oriented Middleware, das Message Queuing. Das Prinzip des Message Queuing führt Warteschlangen, die Queues, ein.
In diesen Warteschlangen werden zum Beispiel ausgehende Nachrichten solange zwischengespeichert, bis der Empfänger sie anfordert und in seine Warteschlange einreiht.

Diese Technik ermöglicht eine asynchrone Kommunikation zwischen den Teilnehmern. Asynchrone Kommunikation bedeutet in dem Fall, dass, falls der Empfänger zu einem Zeitpunkt nicht zur Verfügung stehen sollte, der Sender so lange wartet, also die Nachricht in seiner Queue lässt, bis der Empfänger empfangsbereit ist. Wobei „warten“ in diesem Fall nicht bedeutet, dass der Sender währenddessen blockiert ist. Es können durchaus noch andere Aufgaben ausgeführt werden.

An dieser Stelle sollte jedoch darauf hingewiesen werden, dass mit „Sender“ und „Empfänger“ hier nicht die jeweiligen Anwendungsprogramme, sondern vielmehr die MQ-Elemente gemeint sind. So kann es zum Beispiel auch der Fall sein, dass die sendende Anwendung nicht mehr aktiv ist, aber trotzdem noch zu sendende Nachrichten in der Queue liegen. Auch in diesem Fall werden die Nachrichten sobald wie möglich versendet.

Wie hat man sich Nachrichten (Messages) überhaupt vorzustellen? Grundsätzlich besteht eine Nachricht aus zwei Teilen, dem Header (Message Descriptor) und dem eigentlichen Datenteil. Der Message Descriptor (bei WebSphere MQ als Message queuing message descriptor (MQMD) bezeichnet), enthält Informationen zu der Nachricht bezüglich des Nachrichtentyps, der Lebensdauer der Nachricht, der zuständigen Antwortwarteschlange (Reply Queue), u.a.
Im Datenteil befinden sich die entsprechenden Anwendungsdaten, welche auch Programmbefehle enthalten können. Die voreingestellte Maximalgröße einer Nachricht beträgt 4 MB, bei WebSphere MQ v6 ist eine Nachrichtengröße zwischen 0 und 100 MB möglich.

Diese Aufgabe wird von einem Queue Manager übernommen. Dem Queue Manager obliegt ebenfalls die Aufgabe, die Queues, Channels und die übrigen MQ-Objekte zu verwalten. Er ist also zwingend notwendig, um mit WebSphere MQ arbeiten zu können.

Innerhalb des Queue Managers übernehmen die Message Channel Agents (MCA) die Aufgabe, Nachrichten aus den Queues zu lesen und über die Kanäle an den Gegenpart zu senden, oder umgekehrt, über die Kanäle empfangene Nachrichten in den entsprechenden Queues abzulegen.

Die Channels (Kanäle) dienen als Verbindung zwischen den Queue Managern. Jedem Kanal ist ein eigener Message Channel Agent zugewiesen.
Ein Kanal ist hierbei eine unidirektionale Verbindung, d.h. also über denselben Kanal kann nicht gesendet und empfangen werden.

Der allgemeine Zweck, den Middleware erfüllen soll, nämlich eine plattformenabhängige Anwendungsentwicklung zu ermöglichen, erfüllt WebSphere MQ durch eine einheitliche Programmierschnittstelle (Application Programming Interface, API). Bei WebSphere MQ wird diese Schnittstelle als Message Queue Interface (MQI) bezeichnet.

Wir unterstellen, dass sie mit den Einzelheiten vertraut sind, siehe Band 1, Abschnitt 10.1.11 ff.

2. Tutorial Übersicht

In diesem Tutorial benutzen wir die MQI Interface. (Die JMS Alternative werden wir in Tutorial WebSphere und Message Diven Beans behandeln). Wir werden (indirekt) die folgenden sechs MQ-Funktionen (auch Calls genannt) benutzen. Einzelheiten in Band 1, Abschnitt 10.4.1 ff. .

MQCONN	**Stellt eine Verbindung zum Queue Manager her (Connect)**
MQDISC	**Löst die Verbindung zum Queue Manager (Disconnect**
MQOPEN	**Öffnet eine Verbindung zu einer bestimmten Queue**
MQCLOSE	**Schließt die Verbindung zu einer bestimmten Queue**
MQPUT	**Legt eine Nachricht in einer Queue ab**
MQGET	**Liest und löscht eine Nachricht aus einer Queue**

WebSphere MQ stellt mit der Installation einige Beispiele bereit, mit denen wir hier arbeiten wollen. Es handelt sich hierbei im Speziellen um die COBOL II-Beispiele zu MQPUT und MQGET.

Wir werden hierzu eine lokale Queue auf dem Queue Manager des z/Os Rechnerst-Maschine anlegen und in diese mit den zur Verfügung gestellten Programmen Nachrichten ablegen und wieder auslesen. COBOL-Kenntnisse sind hierfür nicht erforderlich, Änderungen, also die Übergabe von Parametern für die MQ-Calls, werden in einem, ebenfalls standardmäßig zur Verfügung stehenden, JCL-Skript gemacht.

Wir verwenden hier den Queue Manager CSQ6 auf hobbit (134.2.205.54).
Sollte der Queue Manager nicht existieren, sollte umgehend der Betreuer des Praktikums informiert werden.

Ein Queue Manager kann nur von entsprechend autorisiertem Administrator gestartet werden.

Zur Bearbeitung dieses Tutorials ist lediglich ein TN3270-Emulator erforderlich (z.B. x3270 für Linux-Systeme oder wc3270 für Windows-Systeme).

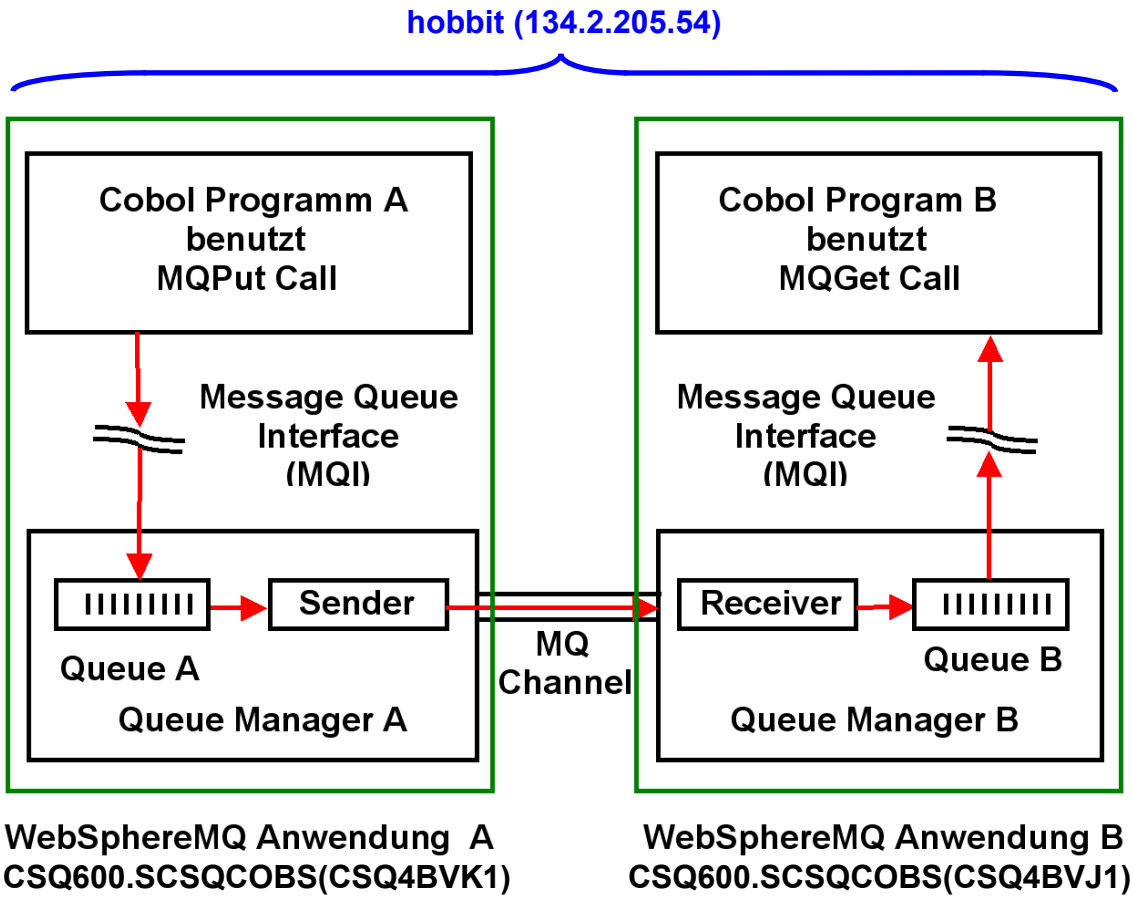

Abb. 6.1

Das von uns erstellte Tutorial erstellt zwei z/OS Anwendungen (Anwendung A und Anwendung B) mit jeweils einem Queue Manager (Queue Manager A) und (Queue Manager B) auf dem gleichen z/OS System.
Für Anwendung A erstellen wir eine Queue A (Transmission Queue, hier als Local Queue bezeichnet). Für Anwendung B erstellen wir eine Queue B (Target Queue).

Sender (MCA), Receiver (MCA und Channel werden durch eine lokale z/OS Struktur implementiert.

Wir speichern in der Transmission Queue eine einfache Nachricht bestehend aus je fünfmal dem Zeichen „Z“ (also dem String „ZZZZZ“).

Wir benutzen ein vorgefertigtes Cobol Programm A, um den Inhalt der Transmission Queue asynchron an die Target Queue zu senden. Wir benutzen ein weiteres vorgefertigtes Cobol Programm B, um den Inhalt der Target Queue auszulesen.

Programm A liegt vorgefertig als Member eines Partitioned Data Sets vor, spezifisch als CSQ600.SCSQCOBS(CSQ4BVK1). Wir können es von dort kopieren. Ebeso liegt Programm B vorgefertigt als CSQ600.SCSQCOBS(CSQ4BVJ1) vor.

Wir legen einen eigenen Partitioned Data Set PRAK222.CSQ6.USERJCL (PRAK 222 durch Ihre eigene User ID ersetzen) an. Wir kopieren Programm A in den Member MQPUT und Programm B- in den Member MQGET.

Wir erstellen ein JCL Script, welches das Programm PRAK222.CSQ6.USERJCL(MQPUT) mittels submit ausführt. Wir benutzen SDSF, um zu verifizieren, dass dies korrekt geschehen ist. Mit einem weiteren JCL Script führen wir das Programm PRAK222.CSQ6.USERJCL(MQGET) mittels submit aus. Wiederum mittels SDSF verifizieren wir, dass unsere Nachricht „ZZZZZ" korrekt eingetroffen ist.

Noch eine Anmerkung zum Schluss: zum Teil zeigen die Abbildungen in diesem Tutorial nur einen Ausschnitt des Bildschirms, davon sollten wir uns jedoch nicht verwirren lassen.

3. WebSphere MQ for z/OS

Zunächst müssen wir eine lokale Warteschlange (local queue) auf dem Server erstellen. Hierfür loggen wir uns auf dem Server (wir benutzen Time-Sharing Option, also „L TSO" oder einfach „LOGON PRAKxxx") ein und rufen das ISPF auf. Hierzu geben wir einfach nach erfolgreichem Login „ispf" ein und bestätigen mit der Eingabetaste.

Es erscheint nun das ISPF Primary Option Menu. Hier werden jedoch nur Basisfunktionen angeboten, wir benötigen für dieses Tutorial jedoch zusätzliche Optionen. Wir geben also in der Eingabezeile „m" für More (im Menü steht hierzu „Additional IBM Products") ein und bestätigen mit der Eingabetaste.

Wir sehen nun das IBM Products Panel (siehe Abbildung 2). Hier finden wir unter Punkt 15 „WMQ Series Operations and Control", genau das also, was wir jetzt benötigen.

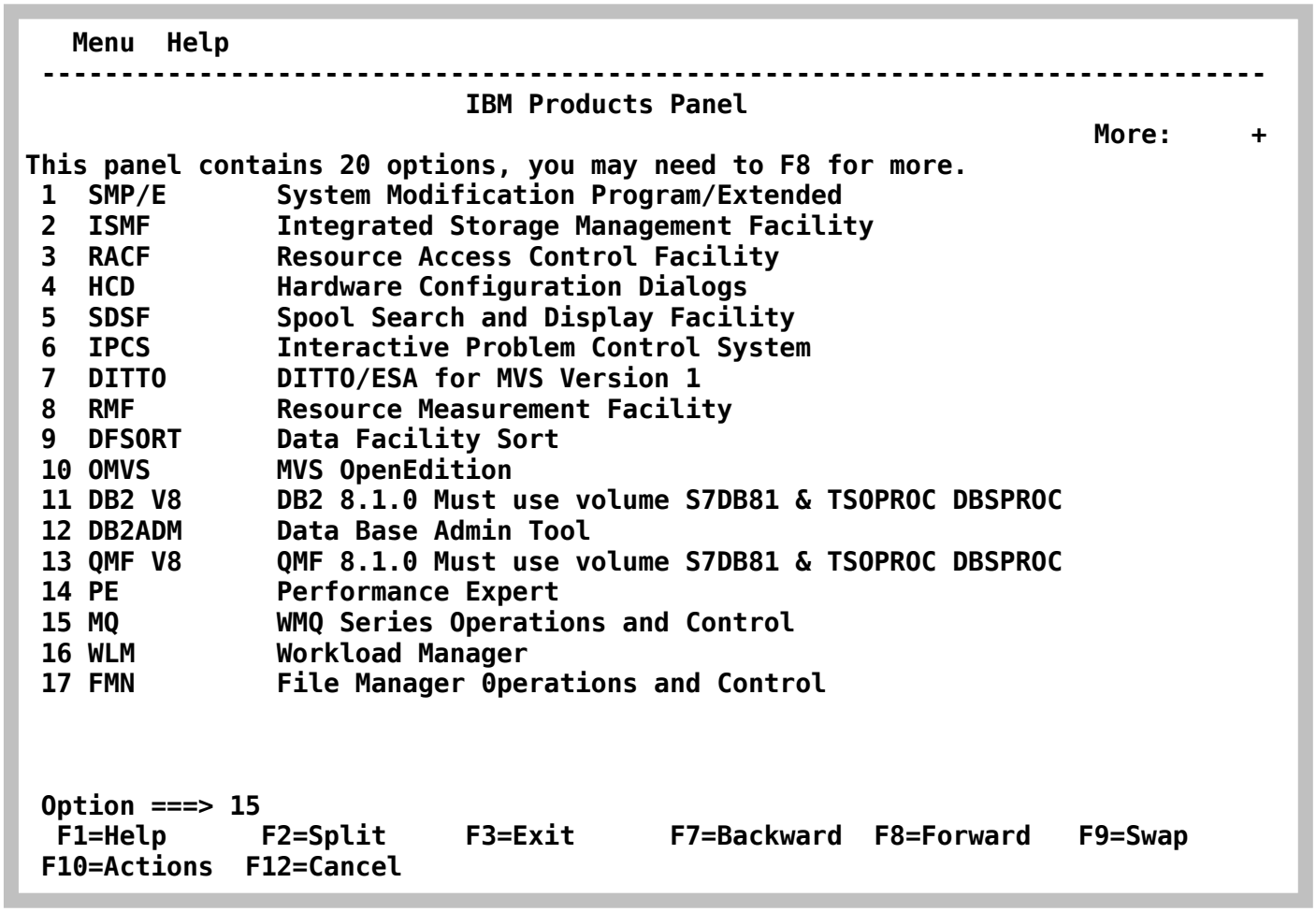

Abbildung 2: Bildschirm des IBM Products Panel

Wir geben also in der Optionszeile „15“ ein und bestätigen wieder mit der Eingabetaste.

Nun befinden wir uns im IBM WebSphere MQ for z/OS – Main Menu.

Zunächst sollte überprüft werden, ob bereits eine lokale Queue mit unserer Praktikums-ID existiert. Hierfür geben wir die Daten wie in Abbildung 3 ein. Da wir zunächst nach einer lokalen Warteschlange suchen wollen, geben wir als „Action“ 1 ein. Als „Object“ wollen wir lokale Queues anzeigen lassen, geben also QLOCAL ein. Andere mögliche Parameter wären zum Beispiel QREMOTE, SENDER oder CHANNEL. In einem anderen Tutorial werden wir einigen noch einmal begegnen. Als „Connect Name“ geben wir unseren Queue Manager CSQ6 ein und bestätigen mit der Eingabetaste

```
                    IBM WebSphere MQ for z/OS - Main Menu

Complete fields. Then press Enter.

Action  . . . . . . . . . . . 1     0. List with filter   4. Manage
                                    1. List or Display    5. Perform
                                    2. Define like        6. Start
                                    3. Alter              7. Stop
Object type . . . . . . . . . QLOCAL        +
Name  . . . . . . . . . . . . PRAK222
Disposition . . . . . . . . .    Q=Qmgr, C=Copy, P=Private, G=Group,
                                 S=Shared, A=All

Connect name  . . . . . . . . CSQ6  - local queue manager or group
Target queue manager  . . .
           - connected or remote queue manager for command input
Action queue manager  . . .        - command scope in group
Response wait time  . . . . 30    5 - 999 seconds

(C) Copyright IBM Corporation 1993,2005. All rights reserved.




Command ===>
 F1=Help      F2=Split     F3=Exit      F4=Prompt    F9=SwapNext F10=Messages
F12=Cancel
```

Abbildung 3: IBM WebSphere MQ for z/OS Hauptmenü

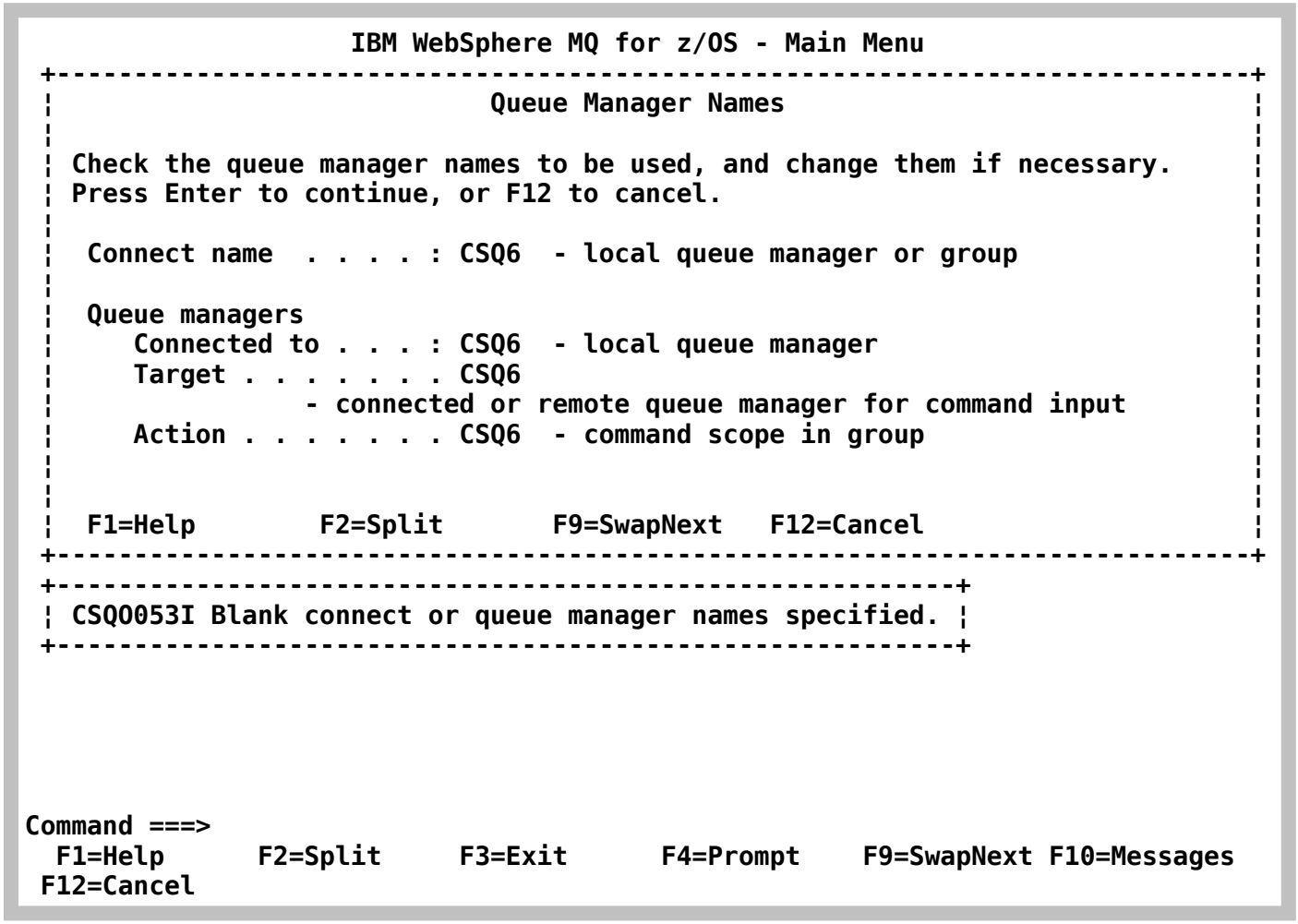

Abbildung 4: Bildschirm: Queue Manager Names

Da wir auf dem letzten Bildschirm (Abbildung 3) die Angaben bei „Target queue manager“ und „Action queue manager“ ausgelassen haben, gelangen wir mit der Meldung „`CSQO053I Blank connect or queue manager names specified.`“ auf den Bildschirm „Queue Manager Names“ (Abbildung 4). Hier geben wir überall den Namen unseres Queue Managers ein.

Wenn wir jetzt wieder im WebSphere MQ Hauptmenü landen und die Meldung „`CSQO050I No objects of type QLOCAL disposition ALL match PRAKxxx.`“ ausgegeben wird, bedeutet dies, dass noch keine Queue mit unserem Namen angelegt wurde. In diesem Fall können wir den folgenden Abschnitt „Löschen einer Local Queue“ überspringen und mit „Anlegen einer Local Queue“ fortfahren.

4. Löschen einer Local Queue

Wenn statt der obigen Meldung folgender Bildschirm (Abbildung 5) erscheint, bedeutet dies, dass bereits eine Queue mit unserem Namen existiert, die wir zunächst löschen müssen.

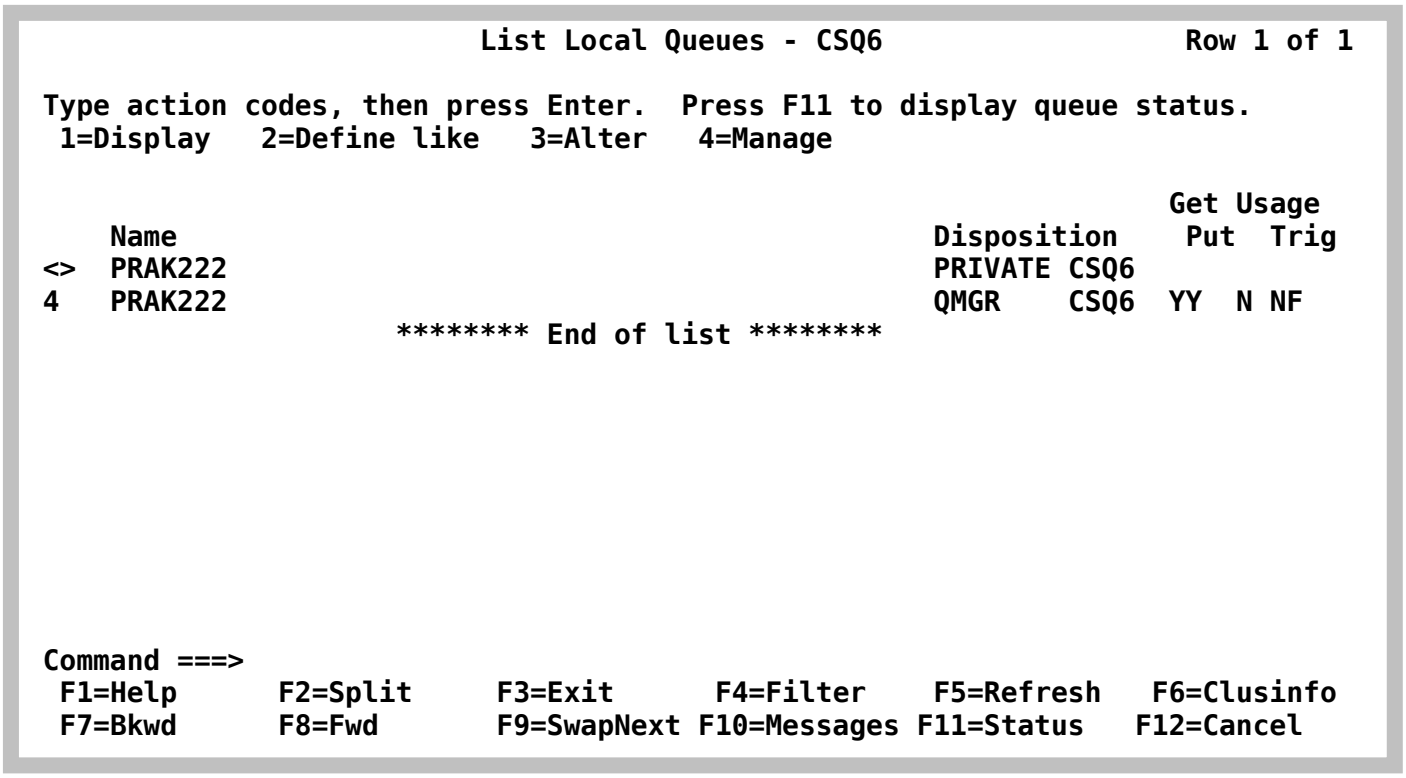

Abbildung 5: Bildschirm: Queue Manager Names

Was wir vor uns sehen, ist eine Liste unserer lokalen Warteschlangen, die wir eigentlich noch gar nicht haben sollten. Wir setzen daher an der Position vor dem Namen PRAKxxx eine 4 und bestätigen mit der Eingabetaste. So gelangen wir auf die Management-Optionen für die lokalen Queues.

Wir haben hier verschiedene Optionen (u.a. Verschieben von Nachrichten in andere Queues und alle Nachrichten löschen) und wählen die 1 für „Delete“.

Wir werden nun nach einer Bestätigung für den Löschvorgang gefragt und bestätigen mit der Eingabetaste. Es erscheint jetzt im „Manage a Local Queue“-Bildschirm die Meldung „`CSQ9022I %CSQ6 CSQMUQLC ' DELETE QLOCAL' NORMAL COMPLETION.`“. Unsere Queue ist gelöscht und wir kehren mit F3 zum Hauptmenü zurück.

5. Anlegen einer Local Queue

Im „IBM WebSphere MQ for z/OS - Main Menu“ erstellen wir jetzt eine neue lokale Warteschlange mit unserer Praktikums-ID als Name.

Wir geben hierfür die Daten wie in Abbildung 6 zu sehen ein und bestätigen mit der Eingabetaste.

```
                    IBM WebSphere MQ for z/OS - Main Menu

Complete fields. Then press Enter.

Action  . . . . . . . . . . 2     0. List with filter   4. Manage
                                  1. List or Display    5. Perform
                                  2. Define like        6. Start
                                  3. Alter              7. Stop
Object type . . . . . . . . QLOCAL        +
Name  . . . . . . . . . . .
Disposition . . . . . . . .    Q=Qmgr, C=Copy, P=Private, G=Group,
                               S=Shared, A=All

Connect name  . . . . . . . CSQ6  - local queue manager or group
Target queue manager  . . . CSQ6
          - connected or remote queue manager for command input
Action queue manager  . . . CSQ6  - command scope in group
Response wait time  . . . . 30    5 - 999 seconds

(C) Copyright IBM Corporation 1993,2005. All rights reserved.




Command ===>
 F1=Help      F2=Split     F3=Exit      F4=Prompt    F9=SwapNext F10=Messages
F12=Cancel
```

Abbildung 6: Erstellen einer Local Queue

Wir lassen hier bewusst den Namen PRAKxxx noch weg, damit eine Queue mit den Standard-Werten erstellt wird. Mit der Eingabetaste gelangen wird zu den Screens mit den Parametern für die Queue.

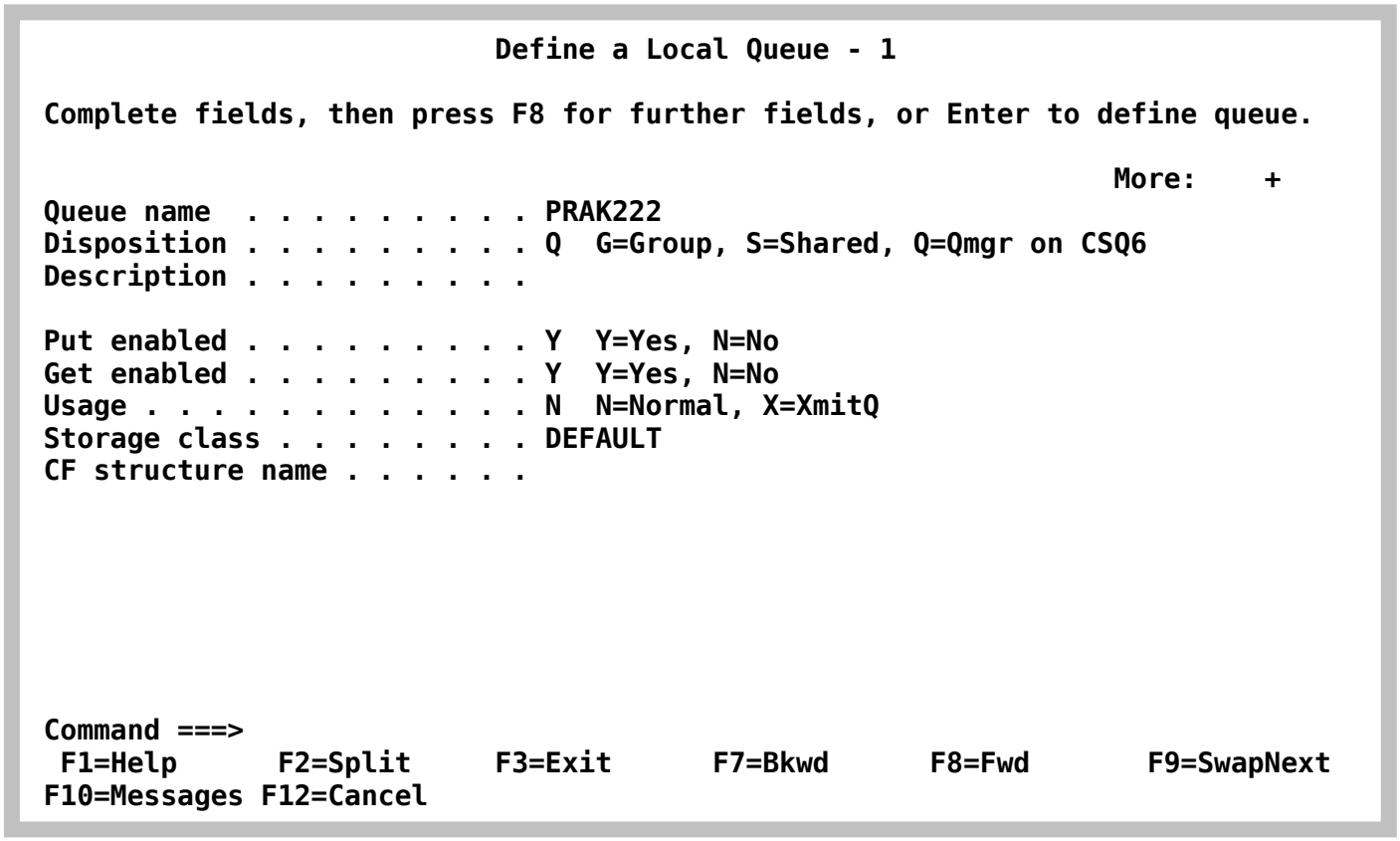

Abbildung 7: Define a Local Queue - 1

Im ersten Bildschirm, Abbildung 7, geben wir als Queue Name unsere Praktikums-ID ein und vergewissern uns, dass die sonstigen Angaben denen in der Abbildung entsprechen. So sollten zum Beispiel PUT und GET jeweils aktiv sein.

```
                          Define a Local Queue - 2

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

                                                                  More:  - +
Queue name  . . . . . . . . : PRAK222
Disposition . . . . . . . . : QMGR    CSQ6

Cluster name  . . . . . . . .
Cluster namelist name . . . .
Default bind  . . . . . . . . O  O=Open, N=Notfixed
Cluster workload rank . . . . 0  0 - 9
Cluster workload priority . . 0  0 - 9
Cluster workload queue use    Q  A=Any, L=Local, Q=Qmgr









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

Abbildung 8: Define a Local Queue – 2

Mit F8 wechseln wir zum nächsten Definitionsbildschirm. Bei diesem Bildschirm brauchen wir keine Änderungen vorzunehmen, wir übernehmen die Daten wie in Abbildung 8 und blättern mit F8 weiter.

```
                          Define a Local Queue - 3

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

                                                                  More:  - +
Queue name  . . . . . . . . : PRAK222
Disposition . . . . . . . . : QMGR    CSQ6

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, G=GroupId,
                                 T=MsgToken

Maximum queue depth . . . . . 999999999  0 - 999999999
Maximum message length  . . . 4194304    0 - 104857600

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

Abbildung 9: Define a Local Queue - 3

Im dritten Definitionsbildschirm (Abbildung 9) müssen wir uns unter anderem entscheiden, ob wir persistente oder nichtpersistente Nachrichten wollen. Eine Nachricht wird dann als „persistent“ bezeichnet, wenn sie nach einem Neustart des Queue Managers wiederhergestellt werden soll. Persistente Nachrichten werden daher geloggt, um zum Beispiel im Falle eines unerwarteten Queue Manager-Ausfalls, keine Nachrichten zu verlieren.

Für unsere Zwecke ist dies aber nicht notwendig, daher deaktivieren wir persistente Nachrichten mit „N“. Des Weiteren haben wir die Möglichkeit, uns zwischen einem prioritisierten und einem FIFO-Abarbeiten der Queue zu entscheiden. Im ersten Fall wird den Nachrichten in der Warteschlange jeweils eine Priorität zugewiesen, die Nachrichten mit der höchsten Priorität werden als Erstes bearbeitet. Bei FIFO (First-In First-Out) wird die Nachricht, welche als Erste in die Schlange eingereiht wurde, auch als Erstes bearbeitet.

Wir entscheiden uns hier für die Priorität („P“).

Wir überprüfen die Daten entsprechend Abbildung 9 und blättern zum vierten Definitionsbildschirm.

```
                        Define a Local Queue - 4

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

                                                                More:  - +
Queue name  . . . . . . . . : PRAK222
Disposition . . . . . . . . : QMGR    CSQ6

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

   Initiation queue . . . . .
   Process name . . . . . . .
   Trigger data . . . . . . .




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

Abbildung 10: Define a Local Queue - 4

Im vierten Bildschirm (Abbildung 10) tragen wir bei „Trigger type“ ein „F“ (für First), bei „Trigger set“ „N“ (für No) ein.

Es folgen jetzt noch zwei weitere Definitionsbildschirme. Hier sind jedoch keine Änderungen erforderlich und so können wir mit der Eingabetaste bestätigen und unsere Queue damit anlegen. Wir können uns natürlich auch noch die restlichen Screens anschauen (Blättern mit F7 und F8) und erst dann die Eingabetaste betätigen.

Wir landen jetzt wieder auf dem ersten Definitionsbildschrim, auf dem die Meldung „CSQ9022I %CSQ6 CSQMAQLC ' DEFINE QLOCAL' NORMAL COMPLETION“ erscheinen sollte (Abbildung 11).

```
                        Define a Local Queue - 1

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

                                                                More:    +
Queue name  . . . . . . . . . PRAK222
Disposition . . . . . . . . . Q  G=Group, S=Shared, Q=Qmgr on CSQ6
Description . . . . . . . . .

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

      +-------------------------------------------------------------+
      ¦ CSQ9022I %CSQ6 CSQMAQLC ' DEFINE QLOCAL' NORMAL COMPLETION ¦
      +-------------------------------------------------------------+
Command ===>
 F1=Help      F2=Split     F3=Exit      F7=Bkwd      F8=Fwd       F9=SwapNext
F10=Messages F12=Cancel
```

Abbildung 11: Queue erfolgreich angelegt

Jetzt überprüfen wir noch, ob unsere Queue auch in der Liste erscheint. Hierzu gehen wir mit F3 zurück zum IBM WebSphere MQ for z/OS – Main Menu. Hier geben wir als Action wieder „1“ für „List or Display“ an und bei Name „PRAK*“ ein (Abbildung 12).

```
                    IBM WebSphere MQ for z/OS - Main Menu

Complete fields. Then press Enter.

Action  . . . . . . . . . . . 1     0. List with filter   4. Manage
                                    1. List or Display    5. Perform
                                    2. Define like        6. Start
                                    3. Alter              7. Stop
Object type . . . . . . . . . QLOCAL        +
Name  . . . . . . . . . . . . PRAK*
Disposition . . . . . . . . .    Q=Qmgr, C=Copy, P=Private, G=Group,
                                 S=Shared, A=All

Connect name  . . . . . . . . CSQ6  - local queue manager or group
Target queue manager  . . . CSQ6
           - connected or remote queue manager for command input
Action queue manager  . . . CSQ6  - command scope in group
Response wait time  . . . . 30    5 - 999 seconds

(C) Copyright IBM Corporation 1993,2005. All rights reserved.




Command ===>
 F1=Help      F2=Split     F3=Exit      F4=Prompt    F9=SwapNext F10=Messages
F12=Cancel
```

Abbildung 12: Queues ausgeben

Wir bestätigen die Eingaben mit der Eingabetaste und gelangen zum Bildschirm mit der Liste der Queues auf unserem Queue Manager CSQ6, die unseren Anforderungen (beginnen mit PRAK) entsprechen (Abbildung 13).

*** ist eine Wildcard und sorgt dafür, dass uns alle Queues, die mit PRAK beginnen, ausgegeben werden.**

Falls es notwendig sein sollte, können wir mit F7 und F8 durch die Liste blättern.

Aufgabe: *Legen Sie eine lokale Queue mit dem Namen PRAKxxx (Entsprechend Ihrer Praktikums-ID) an und überprüfen Sie, ob die Queue angelegt wurde.*

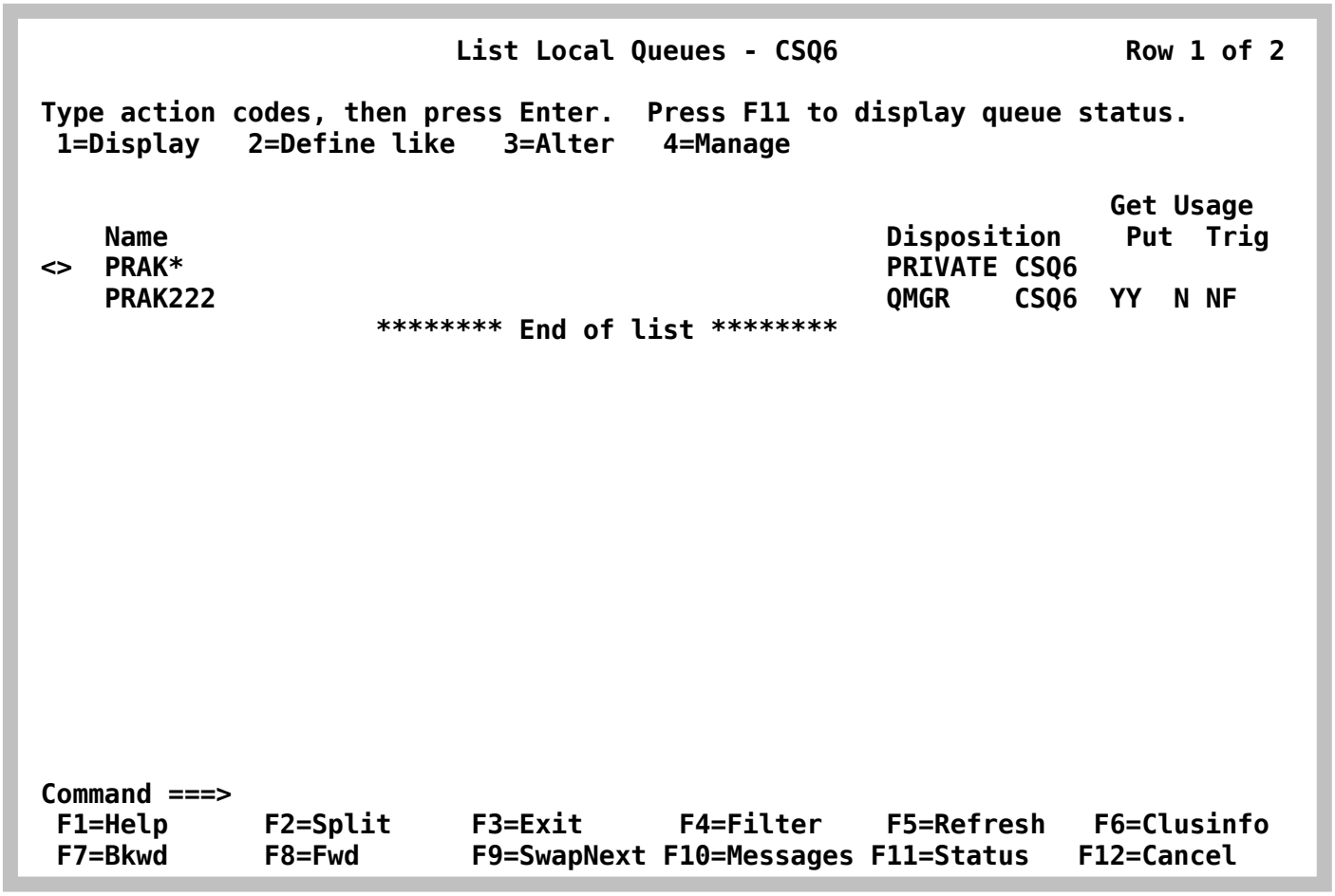

Abbildung 13: Queues ausgeben

Da wir jetzt erfolgreich eine Queue erstellt haben, können wir uns den Anwendungsprogrammen zuwenden. Hierzu wird der Quellcode in COBOL II generiert. Da nicht davon ausgegangen werden kann, dass jeder der dieses Tutorial bearbeitet, COBOL-Kenntnisse besitzt, greifen wir hierfür auf Quellcodes der IBM WebSphere MQ-Bibliotheken zurück. Diese Bibliotheken werden bereits vorkompiliert auf dem Server zur Verfügung gestellt.

Die Bibliothek für die COBOL-Quellcodes hat die Bezeichnung CSQ600.SCSQCOBS.

Um fortzufahren, gehen wir nun ins ISPF Primary Option Menu (also nicht in das IBM Product Panel, von dem aus wir WebSphere MQ aufgerufen haben).

Hier geben wir „3;4“ um direkt ins Data Set List Utility (Abbildung 14) zu gelangen.

6. COBOL Quelltext-Module für WebSphere MQ

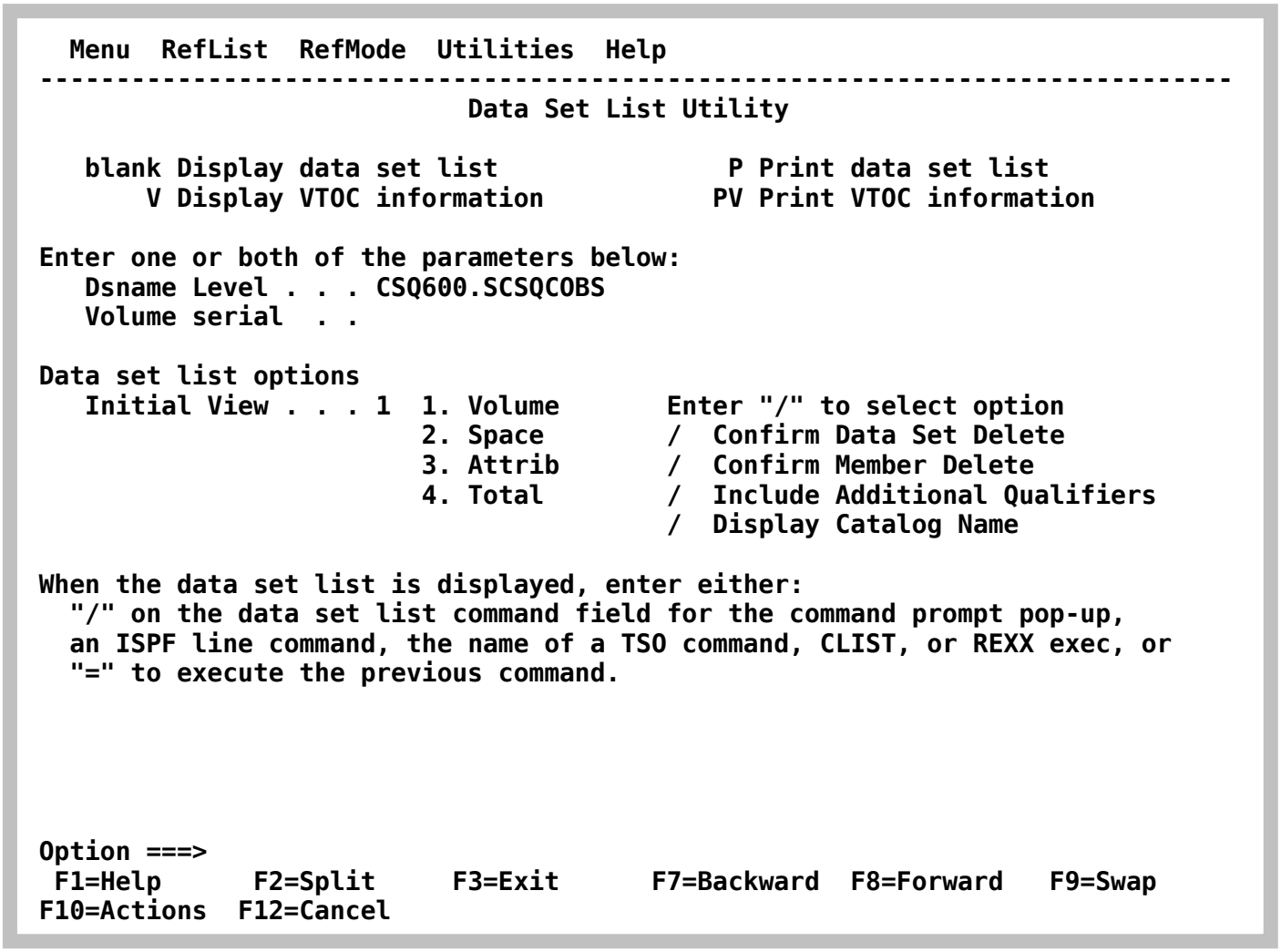

Abbildung 14: Data Set List Utility

Im Data Set List Utility geben wir bei „Dsname Level“ die Data Set-Bezeichnung CSQ600.SCSQCOBS ein. Diese Bibliothek enthält COBOL II-Quelltext-Module für WebSphere MQ, unter anderem die von uns benötigten PUT und GET.

Die „Schwester“-Bibliothek dazu ist SCSQCOBC, welche die COBOL Copy Books enthält. Man erkennt dies leicht an den Endungen der beiden Bibliotheken (S für Source, C für Copy).

Bei „Initial View“ begnügen wir uns mit „1“ für Volume.

Wir bestätigen mit der Eingabetaste und gelangen zum Bildschirm „DSLIST - Data Sets Matching CSQ600.SCSQCOBS“, auf welchem uns die Bezeichnung des Datenträgers angezeigt wird, auf dem das Data Set liegt (Abbildung 15). Diese Angabe ist für uns aber jetzt nicht von Interesse.

In der Command-Spalte (die erste Spalte, vor der Bezeichnung des DataSet) setzen wir ein „b“ (für browse) und bestätigen mit der Eingabetaste. Als Ergebnis bekommen wir eine Ausgabe aller Member des Data Set (Abbildung 16).

```
  Menu  Options  View  Utilities  Compilers  Help
------------------------------------------------------------------------------
DSLIST - Data Sets Matching CSQ600.SCSQCOBS                     Row 1 of 1

Command - Enter "/" to select action                Message          Volume
------------------------------------------------------------------------------
b        CSQ600.SCSQCOBS                                          Z8RES2
****************************** End of Data Set list ****************************
                                                                  Scroll ===> PAGE
Command ===>
 F1=Help    F2=Split   F3=Exit    F5=Rfind   F7=Up      F8=Down    F9=Swap
F10=Left   F11=Right  F12=Cancel
```

Abbildung 15: DSLIST Data Set Matching

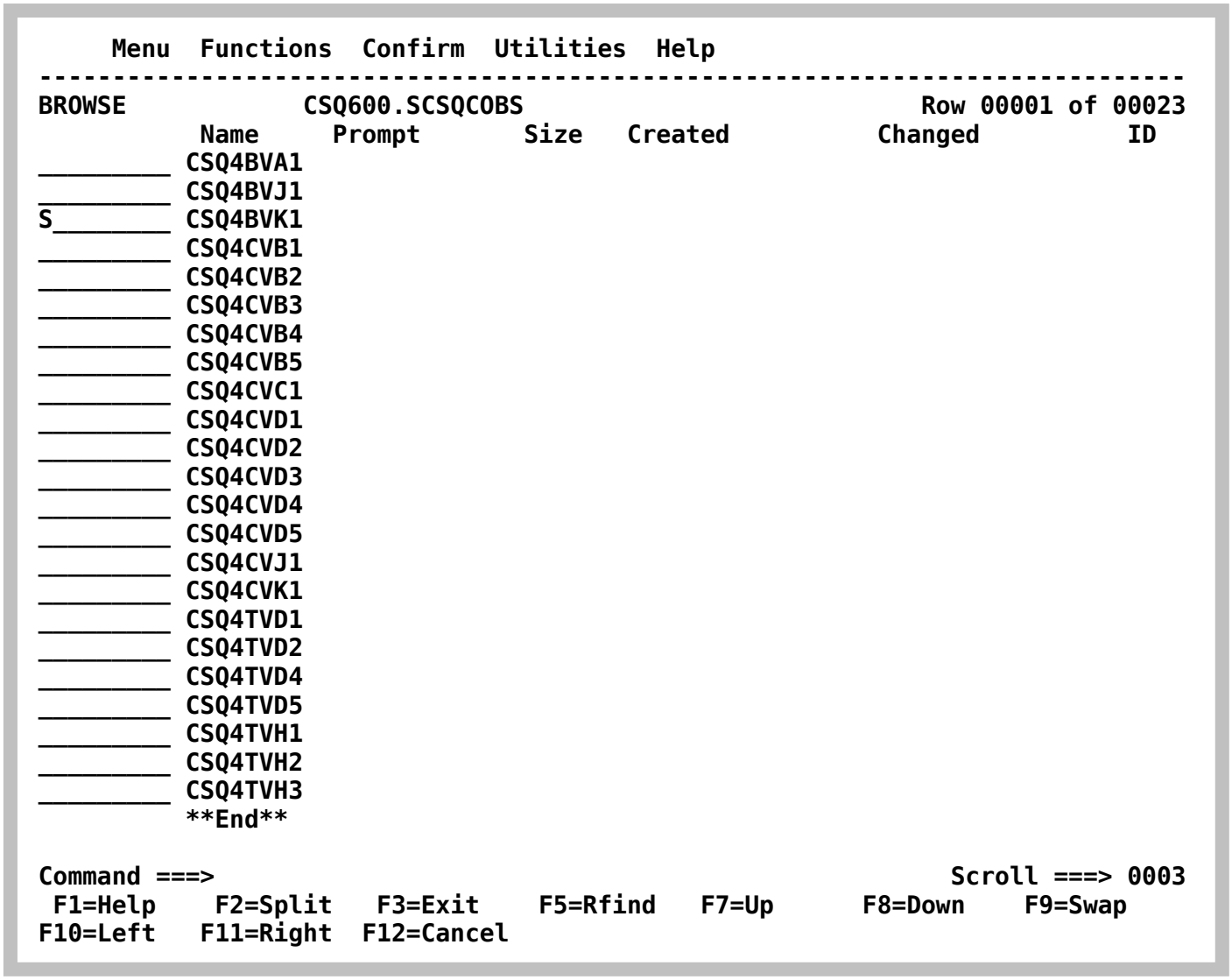

Abbildung 16: DSLIST Data Set Matching

Wenn wir in der Browse-Spalte ein „S“ (für select) eingeben, können wir uns den Quellcode der Module ansehen.

Wir schauen uns die Quellcodes der beiden Module CSQ4BVK1 (put-Modul) und CSQ4BVJ1 (get-Modul) an. Mit den Tasten F7 und F8 können wir durch den Code blättern, mit F3 kommen wir zurück zum Bildschirm aus Abbildung 15.

Im Anhang ist der Cobol Quellcode für den Member CSQ600.SCSQCOBS(CSQ4BVK1) wiedergegeben

Was uns jetzt noch fehlt, ist ein eigenes Data Set für unsere zwei zu kopierende JCL-Scripte, welche die Nachrichten zur Message Queue schicken (MQPUT) und wieder empfangen (MQGET).

Wir legen also, wie in Tutorial 1, ein Data Set mit dem Namen PRAKxxx.CSQ6.USERJCL mit den Werten aus Abbildung 17 an.

7. Anlegen und Kopieren von Datasets

```
  Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
                            Allocate New Data Set

Data Set Name  . . . : PRAK222.CSQ6.USERJCL

Management class . . . DEFAULT        (Blank for default management class)
Storage class  . . . . PRIM90         (Blank for default storage class)
 Volume serial . . . . Z8IMS1         (Blank for system default volume) **
 Device type . . . . .                (Generic unit or device address) **
Data class . . . . . .                (Blank for default data class)
 Space units . . . . . TRACK          (BLKS, TRKS, CYLS, KB, MB, BYTES
                                       or RECORDS)
 Average record unit                  (M, K, or U)
 Primary quantity  . . 3              (In above units)
 Secondary quantity    1              (In above units)
 Directory blocks  . . 5              (Zero for sequential data set) *
 Record format . . . . FB
 Record length . . . . 80
 Block size  . . . . . 6160
 Data set name type    PDS            (LIBRARY, HFS, PDS, LARGE, BASIC, *
                                       EXTREQ, EXTPREF or blank)
 Expiration date . . .                (YY/MM/DD, YYYY/MM/DD
Enter "/" to select option             YY.DDD, YYYY.DDD in Julian form
   Allocate Multiple Volumes           DDDD for retention period in days
                                       or blank)

( * Specifying LIBRARY may override zero directory block)

( ** Only one of these fields may be specified)


Command ===>
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F10=Actions  F12=Cancel
```

Abbildung 17: Neuen Data Set anlegen

In diesem Data Set werden gleich unsere JCL-Scripte für PUT und GET abgelegt. Hierzu kopieren wir aus dem CSQ600.SCSQPROC den Member CSQ4BVJR zweimal, einmal mit der Bezeichnung MQPUT und einmal mit MQGET.

Für die beiden Skripte benötigen wir die Parameter für den Queue Manager QMGR, die lokale Warteschlange QUEUE, die Anzahl der Nachrichten MSGS, das Füllzeichen PAD, die Länge LEN und das Flag persistente/nicht-persistente Nachrichten PERS.

MQPUT wird das Modul CSQ4BVK1 aufrufen und eine Nachricht in die lokale Queue geben, MQGET wird CSQ4BVJ1 aufrufen und das Ergebnis aus der Queue auslesen.

Mit dem Move/Copy Utility können wir den Member bequem in unser Data Set kopieren. Um zum Move/Copy Utility zu gelangen, geben wir im ISPF Primary Option Menu „3;3“ ein und bestätigen mit der Eingabetaste.

Im Move/Copy Utility geben wir, wie in Abbildung 18, bei „From Other Partitioned or Sequential Data Set“ den Data Set Name (Achtung, unbedingt in Hochkomma setzen) 'CSQ600.SCSQPROC(CSQ4BVJR)' ein, als Option wählen wir C (für copy).

```
     Menu  RefList  Utilities  Help
--------------------------------------------------------------------------------
                              Move/Copy Utility

 C  Copy data set or member(s)          CP Copy and print
 M  Move data set or member(s)          MP Move and print

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

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

From Other Partitioned or Sequential Data Set:
   Data Set Name . . . 'CSQ600.SCSQPROC(CSQ4BVJR)'
   Volume Serial . . .            (If not cataloged)

Data Set Password  . .            (If password protected)



Option ===> C
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F10=Actions  F12=Cancel
```

Abbildung 18: Member kopieren

Als nächstes müssen wir noch die Destination angeben, also wohin kopiert werden soll. Wir geben hierfür im nächsten Bildschirm (Abbildung 19) unter „To ISPF Library“ unser zuvor erstelltes Data Set an.
Zunächst kopieren wir das Modul in den Member MQPUT indem wir alle Daten wie in der Abbildung eintragen und mit der Eingabetaste bestätigen.

Als Erfolgsmeldung sollte danach in der oberen rechten Ecke „Member CSQ4BVJR copied“ erscheinen.

```
  Menu  RefList  Utilities  Help
--------------------------------------------------------------------------------
COPY     From CSQ600.SCSQPROC(CSQ4BVJR)

Specify "To" Data Set Below

To ISPF Library:                 Options:
   Project  . . PRAK222             Enter "/" to select option
   Group  . . . CSQ6                   Replace like-named members
   Type . . . . USERJCL             /  Process member aliases
   Member . . . MQPUT     (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
                                    3. Default          3. As is









Command ===>
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F10=Actions  F12=Cancel
```

Abbildung 19: Member kopieren (2)

Jetzt kopieren wir den Member noch einmal, diesmal jedoch in MQGET.

<u>Aufgabe:</u> Erstellen Sie den neuen Dataset und kopieren Sie das JCL Skript zweimal (MQGET und MQPUT).

8. JCL Scripte für MQPUT und MQGET

Als nächstes werden die beiden JCL-Scripte MQPUT und MQGET entsprechend unserer Anforderungen editiert.. Wir beginnen mit MQPUT und öffnen es mit dem TSO-Editor (ISPF Primary Option Menu → „2“).

Innerhalb des Editors lässt sich wieder mit F7 und F8 blättern. Zeilen, die mit „//*“ beginnen, sind auskommentiert.

Als erste Handlung ändern wir die Job-Card (die ersten Zeilen) des JCL-Scripts .
Anstatt CSQ4BVJR geben wir unsere Praktikums-ID gefolgt von einem „P“ für Put ein. Die restlichen Angaben entnehmen wir der Abbildung 20.

```
000001 //PRAK222P JOB (),CLASS=A,MSGCLASS=M,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
000002 //         REGION=4M
000003 //*********************************************************************
```

Abbildung 20: Jobcard

Werfen wir zunächst einen Blick auf den entscheidenden Teil dieses Skripts (Abbildung 21).

```
000059 //JOBLIB   DD   DSN=++THLQUAL++.SCSQLOAD,DISP=SHR
000060 //         DD   DSN=++THLQUAL++.SCSQANL++LANGLETTER++,DISP=SHR
000061 //         DD   DSN=++THLQUAL++.SCSQAUTH,DISP=SHR
000062 //*
000063 //PUTMSGS  EXEC PGM=CSQ4BVK1,REGION=1024K,
000064 //  PARM=('++QMGR++,++QUEUE++,++MSGS++,++PAD++,++LEN++,++PERS++')
000065 //*
000066 //*GETMSGS  EXEC PGM=CSQ4BVJ1,REGION=1024K,
000067 //*  PARM=('++QMGR++,++QUEUE++,++MSGS++,++GET++,++SYNC++')
000068 //*
000069 //SYSDBOUT DD   SYSOUT=*
000070 //SYSABOUT DD   SYSOUT=*
000071 //SYSPRINT DD   SYSOUT=*
000072 //SYSOUT   DD   SYSOUT=*
```

Abbildung 21: Unveränderter Inhalt der Skripte

Wir passen diese Skripte nun entsprechend unserer Anforderungen an (Abbildung 22):

```
000059 //JOBLIB   DD   DSN=CSQ600.SCSQLOAD,DISP=SHR
000060 //         DD   DSN=CSQ600.SCSQANLE,DISP=SHR
000061 //         DD   DSN=CSQ600.SCSQAUTH,DISP=SHR
000062 //*
000063 //PUTMSGS  EXEC PGM=CSQ4BVK1,REGION=1024K,
000064 //  PARM=(CSQ6,PRAK222,3,Z,5,N)
000065 //*
000066 //*GETMSGS  EXEC PGM=CSQ4BVJ1,REGION=1024K,
000067 //*  PARM=('++QMGR++,++QUEUE++,++MSGS++,++GET++,++SYNC++')
000068 //*
000069 //SYSDBOUT DD   SYSOUT=*
000070 //SYSABOUT DD   SYSOUT=*
000071 //SYSPRINT DD   SYSOUT=*
000072 //SYSOUT   DD   SYSOUT=*
```

Abbildung 22: Unser Skript MQPUT

PUTMSGS ruft hier das (COBOL-)Programm CSQ4BVK1 mit den angegebenen Parametern auf.

Neben dem Queue Manager (CSQ6) und unserer Queue geben wir an, dass 3 Nachrichten, bestehend aus je fünfmal dem Zeichen „Z“ (also dem String „ZZZZZ“) gesendet werden sollen. Mit „N“ geben wir an, dass die Nachrichten nicht-persistent sind.

Nachdem wir die Zeilen entsprechend geändert haben, übermitteln wir den Job mit dem „sub“-Kommando. Wir erhalten jetzt die Meldung `JOB PRAK222P(JOB00398) SUBMITTED.` Unser Job hat damit die Bezeichnung PRAKxxxP (Abbildung 23).

```
000058 //JOBLIB   DD   DSN=CSQ600.SCSQLOAD,DISP=SHR
000059 //         DD   DSN=CSQ600.SCSQANLE,DISP=SHR
000060 //         DD   DSN=CSQ600.SCSQAUTH,DISP=SHR
000061 //*
000062 //PUTMSGS  EXEC PGM=CSQ4BVK1,REGION=1024K,
000063 //  PARM=(CSQ6,PRAK222,3,Z,5,N)
000064 //*
000065 //*GETMSGS  EXEC PGM=CSQ4BVJ1,REGION=1024K,
000066 //*  PARM=('++QMGR++,++QUEUE++,++MSGS++,++GET++,++SYNC++')
000067 //*
000068 //SYSDBOUT DD   SYSOUT=*


IKJ56250I JOB PRAK222P(JOB00398) SUBMITTED
***
```

Abbildung 23: Der PUT-Job

```
19.32.43 JOB00398 $HASP165 PRAK222P ENDED AT N1  MAXCC=0 CN(INTERNAL)
 ***
```

Abbildung 24: Rückmeldung auf sub

Wir erhalten als Antwort eine Meldung wie in Abbildung 24. MAXCC=0 bedeutet, dass keine Fehler aufgetreten sind. Eine Ausgabe MAXCC=4 ist eine Warnung, man kann sie jedoch auch vernachlässigen. Alles andere bedeutet einen Fehler. In diesem Fall sollten wir uns das Skript noch einmal anschauen.
Sollte z.B. MAXCC=2085 sein, haben wir vermutlich die falsche Bezeichnung für unsere Queue angegeben.

Wir schreiben jetzt also Nachrichten in die Queue. Jetzt ist es Zeit, diese wieder auszulesen. Dafür wenden wir uns dem Member MQGET zu.

Auch hier passen wir wieder die erste Zeilen (die Job-Card) an. Diesmal vergeben wir als Bezeichnung des Jobs PRAKxxxG.
Die restlichen Zeilen passen wir entsprechend Abbildung 25 an.
Achtung, diesmal nehmen wir GETMSGS, nicht PUTMSGS.

```
000058 //JOBLIB   DD   DSN=CSQ600.SCSQLOAD,DISP=SHR
000059 //         DD   DSN=CSQ600.SCSQANLE,DISP=SHR
000060 //         DD   DSN=CSQ600.SCSQAUTH,DISP=SHR
000061 //*
000062 //*PUTMSGS  EXEC PGM=CSQ4BVK1,REGION=1024K,
000063 //*  PARM=('++QMGR++,++QUEUE++,++MSGS++,++PAD++,++LEN++,++PERS++')
000064 //*
000065 //GETMSGS  EXEC PGM=CSQ4BVJ1,REGION=1024K,
000066 //  PARM=(CSQ6,PRAK222,3,D,N)
000067 //*
000068 //SYSDBOUT DD   SYSOUT=*
000069 //SYSABOUT DD   SYSOUT=*
000070 //SYSPRINT DD   SYSOUT=*
000071 //SYSOUT   DD   SYSOUT=*


Command ===> sub                                          Scroll ===> PAGE
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Abbildung 25: Unser MQGET

Wir übermitteln den Job mit „sub“ und erhalten die Bestätigung, dass der Job PRAKxxxG übertragen wurde.

Auch jetzt sollten wir im Idealfall eine Meldung mit MAXCC=0 bekommen.

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

9. Benutzung von SDSF

Um zu sehen, was wir gerade gemacht haben, gehen wir zunächst durch mehrmaliges Drücken von F3 zurück ins ISPF Primary Option Menu. Von dort aus gelangen wir mit „m;5“ zur Spool Search and Display Facility, SDSF (Abbildung 26).

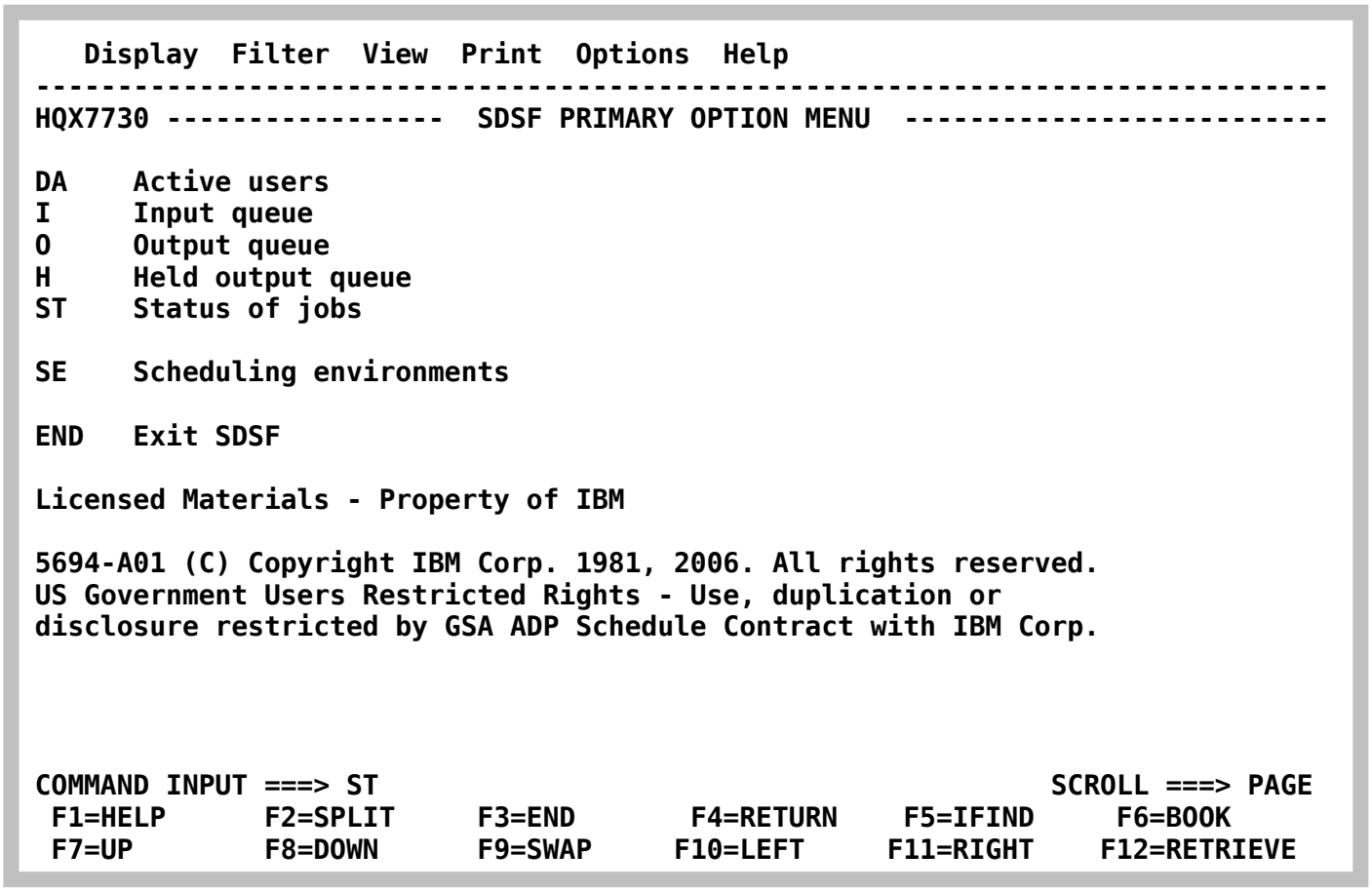

Abbildung 26: SDSF

Wir geben hier „ST“ für „Status of jobs“ ein und gelangen so zu einer Übersicht über alle unsere laufenden Jobs (Abbildung 27).

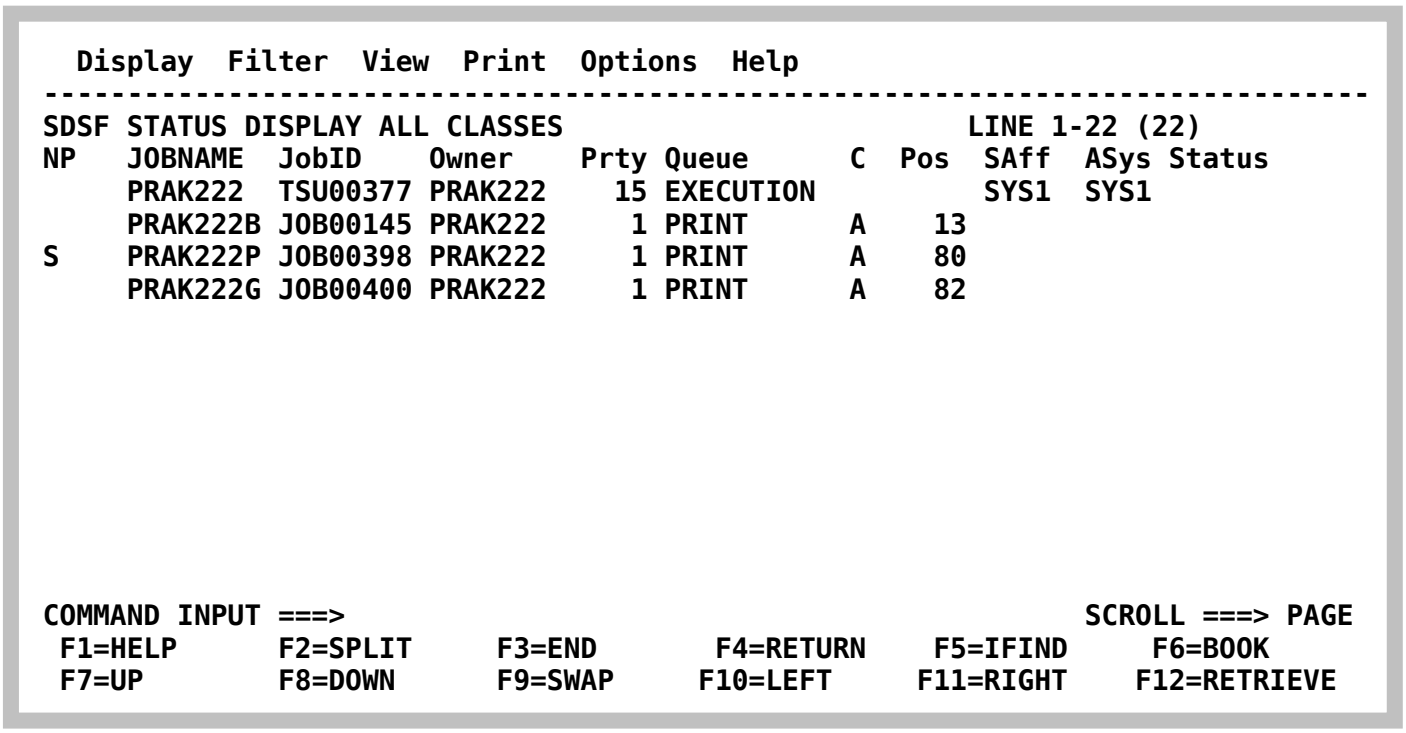

Abbildung 27: SDSF

In der ersten Spalte unseres Put-Jobs (PRAKxxxP) geben wir ein „S“ (für Select) ein und erhalten damit eine Ausgabe unseres Skriptes.

Mit F7 und F8 können wir durch die Ausgabe blättern. Wir blättern bis zum Ende der Ausgabe (Abbildung 28).

```
==========================================
PARAMETERS PASSED :
   QMGR        - CSQ6
   QNAME       - PRAK222
   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 28 Ausgabe MQPUT

Wie wir sehen, hat das Skript mit unseren Parametern MQCONN und MQOPEN ausgeführt, dann die drei Nachrichten in die Queue gelegt und den Prozess mit MQCLOSE und MQDISC erfolgreich beendet.

Mit F3 gelangen wir zurück zur Übersicht und schauen uns auf die gleiche Weise die Ausgabe von PRAKxxxG an (Abbildung 29) .

```
==========================================
PARAMETERS PASSED :
   QMGR        - CSQ6
   QNAME       - PRAK222
   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 29: Ausgabe MQGET

Wir sehen wieder unsere Parameter und die diversen Erfolgsmeldungen der MQ-Befehle.

Des Weiteren erkennen wir unsere drei abgelegten Nachrichten (ZZZZZ).

Aufgabe: ***Bearbeiten Sie die vorher gegebenen Aufgaben und schicken Sie die Print-Screens 28 und 29 im Bitmap- oder JPEG-Format (pro Bild maximal 250 KByte) an die Mailadresse Ihres Betreuers.***

10. Anhang

Code des vorgefertigten Cobol Programms

CSQ600.SCSQCOBS(CSQ4BVK1)

```
5 000001 CBL NODYNAM,LIB,OBJECT,RENT,RES,APOST
 000002       *                                                               *
 000003       * ------------------------------------------------------------- *
 000004        IDENTIFICATION DIVISION.
 000005       * ------------------------------------------------------------- *
 000006        PROGRAM-ID. CSQ4BVK1.
 000007       *REMARKS
 000008       ****************************************************************
 000009       * @START_COPYRIGHT@                                             *
 000010       * Statement:     Licensed Materials - Property of IBM           *
 000011       *                                                               *
 000012       *                5655-F10                                       *
 000013       *                (C) Copyright IBM Corporation. 1993, 2002      *
 000014       *                                                               *
 000015       * Status:        Version 5 Release 3                            *
 000016       * @END_COPYRIGHT@                                               *
 000017       ****************************************************************
 000018       *                   IBM WebSphere MQ for z/OS                   *
 000019       *                                                               *
 000020       *  Module Name      : CSQ4BVK1                                  *
 000021       *                                                               *
 000022       *  Environment      : z/OS Batch; COBOL II                      *
 000023       *                                                               *
 000024       *  Description      : Sample program to put a number of         *
 000025       *                     messages to a queue.                      *
 000026       *                                                               *
 000027       *  Limitation       : Maximum message length set at 65535.      *
 000028       *                                                               *
 000029       ****************************************************************
 000030       *                                                               *
 000031       *                       Program  Logic                          *
 000032       *                       --------------                          *
 000033       *                                                               *
 000034       *  main                                                         *
 000035       *  ----                                                         *
 000036       *                                                               *
 000037       *   Move parameters into corresponding variables.               *
 000038       *   If parameters are invalid then                              *
 000039       *            Call USAGE-ERROR and exit.                         *
 000040       *                                                               *
 000041       *   Display the parameters passed to the program.               *
 000042       *                                                               *
 000043       *   Connect to the queue manager.                               *
 000044       *   If connection failed then                                   *
 000045       *            Call DISPLAY-ERROR-MESSAGE and exit                *
 000046       *                                                               *
 000047       *   Set the open options.                                       *
 000048       *   Open the specified message queue (MQOPEN).                  *
 000049       *   If open failed then                                         *
 000050       *            Disconnect from queue manager                      *
 000051       *            Call DISPLAY-ERROR-MESSAGE and exit                *
```

CSQ600.SCSQCOBS(CSQ4BVK1) Teil 1

```
000052       *   Endif.                                                    *
000053       *                                                             *
000054       *   Set some message properties.                              *
000055       *   Set the put message options.                              *
000056       *   Loop while the messages are put to queue                  *
000057       *            Put message to queue (MQPUT)                     *
000058       *            If put failed                                    *
000059       *                     Call DISPLAY-ERROR-MESSAGE              *
000060       *                     Break from loop                         *
000061       *            Endif                                            *
000062       *   Endloop.                                                  *
000063       *   Display number of messages put to the queue.              *
000064       *                                                             *
000065       *   Close the message queue.                                  *
000066       *   If close failed then                                      *
000067       *            Call DISPLAY-ERROR-MESSAGE.                      *
000068       *                                                             *
000069       *   Disconnect from the queue manager.                        *
000070       *   If disconnect failed then                                 *
000071       *            Call DISPLAY-ERROR-MESSAGE.                      *
000072       *                                                             *
000073       *   Exit program.                                             *
000074       *                                                             *
000075       *-------------------------------------------------------------*
000076       *                                                             *
000077       *  USAGE-ERROR                                                *
000078       *  -----------                                                *
000079       *                                                             *
000080       *   Print message showing correct usage for program.          *
000081       *                                                             *
000082       *-------------------------------------------------------------*
000083       *                                                             *
000084       *  DISPLAY-ERROR-MESSAGE                                      *
000085       *  ---------------------                                      *
000086       *                                                             *
000087       *   Create error message and display.                         *
000088       *                                                             *
000089       ***************************************************************
000090       * ----------------------------------------------------------- *
000091        ENVIRONMENT DIVISION.
000092       * ----------------------------------------------------------- *
000093       * ----------------------------------------------------------- *
000094        DATA DIVISION.
000095       * ----------------------------------------------------------- *
000096        FILE SECTION.
000097       * ----------------------------------------------------------- *
000098        WORKING-STORAGE SECTION.
000099       * ----------------------------------------------------------- *
000100       *
000101       *    W00 - General work fields
000102       *
000103        01  W00-RETURN-CODE             PIC S9(4) BINARY VALUE ZERO.
000104        01  W00-LOOP                    PIC S9(9) BINARY VALUE 0.
000105        01  W00-NUMPUTS                 PIC S9(9) BINARY VALUE 0.
```

CSQ600.SCSQCOBS(CSQ4BVK1) Teil 2

```
000106        01  W00-ERROR-MESSAGE           PIC X(48) VALUE SPACES.
000107       *
000108       *   Parameter variables
000109       *
000110        01  W00-QMGR                    PIC X(48).
000111        01  W00-QNAME                   PIC X(48).
000112        01  W00-PADCHAR                 PIC X(1) VALUE '*'.
000113        01  W00-MSGBUFFER.
000114          02  W00-MSGBUFFER-ARRAY       PIC X(1) OCCURS 65535 TIMES.
000115        01  W00-NUMMSGS-NUM             PIC 9(4) VALUE  0.
000116        01  W00-NUMMSGS                 PIC S9(9) BINARY VALUE 1.
000117        01  W00-MSGLENGTH-NUM           PIC 9(4) VALUE 0.
000118        01  W00-MSGLENGTH               PIC S9(9) BINARY VALUE 100.
000119        01  W00-PERSISTENCE             PIC X(1) VALUE 'N'.
000120            88 PERSISTENT     VALUE 'P'.
000121            88 NOT-PERSISTENT VALUE 'N'.
000122       *
000123       *    W03 - API fields
000124       *
000125        01  W03-HCONN                   PIC S9(9) BINARY VALUE 0.
000126        01  W03-HOBJ                    PIC S9(9) BINARY VALUE 0.
000127        01  W03-OPENOPTIONS             PIC S9(9) BINARY.
000128        01  W03-COMPCODE                PIC S9(9) BINARY.
000129        01  W03-REASON                  PIC S9(9) BINARY.
000130       *
000131       *    API control blocks
000132       *
000133        01  MQM-OBJECT-DESCRIPTOR.
000134            COPY CMQODV.
000135        01  MQM-MESSAGE-DESCRIPTOR.
000136            COPY CMQMDV.
000137        01  MQM-PUT-MESSAGE-OPTIONS.
000138            COPY CMQPMOV.
000139       *
000140       *    MQV contains constants (for filling in the control blocks)
000141       *    and return codes (for testing the result of a call)
000142       *
000143        01  MQM-CONSTANTS.
000144            COPY CMQV SUPPRESS.
000145       *
000146       *
000147       * ------------------------------------------------------------- *
000148        LINKAGE SECTION.
000149       * ------------------------------------------------------------- *
000150        01  PARMDATA.
000151            05  PARM-LEN                PIC S9(03) BINARY.
000152            05  PARM-STRING             PIC X(100).
000153       *
000154            EJECT
000155       * ------------------------------------------------------------- *
000156        PROCEDURE DIVISION USING PARMDATA.
000157       * ------------------------------------------------------------- *
000158       * ------------------------------------------------------------- *
000159        A-MAIN SECTION.
```

CSQ600.SCSQCOBS(CSQ4BVK1) Teil 3

```
000160       * ------------------------------------------------------------ *
000161       *
000162       *    If no parameters passed to program then
000163       *    call USAGE-ERROR and exit
000164       *
000165            IF PARM-LEN = 0 THEN
000166               PERFORM USAGE-ERROR
000167               MOVE 8 TO W00-RETURN-CODE
000168               GO TO A-MAIN-END
000169            END-IF.
000170       *
000171       *    Move parameters into corresponding variables
000172       *
000173            UNSTRING PARM-STRING DELIMITED BY ALL ','
000174                        INTO W00-QMGR
000175                             W00-QNAME
000176                             W00-NUMMSGS-NUM
000177                             W00-PADCHAR
000178                             W00-MSGLENGTH-NUM
000179                             W00-PERSISTENCE.
000180            MOVE W00-MSGLENGTH-NUM TO W00-MSGLENGTH.
000181            MOVE W00-NUMMSGS-NUM   TO W00-NUMMSGS.
000182       *
000183       *    Display parameters to be used in the program
000184       *
000185            DISPLAY '=========================================='.
000186            DISPLAY 'PARAMETERS PASSED :'.
000187            DISPLAY '   QMGR        - ', W00-QMGR.
000188            DISPLAY '   QNAME       - ', W00-QNAME.
000189            DISPLAY '   NUMMSGS     - ', W00-NUMMSGS.
000190            DISPLAY '   PADCHAR     - ', W00-PADCHAR.
000191            DISPLAY '   MSGLENGTH   - ', W00-MSGLENGTH.
000192            DISPLAY '   PERSISTENCE - ', W00-PERSISTENCE.
000193            DISPLAY '=========================================='.
000194       *
000195       *    Setup the message buffer
000196       *
000197            PERFORM WITH TEST BEFORE VARYING W00-LOOP FROM 1 BY 1
000198                UNTIL (W00-LOOP > W00-MSGLENGTH)
000199       *
000200                MOVE W00-PADCHAR TO W00-MSGBUFFER-ARRAY(W00-LOOP)
000201       *
000202            END-PERFORM.
000203       *
000204       *
000205       *    Connect to the queue manager
000206       *
000207            CALL 'MQCONN' USING W00-QMGR
000208                                W03-HCONN
000209                                W03-COMPCODE
000210                                W03-REASON.
000211       *
000212       *    If connection failed then display error message
000213       *    and exit
```

CSQ600.SCSQCOBS(CSQ4BVK1) Teil 4

```
000214       *
000215            IF (W03-COMPCODE NOT = MQCC-OK) THEN
000216               MOVE 'MQCONN'   TO W00-ERROR-MESSAGE
000217               PERFORM DISPLAY-ERROR-MESSAGE
000218               MOVE W03-REASON TO W00-RETURN-CODE
000219               GO TO A-MAIN-END
000220            END-IF.
000221            DISPLAY 'MQCONN SUCCESSFUL'.
000222       *
000223       *
000224       *    Open the queue for output. Fail the call if the queue
000225       *    manager is quiescing.
000226       *
000227            COMPUTE W03-OPENOPTIONS = MQOO-OUTPUT +
000228                                      MQOO-FAIL-IF-QUIESCING.
000229
000230            MOVE W00-QNAME   TO MQOD-OBJECTNAME.
000231       *
000232            CALL 'MQOPEN' USING W03-HCONN
000233                                MQOD
000234                                W03-OPENOPTIONS
000235                                W03-HOBJ
000236                                W03-COMPCODE
000237                                W03-REASON.
000238       *
000239       *    If open failed then display error message
000240       *    and exit.
000241       *
000242            IF (W03-COMPCODE NOT = MQCC-OK) THEN
000243               MOVE 'MQOPEN'   TO W00-ERROR-MESSAGE
000244               PERFORM DISPLAY-ERROR-MESSAGE
000245               MOVE W03-REASON TO W00-RETURN-CODE
000246               GO TO A-MAIN-DISCONNECT
000247            END-IF.
000248            DISPLAY 'MQOPEN SUCCESSFUL'.
000249       *
000250       *    Set persistence depending on parameter passed
000251       *
000252            IF PERSISTENT THEN
000253               MOVE MQPER-PERSISTENT     TO MQMD-PERSISTENCE
000254            ELSE
000255               MOVE MQPER-NOT-PERSISTENT TO MQMD-PERSISTENCE
000256            END-IF.
000257       *
000258       *    Put string format messages
000259       *
000260            MOVE MQFMT-STRING TO MQMD-FORMAT.
000261       *
000262       *    Set the put message options to fail the call if the
000263       *    queue manager is quiescing
000264       *
000265            MOVE MQPMO-FAIL-IF-QUIESCING TO MQPMO-OPTIONS.
000266       *
000267       *    Loop until specified number of messages put to queue
```

CSQ600.SCSQCOBS(CSQ4BVK1) Teil 5

```
000268       *
000269            PERFORM WITH TEST BEFORE VARYING W00-LOOP FROM 0 BY 1
000270                UNTIL (W00-LOOP >= W00-NUMMSGS)
000271       *
000272                MOVE MQMI-NONE TO MQMD-MSGID
000273                MOVE MQCI-NONE TO MQMD-CORRELID
000274       *
000275                CALL 'MQPUT' USING W03-HCONN
000276                                   W03-HOBJ
000277                                   MQMD
000278                                   MQPMO
000279                                   W00-MSGLENGTH
000280                                   W00-MSGBUFFER
000281                                   W03-COMPCODE
000282                                   W03-REASON
000283       *
000284       *        If put failed then display error message
000285       *        and break out of loop
000286       *
000287                IF (W03-COMPCODE NOT = MQCC-OK) THEN
000288                   MOVE 'MQPUT'     TO W00-ERROR-MESSAGE
000289                   PERFORM DISPLAY-ERROR-MESSAGE
000290                   MOVE W00-NUMMSGS TO W00-LOOP
000291                   MOVE W03-REASON  TO W00-RETURN-CODE
000292                ELSE
000293                   ADD 1 TO W00-NUMPUTS
000294                END-IF
000295       *
000296            END-PERFORM.
000297       *
000298       *    Display the number of messages successfully put
000299       *    to the queue
000300       *
000301            DISPLAY W00-NUMPUTS, ' MESSAGES PUT TO QUEUE'.
000302       *
000303       *
000304       *    Close the queue
000305       *
000306            CALL 'MQCLOSE' USING W03-HCONN
000307                                 W03-HOBJ
000308                                 MQCO-NONE
000309                                 W03-COMPCODE
000310                                 W03-REASON.
000311            IF (W03-COMPCODE NOT = MQCC-OK) THEN
000312               MOVE 'MQCLOSE'  TO W00-ERROR-MESSAGE
000313               PERFORM DISPLAY-ERROR-MESSAGE
000314               MOVE W03-REASON TO W00-RETURN-CODE
000315            ELSE
000316               DISPLAY 'MQCLOSE SUCCESSFUL'
000317            END-IF.
000318       *
000319       *
000320       *
000321        A-MAIN-DISCONNECT.
```

CSQ600.SCSQCOBS(CSQ4BVK1) Teil 6

```
000322       *
000323       *    Disconnect from the queue manager
000324       *
000325            CALL 'MQDISC' USING W03-HCONN
000326                                W03-COMPCODE
000327                                W03-REASON.
000328            IF (W03-COMPCODE NOT = MQCC-OK) THEN
000329               MOVE 'MQDISC'   TO W00-ERROR-MESSAGE
000330               PERFORM DISPLAY-ERROR-MESSAGE
000331               MOVE W03-REASON TO W00-RETURN-CODE
000332            ELSE
000333               DISPLAY 'MQDISC SUCCESSFUL'
000334            END-IF.
000335       *
000336        A-MAIN-END.
000337       *
000338       *
000339            MOVE W00-RETURN-CODE TO RETURN-CODE
000340            STOP RUN.
000341       *
000342       * ------------------------------------------------------------ *
000343        USAGE-ERROR SECTION.
000344       * ------------------------------------------------------------ *
000345       *
000346            DISPLAY '================================================'.
000347            DISPLAY 'PARAMETERS FOR PROGRAM :'.
000348            DISPLAY '     QMGR        - QUEUE MANGER'.
000349            DISPLAY '     QNAME       - QUEUE NAME'.
000350            DISPLAY '     NUMMSGS     - NUMBER OF MESSAGES'.
000351            DISPLAY '     PADCHAR     - MESSAGE PADDING CHARACTER'.
000352            DISPLAY '     MSGLENGTH   - LENGTH OF MESSAGE(S)'.
000353            DISPLAY '     PERSISTENCE - PERSISTENCE OF MESSAGE(S)'.
000354            DISPLAY '================================================'.
000355       *
000356        USAGE-ERROR-END.
000357       *
000358       *    RETURN TO PERFORMING FUNCTION
000359       *
000360            EXIT.
000361       *
000362       * ------------------------------------------------------------ *
000363        DISPLAY-ERROR-MESSAGE SECTION.
000364       * ------------------------------------------------------------ *
000365       *
000366            DISPLAY '************************************************'.
000367            DISPLAY '* ', W00-ERROR-MESSAGE.
000368            DISPLAY '* COMPLETION CODE : ', W03-COMPCODE.
000369            DISPLAY '* REASON CODE     : ', W03-REASON.
000370            DISPLAY '************************************************'.
000371       *
000372        DISPLAY-ERROR-MESSAGE-END.
000373       *
000374       *    RETURN TO PERFORMING FUNCTION
000375       *
```

CSQ600.SCSQCOBS(CSQ4BVK1) Teil 7

```
000376            EXIT.
000377       *
000378       * ---------------------------------------------------------- *
000379       *                          END OF PROGRAM
000380       * ---------------------------------------------------------- *
```

CSQ600.SCSQCOBS(CSQ4BVK1) Teil 8