```
/*  Diese statische Methode versucht den Versand zu einem Server
   <p>

   @author     Christian Koenig
   @version    1.0 Erste Version

   @param      bytearray - Das Byte-Array mit der zu versendenden Nachricht.
   @param      nachrichteninfo - Nachrichteninfo-Objekt, das die Nachricht beschreibt; kann null
sein
   @param      server - Name des Servers
   @param      qMgrName - Name des Queuemanagers
   @param      qMgrPort - Port des Queuemanagers
   @param      channel - Name des Server Connection Channels
   @param      id - Die Benutzerkennung
   @return     '0', wenn alles geklappt hat, die Fehlernummer sonst
   */
   private static int versucheVersand (byte[]   bytearray,
                                       SN_Nachrichteninfo nachrichteninfo,
                                       String   server,
                                       String   qMgrName,
                                       int      qMgrPort,
                                       String   channel,
                                       String   id)
   {
      // Variable für den Rückgabewert
      int          rueckgabewert = 0;

      // ID ggf. in einen Leerstring umsetzen
      if(id==null) id = "";

      // Environment definieren
      MQEnvironment.hostname = server;
      MQEnvironment.port     = qMgrPort;
      MQEnvironment.channel  = channel;
      MQEnvironment.userID   = id;
      MQEnvironment.disableTracing();

      // Log abdrehen
      MQException.log = null;

        // Erzeugen des Queuemanagers
      try
      {
         // Queue Manager erzeugen
         qMgr = new MQQueueManager(qMgrName);

      } // Ende try
      catch (Exception e)
      {
         // Übler Fehler: Die Connection zu dem MQS-Server ließ sich nicht erzeugen :-(

         // Falls nötig, Queuemanager schließen
         schliesseQMgr();

         // Rueckgabewert ist nun -1; dieser wird von der aufrufenden Funktion je nach Server
gemapped.
```

```
        rueckgabewert = -1;
        return rueckgabewert;
      } // Ende catch

      // Optionen für den Queuezugriff festlegen
      int openOptionen = MQC.MQOO_OUTPUT | MQC.MQOO_SET_IDENTITY_CONTEXT |
MQC.MQOO_FAIL_IF_QUIESCING ;

      // Queuenamen basteln
      String queueName = <Name>;

      try
      {
        queue = qMgr.accessQueue(queueName,
                                 openOptionen,
                                 null,
                                 null,
                                 null);
      } // Ende try
      catch (Exception e)
      {


        // Falls nötig, Queuemanager schließen
        schliesseQMgr();

        // Die Connection zur Queue ließ sich nicht erzeugen :-(
        rueckgabewert = 4;
        return rueckgabewert;

      } // Ende catch

      int characterSet = 0;

// Erstellen und Versenden der Nachricht

      if ( !willGruppenversand )
      {
        // Versand als Stücknachricht

        try
        {
          // Erstellen

          MQMessage nachricht = new MQMessage();
          nachricht.format    = MQC.MQFMT_NONE;
          nachricht.userId    = id;
          nachricht.write(bytearray);

          // Options festlegen
          MQPutMessageOptions pmo = new MQPutMessageOptions();

          pmo.options = MQC.MQGMO_NO_SYNCPOINT |
MQC.MQPMO_SET_IDENTITY_CONTEXT | MQC.MQPMO_FAIL_IF_QUIESCING  ;

          // ab damit
```

```
            queue.put(nachricht,pmo);

            // Und schließen
            queue.close();

          } // Ende try
          catch (Exception e)
          {
            // Nachricht ließ sich nicht in die Queue stellen

            // Falls nötig, Queuemanager schließen
            schliesseQMgr();

            rueckgabewert = 5;
            return rueckgabewert;

          } // Ende catch
        } // Ende if (Stückversand)

        else

        {
          // Versand als Gruppennachricht

          try
          {
            // Optionen festlegen
            MQPutMessageOptions pmo = new MQPutMessageOptions();
            pmo.options = MQC.MQGMO_NO_SYNCPOINT |
MQC.MQPMO_SET_IDENTITY_CONTEXT | MQC.MQPMO_FAIL_IF_QUIESCING |
MQC.MQPMO_LOGICAL_ORDER ;

            // Teilnachricht fuer den Vorsatz erzeugen
            MQMessage nachricht = new MQMessage();
            nachricht.format    = MQC.MQFMT_STRING;
            nachricht.userId    = id;
            nachricht.write(nachrichteninfo.gibVorsatzUndDaten().bytearray);
            nachricht.messageFlags = MQC.MQMF_MSG_IN_GROUP;

            // Teilnachricht absenden
            queue.put(nachricht,pmo);

            // nun die Anhaenge, der Reihe nach:
            for (int i = 1; i <= nachrichteninfo.gibAnzahlAnhaenge(); i++)
            {
              // Teilnachricht erzeugen
              nachricht = new MQMessage();
              nachricht.format    = MQC.MQFMT_NONE;
              nachricht.userId    = id;
              nachricht.write(nachrichteninfo.gibAnhang(i).bytearray);
              nachricht.messageFlags = MQC.MQMF_MSG_IN_GROUP;
              // das characterSet darf hier nicht gesetzt werden, weil MQSeries die Teilnachricht sonst
nicht lesen kann!

              // Teilnachricht senden
              queue.put(nachricht,pmo);
```

```
        } // Ende for

        // Teilnachricht fuer den Nachsatz erzeugen
        nachricht = new MQMessage();
        nachricht.format    = MQC.MQFMT_STRING;
        nachricht.userId    = id;
        nachricht.write(nachrichteninfo.gibNachsatz().bytearray);
        nachricht.messageFlags = MQC.MQMF_LAST_MSG_IN_GROUP;

        // Teilnachricht mit dem Nachsatz senden:
        queue.put(nachricht,pmo);

        // Und schließen
        queue.close();

      } // Ende try
      catch (Exception e)
      {
        // Nachricht ließ sich nicht in die Queue stellen

        // Falls nötig, Queuemanager schließen
        schliesseQMgr();

        rueckgabewert = 5;
        return rueckgabewert;

      } // Ende catch


    }


    // Falls nötig, Queuemanager schließen
    schliesseQMgr();


    return rueckgabewert;
  } // Ende versucheVersand



/**
  Diese statische Methode versucht das Auslesen einer Nachricht in ein Bytearray
  <p>

  @author     Christian Koenig
  @version    1.0 Erste Version

  @param      queueName - Der Name der Queue
  @param      bytearray - Das SN_Data Byte-Array für die Nachricht.
  @param      qMgr - Der Queuemanager
  @return     '0', wenn alles geklappt hat, die Fehlernummer sonst
  */
  public static int liesNachricht (String             queueName,
                                   SN_Data_Bytearray  snBytearray,
```

```
                        MQQueueManager      qMgr)
  {
    // Variable für den Rückgabewert
    int             rueckgabewert = 0;

    // Queue erzeugen
    MQQueue         queue;

    // Log abdrehen
    MQException.log = null;

     // testen, ob der uebergebene Queuemanager existiert
    if (qMgr == null)
    {
      // offensichtlich nicht :-(
      rueckgabewert = 1;
      return rueckgabewert;

    } // Ende if


    // Optionen für den Queuezugriff festlegen
    int openOptionen = MQC.MQOO_INPUT_AS_Q_DEF | MQC.MQOO_FAIL_IF_QUIESCING;

    try
    {
      queue = qMgr.accessQueue(queueName,
                        openOptionen,
                        null,
                        null,
                        null);
    } // Ende try
    catch (Exception e)
    {
      // Die Connection zur Queue ließ sich nicht erzeugen :-(

      rueckgabewert = 2;
      return rueckgabewert;

    } // Ende catch

    // Optionen fuer das Holen der Nachricht oder Teilnachricht
    MQGetMessageOptions gmo = new MQGetMessageOptions();
    gmo.options = MQC.MQGMO_NO_WAIT | MQC.MQGMO_NO_SYNCPOINT |
MQC.MQGMO_FAIL_IF_QUIESCING
          | MQC.MQGMO_ALL_MSGS_AVAILABLE | MQC.MQGMO_LOGICAL_ORDER
          | MQC.MQGMO_CONVERT;

    // Holen der Nachricht oder Teilnachricht
    rueckgabewert = holeTeilnachricht(queue, qMgr, gmo, snBytearray);

    if ( rueckgabewert != 0 )
    {
      // holeTeilnachricht war nicht erfolgreich
      return rueckgabewert;
    }
```

```
// im Fall von Gruppennachrichten werden die weiteren Teilnachrichten gelesen

if ( gmo.groupStatus == MQC.MQGS_MSG_IN_GROUP )
{
  // Vektor, der die Teilnachrichten aufnimmt
  Vector teilnachrichten = new Vector();

  // die Länge des bereits gelesenen ersten Teils wird als Gesamtlaenge uebernommen
  int gesamtlaenge = snBytearray.bytearray.length;

  // speichere den bereits gelesenen ersten Teil der Nachricht im Vector
  teilnachrichten.addElement (snBytearray.bytearray);

  // hole die restlichen Teilnachrichten in den Vector
  while ( gmo.groupStatus == MQC.MQGS_MSG_IN_GROUP )
  {
    rueckgabewert = holeTeilnachricht(queue, qMgr, gmo, snBytearray);

    if ( rueckgabewert != 0 )
    {
      // holeTeilnachricht war nicht erfolgreich
      return rueckgabewert;
    } // Ende if

    // Teilnachricht im Vektor speichern und Gesamtlaenge aktualisieren
    teilnachrichten.addElement (snBytearray.bytearray);
    gesamtlaenge = gesamtlaenge + snBytearray.bytearray.length;
  } // Ende while

  // Nun sind alle Teile gelesen. Das Bytearray wird neu zusammengesetzt.

  byte[] gesamtnachricht = new byte[gesamtlaenge];
  int offset = 0;

  try
  {
    // Kopiere jede Teilnachricht in das Array.
    // offset zeigt dabei das Array-Offset für den naechsten Teil an.

    for (int i = 0; i < teilnachrichten.size(); i++)
    {
      // Nimm die Teilnachricht i des Vektors...
      byte[] teilnachricht = (byte[])teilnachrichten.elementAt(i);

      // ...und kopiere sie in die Gesamtnachricht.
      System.arraycopy(teilnachricht,0,gesamtnachricht, offset, teilnachricht.length);
      offset = offset + teilnachricht.length;

    } // Ende for

  } // Ende try
  catch (Exception e)
  {
    // arraycopy warf Exception
```

```
          rueckgabewert = 5;

          return rueckgabewert;

        } // Ende catch

        // der (Ausgabe-)Parameter snBytearray wird mit der Gesamtnachricht belegt
        snBytearray.bytearray = gesamtnachricht;

      } // Ende if ( Gruppennachricht )

      // Falls nötig, Queue schließen
      schliesseQueue(queue,qMgr);

      return rueckgabewert;
    } // Ende liesNachricht


/**
      holeTeilnachricht holt eine Stück- oder Teilnachricht aus der Queue.

      @author     Keno Hamer
      @version    1.0
      @param      queue          die MQSeries-Queue
      @param      qMgr           der MQSeries-Queuemanager
      @param      gmo            Optionen (MQGetMessageOptions) fuer das Lesen aus der Queue
      @param      snBytearray    das Array, in dem die gelesene Nachricht abgelegt wird
      @return     int - Returncode
                  0 - alles ok
                  sonst die Fehlernummer
                  6 - Konvertierung misslungen
    */

    private static int holeTeilnachricht (MQQueue queue, MQQueueManager qMgr,
MQGetMessageOptions gmo, SN_Data_Bytearray snBytearray)
    {
      int rueckgabewert = 0;

      // Erstellen einer Nachricht
      MQMessage nachricht = new MQMessage();

      // Holen der Nachricht
      try
      {
         queue.get(nachricht,gmo);

      } // Ende try
      catch (MQException e)
      {

        // Nachricht ließ sich nicht aus der Queue holen

        // Vor dem Schliessen der Queue wird versucht, restliche Teilnachrichten
        // aus der Queue zu entfernen.
        if ( gmo.groupStatus == MQC.MQGS_MSG_IN_GROUP )
        {
```

```
            // printGMO("Fehler bei Teilnachricht.", gmo, nachricht);
            liesRestlicheTeilnachrichten(queue,gmo);
          } // Ende if

          // Falls möglich, Queue schließen
          schliesseQueue(queue,qMgr);

          // InitQ-Leiche ?
          if (e.reasonCode==MQException.MQRC_NO_MSG_AVAILABLE)
          {
            rueckgabewert = 4;
          } // Ende if

          // Konvertierung gescheitert?
          else if (e.reasonCode == MQException.MQRC_NOT_CONVERTED || e.reasonCode ==
MQException.MQRC_SOURCE_CCSID_ERROR)
          {
            rueckgabewert = 6;
          }

          else
          {
            rueckgabewert = 3;
          } // Ende else

          return rueckgabewert;

        } // Ende catch

        try
        {
          // Bytearray anlegen
          snBytearray.bytearray = new byte[nachricht.getMessageLength()];

          // Lesen der Nachricht
          nachricht.readFully(snBytearray.bytearray);

        } // Ende try
        catch (Exception e)
        {
          // Nachricht ließ sich nicht in das Bytearray stellen

          // Falls möglich, Queue schließen
          schliesseQueue(queue,qMgr);

          rueckgabewert = 5;
          return rueckgabewert;

        } // Ende catch

        return rueckgabewert;

      } // Ende holeTeilnachricht


      /**
```

```
    Die Methode liest im Fehlerfall die restlichen Teile einer Gruppennachricht aus
    der Queue, damit kein Muell in der Queue zurueckbleibt.
    Die Teilnachrichten werden nicht gespeichert.

    @author     Keno Hamer
    @version    1.0
    @param      queue         die MQSeries-Queue
    @param      gmo           Optionen (MQGetMessageOptions) fuer das Lesen aus der Queue

  */

  private static void liesRestlicheTeilnachrichten(MQQueue queue, MQGetMessageOptions gmo)
  {

    MQMessage nachricht = new MQMessage();

    try
    {
      // Optionen: Im Gegensatz zum normalen Lesen wird
      // MQC.MQGMO_ALL_MSGS_AVAILABLE nicht gesetzt, um an die unvollständige Gruppe
      // zu kommen, und MQC.MQGMO_CONVERT wird nicht gesetzt, um Konvertierungsfehler
auszuschliessen

      gmo.options = MQC.MQGMO_NO_WAIT | MQC.MQGMO_NO_SYNCPOINT |
MQC.MQGMO_FAIL_IF_QUIESCING
          |  MQC.MQGMO_LOGICAL_ORDER;

      // MQMO_MATCH_GROUP_ID muss gesetzt werden, um die naechste Nachricht der
      // zuletzt gelesenen Gruppe zu erhalten

      gmo.matchOptions = MQC.MQMO_MATCH_GROUP_ID;

      // Hole alle Teilnachrichten, die zur Gruppe gehören, aus der Queue.
      // Die Schleife endet nach Lesen der letzten Teilnachricht (groupStatus ist dann
MQGS_LAST_MSG_IN_GRP)
      do
      {
        // Teilnachricht holen, nachricht wird nicht gespeichert.
        queue.get(nachricht,gmo);

        // printGMO("Teilnachricht", gmo, nachricht);

      } while ( gmo.groupStatus == MQC.MQGS_MSG_IN_GROUP );

    } // Ende try

    catch (Exception e)
    {
      // Fehler werden ignoriert

      // ... aber trotzdem eine kleine Meldung:
      System.out.println("Exception bei liesRestlicheTeilnachrichten " + e);

    } // Ende catch

  }
```