Deutsch

FUJITSU Software

openUTM V7.0

Einsatz von UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen

Benutzerhandbuch

November 2019

Kritik… Anregungen… Korrekturen…

Die Redaktion ist interessiert an Ihren Kommentaren zu diesem Handbuch. Ihre Rückmeldungen helfen uns, die Dokumentation zu optimieren und auf Ihre Wünsche und Bedürfnisse abzustimmen.

Sie können uns Ihre Kommentare per E-Mail an bs2000services@ts.fujitsu.com senden.

Zertifizierte Dokumentation nach DIN EN ISO 9001:2015

Um eine gleichbleibend hohe Qualität und Anwenderfreundlichkeit zu gewährleisten, wurde diese Dokumentation nach den Vorgaben eines Qualitätsmanagementsystems erstellt, welches die Forderungen der DIN EN ISO 9001: 2015 erfüllt.

Copyright und Handelsmarken

Inhaltsverzeichnis

Einsatz von UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen

1 Einleitung

Die IT-Infrastruktur heutiger Unternehmen als Herzstück und Motor des Geschäftes muss den Anforderungen des digitalen Zeitalters gerecht werden. Dabei muss sie mit vermehrten Datenmengen genauso zurechtkommen wie mit verschärften Anforderungen aus dem Umfeld, z.B. Einhaltung von Compliance-Vorgaben. Ebenso muss die Möglichkeit der kurzfristigen Integration weiterer Applikationen gegeben sein. Und alles dies unter dem Gesichtspunkt einer gewährleisteten Sicherheit.
Somit bestehen wesentliche Anforderungen an eine moderne IT-Infrastruktur u.a. aus

- Flexibilität und schier grenzenloser Skalierbarkeit auch für zukünftige Anforderungen
- hohe Robustheit bei höchster Verfügbarkeit
- absoluter Sicherheit in allen Belangen
- Anpassbarkeit an individuelle Bedürfnisse
- Verursachen geringer Kosten

Fujitsu bietet zur Bewältigung dieser Herausforderungen ein umfangreiches Portfolio innovativer Enterprise Hardware, Software und Support Services im Umfeld unserer Enterprise Mainframe Plattformen an und ist damit Ihr

- verlässlicher Service Provider, der Sie langfristig, flexibel und innovativ beim Betrieb der Mainframe-basierten Kernanwendungen Ihres Geschäftes unterstützt,
- optimaler Partner für die gemeinsame Abdeckung der Anforderungen einer Digitalen Transformation und
- langfristiger Partner aufgrund kontinuierlicher Anpassung moderner Schnittstellen, die eine moderne IT Landschaft mit all ihren Anforderungen mit sich bringt.

Mit openUTM stellt Ihnen Fujitsu eine vielfach erprobte und bewährte Lösung aus dem Middleware-Bereich zur Verfügung.

Die High-End-Plattform für Transaktionsverarbeitung openUTM bietet eine Ablaufumgebung, die all diesen Anforderungen moderner unternehmenskritischer Anwendungen gewachsen ist, denn openUTM verbindet alle Standards und Vorteile von transaktionsorientierten Middleware-Plattformen und Message Queuing Systemen:

- Konsistenz der Daten und der Verarbeitung
- Hohe Verfügbarkeit der Anwendungen
- Hohen Durchsatz auch bei großen Benutzerzahlen, d.h. höchste Skalierbarkeit
- Flexibilität bezüglich Änderungen und Anpassungen des IT-Systems

Eine UTM-Anwendung auf Unix-, Linux- und Windows-Systemen kann auf einem einzelnen Rechner als stand-alone UTM-Anwendung oder auf mehreren Rechnern gleichzeitig als UTM-Cluster-Anwendung betrieben werden.

openUTM ist Teil des umfassenden Angebots von **openSEAS**. Gemeinsam mit der Oracle Fusion Middleware bietet openSEAS die komplette Funktionalität für Anwendungsinnovation und moderne Anwendungsentwicklung. Im Rahmen des Produktangebots **openSEAS** nutzen innovative Produkte die ausgereifte Technologie von openUTM:

- BeanConnect ist ein Adapter gemäß der Java EE Connector Architecture (JCA) und bietet den standardisierten Anschluss von UTM-Anwendungen an Java EE Application Server. Dadurch können bewährte Legacy-Anwendungen in neue Geschäftsprozesse integriert werden.
- Bestehende UTM-Anwendungen können unverändert ins Web übernommen werden. Mit dem UTM-HTTP Interface und dem Produkt WebTransactions stehen in openSEAS zwei Alternativen zur Verfügung, welche es ermöglichen, bewährte Host-Anwendungen flexibel in neuen Geschäftsprozessen und modernen Einsatzszenarien zu nutzen.

Die Produkte BeanConnect und WebTransactions werden im Leistungsüberblick kurz dargestellt. Für diese Produkte gibt es eigene Handbücher.

Wenn im Folgenden von Linux-System bzw. Linux-Plattform die Rede ist, dann ist darunter eine Linux-Distribution wie z.B. SUSE oder Red Hat zu verstehen.

Wenn im Folgenden von Windows-System bzw. Windows-Plattform die Rede ist, dann sind damit alle Windows-Varianten gemeint, auf denen openUTM zum Ablauf kommt.

Wenn im Folgenden von Unix-System bzw. Unix-Plattform die Rede ist, dann ist darunter ein Unix-basiertes Betriebssystem wie z.B. Solaris oder HP-UX zu verstehen.

1.1 Zielgruppe und Konzept des Handbuchs

Dieses Handbuch richtet sich an Anwendungsplaner, Anwendungsentwickler, Anwender und Betreuer von UTM-Anwendungen.

Es enthält alle Informationen, um ein UTM-Anwendungsprogramm auf Unix-, Linux- und Windows-Systemen zu erzeugen und eine UTM-Anwendung einzusetzen.

Dieses Handbuch gibt Ihnen in den ersten Kapiteln einen Überblick darüber, wie Sie eine UTM-Anwendung binden und welche Dateien zum Betrieb einer Anwendung notwendig sind. Jeweils eigene Kapitel befassen sich mit dem Starten und Beenden einer UTM-Anwendung und mit dem Programmaustausch bei laufender Anwendung. Die Besonderheiten, die Sie beim Betrieb einer UTM-Cluster-Anwendung bzw. einer UTM-Datenbank-Anwendung beachten müssen, sind zentral in jeweils gleichlautenden Kapiteln zusammengestellt.

Ausführlich wird darauf eingegangen, wie sich Terminal-Benutzer und andere Clients an eine UTM-Anwendung anmelden können.

Zusätzlich gibt es eigene Kapitel über die Tools, die Ihnen für den Betrieb und die Kontrolle einer UTM-Produktivanwendung zur Verfügung stehen.

Kenntnisse des Betriebssystems werden vorausgesetzt.

1.2 Wegweiser durch die Dokumentation zu openUTM

In diesem Abschnitt erhalten Sie einen Überblick über die Handbücher zu openUTM und zum Produktumfeld von openUTM.

1.2.1 openUTM-Dokumentation

Die openUTM-Dokumentation besteht aus Handbüchern, den Online-Hilfen für den grafischen Administrationsarbeitsplatz openUTM WinAdmin und das grafische Administrationstool WebAdmin sowie Freigabemitteilungen.

Es gibt Handbücher und Freigabemitteilungen, die für alle Plattformen gültig sind, sowie Handbücher und Freigabemitteilungen, die jeweils für BS2000-Systeme bzw. für Unix-, Linux- und Windows-Systeme gelten.

Sämtliche Handbücher sind im Internet verfügbar unter der Adresse https://bs2manuals.ts.fujitsu.com. Für die Plattform BS2000 finden Sie die Handbücher auch auf der Softbook-DVD.

Die folgenden Abschnitte geben einen Aufgaben-bezogenen Überblick über die Dokumentation zu openUTM V7.0.

Eine vollständige Liste der Dokumentation zu openUTM finden Sie im Literaturverzeichnis.

Einführung und Überblick

Das Handbuch **Konzepte und Funktionen** gibt einen zusammenhängenden Überblick über die wesentlichen Funktionen, Leistungen und Einsatzmöglichkeiten von openUTM. Es enthält alle Informationen, die Sie zum Planen des UTM-Einsatzes und zum Design einer UTM-Anwendung benötigen. Sie erfahren, was openUTM ist, wie man mit openUTM arbeitet und wie openUTM in die BS2000-, Unix-, Linux- und Windows-Plattformen eingebettet ist.

Programmieren

- Zum Erstellen von Server-Anwendungen über die KDCS-Schnittstelle benötigen Sie das Handbuch **Anwendungen programmieren mit KDCS für COBOL, C und C++**, in dem die KDCS-Schnittstelle in der für COBOL, C und C++ gültigen Form und die Programmschnittstelle UTM-HTTP beschrieben sind. Die KDCS-Schnittstelle umfasst sowohl die Basisfunktionen des universellen Transaktionsmonitors als auch die Aufrufe für verteilte Verarbeitung. Es wird auch die Zusammenarbeit mit Datenbanken beschrieben.
  Die Programm-Schnittstelle UTM-HTTP stellt Funktionen zur Verfügung, die für die Kommunikation mit HTTP-Clients verwendet werden können.
- Wollen Sie die X/Open-Schnittstellen nutzen, benötigen Sie das Handbuch **Anwendungen erstellen mit X/Open-Schnittstellen**. Es enthält die openUTM-spezifischen Ergänzungen zu den X/Open-Programmschnittstellen TX, CPI-C und XATMI sowie Hinweise zu Konfiguration und Betrieb von UTM-Anwendungen, die X/Open-Schnittstellen nutzen. Ergänzend dazu benötigen Sie die X/Open-CAE-Spezifikation für die jeweilige X/Open-Schnittstelle.
- Wenn Sie Daten auf Basis von XML austauschen wollen, benötigen Sie das Dokument **XML für openUTM**. Darin werden die C- und COBOL-Aufrufe beschrieben, die zum Bearbeiten von XML-Dokumenten benötigt werden.
- Für BS2000-Systeme gibt es Ergänzungsbände für die Programmiersprachen Assembler, Fortran, Pascal-XT und PL/1.

Konfigurieren

Zur Definition von Konfigurationen steht Ihnen das Handbuch **Anwendungen generieren** zur Verfügung. Darin ist beschrieben, wie Sie mit Hilfe des UTM-Tools KDCDEF sowohl für eine stand-alone UTM-Anwendung als auch für eine UTM-Cluster-Anwendung auf Unix-, Linux- und Windows-Systemen.

- die Konfiguration definieren,
- die KDCFILE erzeugen,
- und im Falle einer UTM-Cluster-Anwendung die UTM-Cluster-Dateien erzeugen.

Zusätzlich wird gezeigt, wie Sie wichtige Verwaltungs- und Benutzerdaten mit Hilfe des Tools KDCUPD in eine neue KDCFILE übertragen, z.B. beim Umstieg auf eine neue Version von openUTM oder nach Änderungen in der Konfiguration. Für eine UTM-Cluster-Anwendung wird außerdem gezeigt, wie Sie diese Daten mit Hilfe des Tools KDCUPD in die neuen UTM-Cluster-Dateien übertragen.

Binden, Starten und Einsetzen

Um UTM-Anwendungen einsetzen zu können, benötigen Sie für das betreffende Betriebssystem (BS2000- bzw. Unix-, Linux- oder Windows-Systeme) das Handbuch **Einsatz von UTM-Anwendungen**.
Dort ist beschrieben, wie man ein UTM-Anwendungsprogramm bindet und startet, wie man sich bei einer UTM-Anwendung an- und abmeldet und wie man Anwendungsprogramme strukturiert und im laufenden Betrieb austauscht. Außerdem enthält es die UTM-Kommandos, die dem Terminal-Benutzer zur Verfügung stehen. Zudem wird ausführlich auf die Punkte eingegangen, die beim Betrieb von UTM-Cluster-Anwendungen zu beachten sind.

Administrieren und Konfiguration dynamisch ändern

- Für das Administrieren von Anwendungen finden Sie die Beschreibung der Programmschnittstelle zur Administration und die UTM-Administrationskommandos im Handbuch **Anwendungen administrieren**. Es informiert über die Erstellung eigener Administrationsprogramme für den Betrieb einer stand-alone UTM-Anwendung oder einer UTM-Cluster-Anwendung sowie über die Möglichkeiten, mehrere UTM-Anwendungen zentral zu administrieren. Darüber hinaus beschreibt es, wie Sie Message Queues und Drucker mit Hilfe der KDCS-Aufrufe DADM und PADM administrieren können.
- Wenn Sie den grafischen Administrationsarbeitsplatz **openUTM WinAdmin** oder die funktional vergleichbare Web-Anwendung **openUTM WebAdmin** einsetzen, dann steht Ihnen folgende Dokumentation zur Verfügung:
  - Die **WinAdmin-Beschreibung** und die **WebAdmin-Beschreibung** bieten einen umfassenden Überblick über den Funktionsumfang und das Handling von WinAdmin/WebAdmin.
  - Das jeweilige **Online-Hilfesystem** beschreibt kontextsensitiv alle Dialogfelder und die zugehörigen Parameter, die die grafische Oberfläche bietet. Außerdem wird dargestellt, wie man WinAdmin bzw. WebAdmin konfiguriert, um stand-alone UTM-Anwendungen und UTM-Cluster-Anwendungen administrieren zu können.

> **i** Details zur Integration von openUTM WebAdmin in den SE Manager des SE Servers finden Sie im SE Server Handbuch **Bedienen und Verwalten**.

Testen und Fehler diagnostizieren

Für die o.g. Aufgaben benötigen Sie außerdem die Handbücher **Meldungen, Test und Diagnose** (jeweils ein Handbuch für Unix-, Linux- und Windows-Systeme und für BS2000-Systeme). Sie beschreiben das Testen einer UTM-Anwendung, den Inhalt und die Auswertung eines UTM-Dumps, das Meldungswesen von openUTM, sowie alle von openUTM ausgegebenen Meldungen und Returncodes.

openUTM-Clients erstellen

Wenn Sie Client-Anwendungen für die Kommunikation mit UTM-Anwendungen erstellen wollen, stehen Ihnen folgende Handbücher zur Verfügung:

- Das Handbuch **openUTM-Client für Trägersystem UPIC** beschreibt Erstellung und Einsatz von Client-Anwendungen, die auf UPIC basieren. Es zeigt auf, was beim Programmieren einer CPI-C-Anwendung zu beachten ist und welche Einschränkungen es gegenüber der Programmschnittstelle X/Open CPI-C gibt.

- Das Handbuch **openUTM-Client für Trägersystem OpenCPIC** beschreibt, wie man OpenCPIC installiert und konfiguriert. Es zeigt auf, was beim Programmieren einer CPI-C-Anwendung zu beachten ist und welche Einschränkungen es gegenüber der Programmschnittstelle X/Open CPI-C gibt.
- Für das mit **BeanConnect** ausgelieferte Produkt **openUTM-JConnect** existiert neben dem Handbuch eine Java-Dokumentation mit der Beschreibung der Java-Klassen.
- Das Handbuch **BizXML2Cobol** beschreibt, wie Sie bestehende Cobol-Programme einer UTM-Anwendung so erweitern können, dass sie als Standard-Web-Service auf XML-Basis genutzt werden können. Die Arbeit mit der grafischen Bedienoberfläche ist in der zugehörigen **Online-Hilfe** beschrieben.
- Sie können auch mit dem Software-Produkt WS4UTM (WebServices for openUTM) Services von UTM-Anwendungen als Web Services verfügbar machen. Dazu benötigen Sie das Handbuch **Web-Services für openUTM.** Die Arbeit mit der grafischen Bedienoberfläche ist in der zugehörigen **Online-Hilfe** beschrieben.

Kopplung mit der IBM-Welt

Wenn Sie aus Ihrer UTM-Anwendung mit Transaktionssystemen von IBM kommunizieren wollen, benötigen Sie außerdem das Handbuch **Verteilte Transaktionsverarbeitung zwischen openUTM und CICS-, IMS- und LU6.2-Anwendungen**. Es beschreibt die CICS-Kommandos, IMS-Makros und UTM-Aufrufe, die für die Kopplung von UTM-Anwendungen mit CICS- und IMS-Anwendungen benötigt werden. Die Kopplungsmöglichkeiten werden anhand ausführlicher Konfigurations- und Generierungsbeispiele erläutert. Außerdem beschreibt es die Kommunikation über openUTM-LU62, sowie dessen Installation, Generierung und Administration.

Dokumentation zu PCMX

Mit openUTM auf Unix-, Linux- und Windows-Systemen wird die Kommunikationskomponente PCMX ausgeliefert. Die Funktionen von PCMX sind in folgenden Dokumenten beschrieben:

- Handbuch CMX (Unix-Systeme) "Betrieb und Administration" für Unix- und Linux- Systeme
- Online-Hilfe zu PCMX für Windows-Systeme

1.2.2 Dokumentation zum openSEAS-Produktumfeld

Die Verbindung von openUTM zum openSEAS-Produktumfeld wird im openUTM-Handbuch **Konzepte und Funktionen** kurz dargestellt. Die folgenden Abschnitte zeigen, welche der openSEAS-Dokumentationen für openUTM von Bedeutung sind.

Integration von Java EE Application Servern und UTM-Anwendungen

Der Adapter BeanConnect gehört zur Produkt-Suite openSEAS. Der BeanConnect-Adapter realisiert die Verknüpfung zwischen klassischen Transaktionsmonitoren und Java EE Application Servern und ermöglicht damit die effiziente Integration von Legacy-Anwendungen in Java-Anwendungen.

Das Handbuch **BeanConnect** beschreibt das Produkt BeanConnect, das einen JCA 1.5- und JCA 1.6-konformen Adapter bietet, der UTM-Anwendungen mit Anwendungen auf Basis von Java EE, z.B. mit dem Application Server von Oracle, verbindet.

Web-Anbindung und Anwendungsintegration

Anstatt der UTM-HTTP-Programmschnittstelle können Sie alternativ auch das Produkt WebTransactions verwenden. Dann benötigen Sie die Handbücher zu WebTransactions. Die Dokumentation wird durch JavaDocs ergänzt.

1.2.3 Readme-Dateien

Funktionelle Änderungen und Nachträge der aktuellen Produktversion zu diesem Handbuch entnehmen Sie bitte ggf. den Produkt-spezifischen Readme-Dateien.

Readme-Dateien stehen Ihnen online bei dem jeweiligen Produkt zusätzlich zu den Produkthandbüchern unter https://bs2manuals.ts.fujitsu.com zur Verfügung. Für die Plattform BS2000 finden Sie Readme-Dateien auch auf der Softbook-DVD.

Informationen auf BS2000-Systemen

Wenn für eine Produktversion eine Readme-Datei existiert, finden Sie auf BS2000-Systemen die folgende Datei:

`SYSRME.<product>.<version>.<lang>`

Diese Datei enthält eine kurze Information zur Readme-Datei in deutscher oder englischer Sprache (<lang>=D/E). Die Information können Sie am Bildschirm mit dem Kommando `/SHOW-FILE` oder mit einem Editor ansehen. Das Kommando `/SHOW-INSTALLATION-PATH INSTALLATION-UNIT=<product>` zeigt, unter welcher Benutzerkennung die Dateien des Produkts abgelegt sind.

Ergänzende Produkt-Informationen

Aktuelle Informationen, Versions-, Hardware-Abhängigkeiten und Hinweise für Installation und Einsatz einer Produktversion enthält die zugehörige Freigabemitteilung. Solche Freigabemitteilungen finden Sie online unter https://bs2manuals.ts.fujitsu.com.

1.3 Änderungen in openUTM V7.0

Die folgenden Abschnitte gehen näher auf die Änderungen in den einzelnen Funktionsbereichen ein.

1.3.1 Neue Server-Funktionen

UTM als HTTP-Server

Eine UTM-Anwendung kann auch als HTTP-Server fungieren.

Als Methoden werden GET, PUT, POST und DELETE unterstützt. Neben HTTP wird auch der Zugang über HTTPS unterstützt.

Dazu wurden folgende Schnittstellen geändert:

- Generierung

  Alle Systeme:

  - KDCDEF-Anweisung BCAMAPPL:
    - Beim Operand T-PROT= mit Wert SOCKET gibt es eine zusätzliche Angabe zum Transportprotokoll:

      *USP: Auf Verbindungen dieses Zugriffspunktes soll das UTM-Socket-Protokoll verwendet werden.

      *HTTP: Auf Verbindungen dieses Zugriffspunktes soll das HTTP-Protokoll verwendet werden.

      *ANY: Auf Verbindungen dieses Zugriffspunktes werden sowohl das UTM-Socket-Protokoll als auch das HTTP-Protokoll unterstützt.

    - Beim Operand T-PROT= mit Wert SOCKET gibt es zusätzlich die Angabe zur Verschlüsselung:

      SECURE: Auf Verbindungen dieses Zugriffspunktes erfolgt die Kommunikation unter Verwendung von Transport Layer Security (TLS).

    - Neuer Operand USER-AUTH = *NONE | *BASIC. Hiermit kann angegeben werden, welchen Authentisierungsmechanismus HTTP-Clients für diesen Zugangspunkt verwenden müssen.
  - KDCDEF-Anweisung HTTP-DESCRIPTOR:
    Mit dieser Anweisung wird eine Abbildung des in einem HTTP-Request empfangenen Path auf einen TAC definiert und es können zusätzliche Verarbeitungsparameter angegeben werden.

  BS2000-Systeme:

  - KDCDEF-Anweisung CHAR-SET:
    Mit dieser Anweisung können jeder der von openUTM zur Verfügung gestellten vier Code-Konvertierungen von UTM jeweils bis zu vier Character-Set Namen zugeordnet werden.
- Programmierung
  - KDCS- Kommunikationsbereich (KB):
    Im Kopf des KDCS-Kommunikationsbereichs gibt es im Feld *kccp/KCCP* neue Werte für die Client-Protokolle HTTP, USP-SECURE und HTTPS.
  - KDCS-Aufruf INIT PU:
    - Die Version der Schnittstelle wurde auf 7 erhöht.
    - Um die verfügbare Information vollständig zu erhalten, muss im Feld KCLI der Wert 372 angegeben werden.
    - Neue Felder zur Anforderung (KCHTTP/http_info) und Rückgabe (KCHTTPINF/httpInfo) von HTTP-spezifischen Informationen.
- Administrationsschnittstelle KDCADMI
  - Die Datenstrukturversion von KDCADMI wurde auf Version 11 geändert (Feld *version_data* im Parameterbereich).

- Neue Struktur *kc_http_descriptor_str* im Identifikationsbereich für die Unterstützung des HTTP Deskriptors.
- Neue Struktur *kc_character_set_str* im Identifikationsbereich für die Unterstützung des HTTP Charactersets.
- Neue Felder *secure_soc* und *user_auth* in Struktur *kc_bcamappl_str* für die Unterstützung von HTTP Zugangspunkten.

- Programmschnittstelle UTM-HTTP
  Zusätzlich zum KDCS-Interface bietet UTM ein Interface zum Lesen und Schreiben von HTTP Protokollinformationen und zur Behandlung des HTTP Message Bodys.
  Im Folgenden werden die Funktionen des Interface kurz aufgelistet:

- Funktion *kcHttpGetHeaderByIndex()*
  Diese Funktion liefert den Namen und Wert des HTTP-Header-Feldes für den angegebenen Index zurück.
- Funktion *kcHttpGetHeaderByName()*
  Die Funktion liefert den Wert des über den Namen spezifizierten HTTP-Header-Feldes zurück.
- Funktion *kcHttpGetHeaderCount()*
  Diese Funktion liefert die Anzahl der in dem HTTP-Request enthaltenen Header-Felder zurück, die vom Teilprogramm gelesen werden können .
- Funktion *kcHttpGetMethod()*
  Diese Funktion liefert die HTTP-Methode des HTTP-Requests zurück.
- Funktion *kcHttpGetMputMsg()*
  Diese Funktion liefert die vom Teilprogramm erzeugte MPUT-Nachricht zurück.
- Funktion *kcHttpGetPath()*
  Diese Funktion liefert den mit KC_HTTP_NORM_UNRESERVED normierten HTTP- Path des HTTP-Requests zurück.
- Funktion *kcHttpGetQuery()*
  Diese Funktion liefert die mit KC_HTTP_NORM_UNRESERVED normierte HTTP- Query des HTTP -Requests zurück.
- Funktion *kcHttpGetRc2String()*
  Hilfsfunktion um ein Funktionsergebnis vom Typ enum in einen abdruckbaren null-terminierten String umzuwandeln.
- Funktion *kcHttpGetReqMsgBody()*
  Diese Funktion liefert den Message Body des HTTP Requests zurück.
- Funktion *kcHttpGetScheme()*
  Diese Funktion liefert das Schema des HTTP- Requests zurück.
- Funktion *kcHttpGetVersion()*
  Diese Funktion liefert die Version des HTTP- Requests zurück .
- Funktion *kcHttpPercentDecode()*
  Funktion zur Umwandlung von Zeichen in Prozent-Darstellung in Zeichenfolgen in normale Ein-Zeichen-Darstellung.
- Funktion *kcHttpPutHeader()*
  Diese Funktion übergibt einen HTTP-Header für die HTTP-Response .
- Funktion *kcHttpPutMgetMsg()*
  Diese Funktion übergibt eine Nachricht für das Teilprogramm, die mit MGET gelesen werden kann.
- Funktion *kcHttpPutRspMsgBody()*
  Diese Funktion übergibt eine Nachricht für den Message Body der HTTP-Response.
- Function *kcHttpPutStatus()*
  Diese Funktion übergibt einen HTTP-Statuscode für die HTTP-Response.

- Kommunikation über den Secure Socket Layer (SSL)
  BS2000-Systeme:

  - Ist für eine UTM-Anwendung ein BCAMAPPL mit T-PROT=(SOCKET,...,SECURE) generiert, dann wird beim Anwendungsstart von UTM eine zusätzliche Task mit einem Reverse Proxy gestartet, der für die Anwendung als TLS Termination Proxy fungiert und über den sämtliche SSL-Kommunikation abgewickelt wird.

Unix-, Linux- und Windows-Systeme :

- Für einen sicheren Zugang über TLS steht ein weiterer Netzprozess vom Typ *utmnetss/* zur Verfügung. Sind für eine UTM-Anwendung BCAMAPPL mit T-PROT=(SOCKET,...,SECURE) generiert, dann wird beim Anwendungsstart von UTM eine Anzahl von *utmnetss/* Prozessen gestartet. Die Anzahl dieser Prozesse ist abhängig vom Wert LISTENER-ID dieser BCAMAPPL Objekte. In einem *utmnetss/* Prozess wird für die zugeordneten BCAMAPPL Portnummern die gesamte TLS-Kommunikation abgewickelt.

Verschlüsselung

Die Verschlüsselungsfunktionalität in UTM zwischen einer UTM-Anwendung und einem UPIC-Client wurde überarbeitet. Dabei wurden Sicherheitslücken geschlossen, moderne Methoden aufgenommen und die Auslieferung wie folgt vereinfacht:

- UTM-CRYPT Variante
  Bisher stand die Verschlüsselungsfunktionalität in UTM nur zur Verfügung, wenn man das Produkt UTM-CRYPT installiert hatte. Mit UTM V7.0 ist dies nicht mehr erforderlich. Ab dieser Version wird über die Generierung bzw. zum Anwendungsstart entschieden, ob die Verschlüsselungsfunktionalität zum Einsatz kommt oder nicht.
- Security
  Bei der Kommunikation zwischen einer UTM-Anwendung und einem UPIC-Client wurde eine Sicherheitslücke behoben.

  > **!** Das hat zur Folge, dass verschlüsselte Kommunikation einer UTM-Anwendung V7.0 nur zusammen mit UPIC-Client Anwendungen ab UPIC V7.0 möglich ist!

- Verschlüsselung Level 5 (*Unix-, Linux- und Windows-Systeme):*
  KDCDEF-Anweisungen PTERM, TAC und TPOOL
  Beim Operanden ENCRYPTION-LEVEL gibt es einen zusätzlichen Level 5. Dabei wird zur Vereinbarung des Session-Keys das auf Elliptic Curves basierende Diffie-Hellman Verfahren verwendet und Ein-/Ausgabe-Nachrichten werden mit dem AES-GCM Algorithmus verschlüsselt.

OSI-TP Kommunikation und Portnummern

BS2000-Systeme:

- KDCDEF-Anweisung OSI-CON
  Der Operand LISTENER-PORT kann auch auf BS2000-Systemen angegeben werden.
- Administrationsschnittstelle KDCADMI
  In der Struktur *kc_osi_con_str* wird auch auf BS2000-Systemen im Feld *listener-port* die Portnummer angezeigt.

Subnetze

In einer UTM-Anwendung können auch auf BS2000-Systemen Subnetze generiert werden, um den Zugang zu UTM-Anwendungen auf definierte IP-Adressbereiche beschränken zu können. Zusätzlich kann die Namensauflösung per DNS gesteuert werden.

Dazu wurden folgende Schnittstellen geändert:

- Generierung
  BS2000-Systeme:
  - KDCDEF-Anweisung SUBNET:
    Die SUBNET-Anweisung kann auch auf BS2000-Systemen angegeben werden.

  Alle Systeme:
  - KDCDEF-Anweisung SUBNET:
    Mit RESOLVE-NAMES=YES/NO kann angegeben werden, ob nach einem Verbindungsaufbau eine Namensauflösung per DNS stattfinden soll oder nicht.

    Falls eine Namensauflösung erfolgt, dann wird über die Administrationsschnittstelle und in Meldungen der echte Prozessorname des Kommunikationspartners angezeigt. Andernfalls wird als Prozessorname die IP-Adresse der Komunikationspartners sowie der Name des in der Generierung definierten Subnetzes angezeigt.
- Administrationsschnittstelle KDCADMI
  Die Strukturen *kc_subnet_str* und *kc_tpool_str* enthalten ein neues Feld *resolve_names*.

Zugangsdaten für den XA-Datenbank-Anschluss

Ein modifizierter aber noch nicht aktivierter Benutzername für den XA-Datenbank-Anschluss kann per Administration (KDCADMI) gelesen werden:

- Operationscode KC_GET_OBJECT:
  Datenstruktur *kc_db_info_str*: Neues Feld *db_new_userid*.

Reconnect für den XA-Datenbank-Anschluss

Wird bei einer XA Aktion zur Steuerung der Transaktion entdeckt, dass die Verbindung zur Datenbank nicht mehr besteht, wird versucht die Verbindung zu erneuern und die XA Aktion zu wiederholen.

Nur falls dies nicht erfolgreich ist, werden der betroffene UTM Prozess und die UTM-Anwendung abnormal beendet . Bisher wurde bei jedem Verbindungsverlust zur XA Datenbank unmittelbar ohne erneuten Verbindungs-Versuch die UTM-Anwendung abnormal beendet.

Sonstige Änderungen

- XA-Meldungen
  Die Meldungen bzgl. der XA-Schnittstelle wurden jeweils um die Inserts UTM-Userid und TAC erweitert. Betroffen sind die Meldungen K204-K207, K212-K215 und K217-K218.
- UTM-Tool KDCEVAL
  Im TRACE 2 Satz von KDCEVAL wurde im WAITEND Record der Typ des letzten Auftrags (Börsen-Announcements) aufgenommen (ersten beiden Bytes abdruckbar).

1.3.2 Entfallene Server-Funktionen

Im Einzelnen wurden folgende Funktionen gestrichen:

- Dienstprogramm KDCDEF
  Mehrere Funktionen wurden gestrichen und können nicht mehr in KDCDEF generiert werden. Wenn sie dennoch angegeben werden, wird dies im KDCDEF-Lauf mit einem Syntaxfehler abgelehnt.
  - KDCDEF-Anweisung PTERM
    Operanden-Werte 1 und 2 für ENCRYPTION-LEVEL
  - KDCDEF-Anweisung TPOOL
    Operanden-Werte 1 und 2 für ENCRYPTION-LEVEL
  - KDCDEF-Anweisung TAC
    Operanden-Wert 1 für ENCRYPTION-LEVEL
- *BS2000-Systeme*
  - UTM-Cluster:
    Auf BS2000-Systemen werden UTM-Cluster-Anwendungen nicht mehr unterstützt.
- *Unix-, Linux- und Windows-Systeme*
  - TNS Betrieb:
    Beim Start einer UTM-Anwendung wird die TNS-Generierung nicht mehr gelesen. Die Adressierungsinformation muss vollständig bei der Konfiguration mit KDCDEF hinterlegt werden.

1.3.3 Neue Client-Funktionen

Verschlüsselung

Die Verschlüsselungsfunktionalität in openUTM-Client wurde überarbeitet. Dabei wurden Sicherheitslücken geschlossen, moderne Methoden aufgenommen und die Auslieferung wie folgt vereinfacht:

- UTM-CLIENT-CRYPT Variante
  Bisher stand die Verschlüsselungsfunktionalität in openUTM-Client nur zur Verfügung, wenn man das Produkt UTM-CLIENT-CRYPT installiert hatte. Mit openUTM-Client V7.0 ist dies nicht mehr erforderlich. Ab dieser Version wird zum Ablaufzeitpunkt entschieden ob die Verschlüsselungsfunktionalität zum Einsatz kommt oder nicht.
- Security
  Bei der Kommunikation mit einer UTM-Anwendung wurde eine Sicherheitslücke behoben.
- Verschlüsselung Level 5
  openUTM-Client V7.0 unterstützt die Kommunikation mit UTM V7.0 Anwendungen, bei denen für die Verbindungen zum UPIC-Client ENCRYPTION-LEVEL 5 generiert wurde.
  Bei Level 5 wird zur Vereinbarung des Session-Keys das auf Elliptic Curves basierende Diffie-Hellman Verfahren verwendet und Ein-/Ausgabe-Nachrichten werden mit dem AES-GCM Algorithmus verschlüsselt. AES-GCM unterstützt die Authentifikation und die Verschlüsselung von Nachrichten.

  Der Level 5 wird von openUTM-Client auf allen Plattformen unterstützt.
- Verschlüsselung BS2000
  openUTM-Client (BS2000) verwendet analog zu Unix-, Linux- und Windows-Systemen openSSL anstatt BS2000-CRYPT.

1.3.4 Neue Funktionen für openUTM WinAdmin

WinAdmin unterstützt alle Neuerungen der openUTM V7.0 bzgl. der Programmschnittstelle zur Administration.

1.3.5 Neue Funktionen für openUTM WebAdmin

WebAdmin unterstützt alle Neuerungen der openUTM V7.0 bzgl. der Programmschnittstelle zur Administration.

1.4 Darstellungsmittel

Metasyntax

Die in diesem Handbuch verwendete Metasyntax können Sie der folgenden Tabelle entnehmen:

Formale Darstellung	Erläuterung	Beispiel
GROSSBUCHSTABEN	Großbuchstaben bezeichnen Konstanten (Namen von Aufrufen, Anweisungen, Feldnamen, Kommandos und Operanden etc.), die in dieser Form anzugeben sind.	`LOAD-MODE=STARTUP`
kleinbuchstaben	In Kleinbuchstaben sind in Syntaxdiagrammen und Operandenbeschreibung die Platzhalter für Operandenwerte dargestellt.	`KDCFILE=filebase`
kleinbuchstaben	Im Fließtext werden Variablen sowie Namen von Datenstrukturen und Feldern in kursiven Kleinbuchstaben dargestellt.	*utm-installationsverzeichnis* ist das UTM-Installationsverzeichnis
`Schreibmaschinenschrift`	In Schreibmaschinenschrift werden im Fließtext Kommandos, Dateinamen, Meldungen und Beispiele ausgezeichnet, die in genau dieser Form eingegeben werden müssen bzw. die genau diesen Namen oder diese Form besitzen.	Der Aufruf `tpcall`
{ } und |	In geschweiften Klammern stehen alternative Angaben, von denen Sie eine auswählen müssen. Die zur Verfügung stehenden Alternativen werden jeweils durch einen Strich getrennt aufgelistet.	`STATUS={ ON | OFF }`
[ ]	In eckigen Klammern stehen wahlfreie Angaben, die entfallen können.	`KDCFILE=( filebase [, { SINGLE | DOUBLE } ] )`
( )	Kann für einen Operanden eine Liste von Parametern angegeben werden, sind diese in runde Klammern einzuschließen und durch Kommata zu trennen. Wird nur ein Parameter angegeben, kann auf die Klammern verzichtet werden.	`KEYS=(key1, key2,...key`*n*`)`
Unterstreichen	Unterstreichen kennzeichnet den Standardwert.	`CONNECT= { YES | NO }`
Kurzform	Die Standardkurzform für Anweisungen, Operanden und Operandenwerte wird „fett“ hervorgehoben. Die Kurzform kann alternativ angegeben werden.	`TRANSPORT -SELECTOR =c‘C‘`

Formale Darstellung	Erläuterung	Beispiel

`...`	Punkte zeigen die Wiederholbarkeit einer syntaktischen Einheit an.
	Außerdem kennzeichnen die Punkte Ausschnitte aus einem Programm, einer Syntaxbeschreibung o.ä.
		`KDCDEF starten`
		`...`
		`OPTION`
		`DATA=statement_file`
		`...`
		`END`

Symbole

 für Verweise auf umfassende und detaillierte Informationen zum jeweiligen Thema.

 für Hinweistexte.

 für Warnhinweise.

Sonstiges

utmpfad bezeichnet auf Unix-, Linux- und Windows-Systemen das Verzeichnis, unter dem openUTM installiert wurde.

filebase bezeichnet auf Unix-, Linux- und Windows-Systemen das Dateiverzeichnis der UTM-Anwendung. Dies ist der Basisname, der in der KDCDEF-Anweisung MAX KDCFILE= generiert wurde.

$userid bezeichnet auf BS2000-Systemen die Kennung, unter der openUTM installiert wurde.

upic-dir bezeichnet auf Unix-, Linux- und Windows-Systemen das Verzeichnis, unter dem openUTM-Client für Trägersystem UPIC installiert ist.

2 Anwendungsprogramm erzeugen

Die Teilprogramme definieren die Anwendungslogik und müssen vor dem Start der Anwendung geschrieben und übersetzt werden. Sehen Sie dazu das openUTM-Handbuch „Anwendungen programmieren mit KDCS“.

Damit die Teilprogramme unter openUTM ablaufen können, wird das UTM-Anwendungsprogramm wie folgt erzeugt:

- ROOT-Tabellen-Source, die KDCDEF erzeugt hat, übersetzen.
- ROOT-Tabellen, UTM-Main Routine, UTM-Systemmodule für die Main Routine KDCROOT, C-Laufzeitsystem und eventuell weitere Laufzeitsysteme, Meldungsmodul, Benutzerbibliotheken und Teilprogramme binden.

Die Teilprogramme können Sie auch als Shared Objects binden. Shared Objects können im laufenden Betrieb dynamisch ausgetauscht werden.

Informationen zum Programmaustausch mit Shared Objects finden Sie im Kapitel„Programmaustausch im Betrieb“.

Die einzelnen Schritte, die dabei zum Erzeugen eines UTM-Anwendungsprogramms notwendig sind, zeigt die folgende Abbildung.

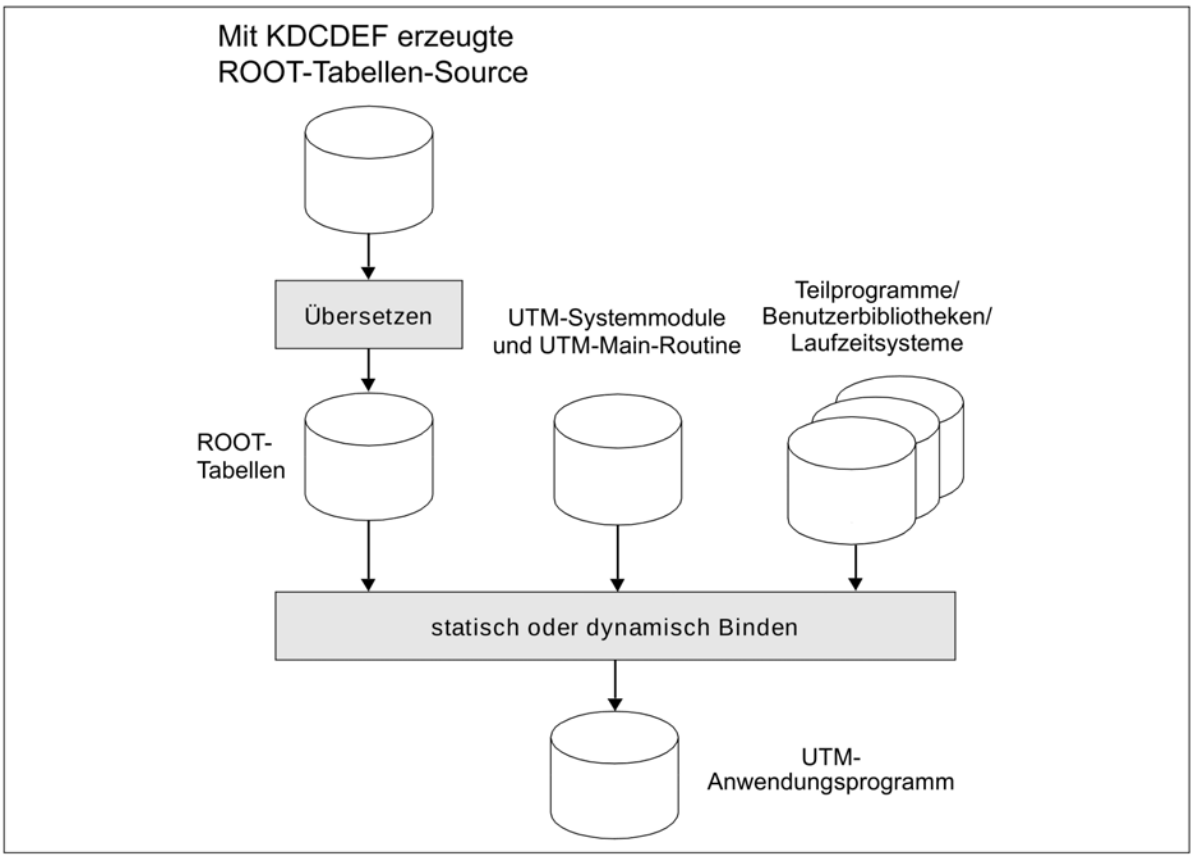


Bild 1: Übersicht: Erzeugen des UTM-Anwendungsprogramms

Main Routine KDCROOT

Aus den ROOT-Tabellen, der UTM-Main-Routine, die bei der Installation von openUTM erzeugt wird, und den UTM-Systemfunktionen, entsteht beim Binden die Main Routine KDCROOT als Teil des Anwendungsprogramms. KDCROOT fungiert beim Ablauf der Anwendung als steuerndes Hauptprogramm und übernimmt u. a. folgende Aufgaben:

- Verbindung zwischen den Teilprogrammen und den UTM-Systemfunktionen
- Ablauf-Koordination von Teilprogrammen unterschiedlicher Programmiersprachen
- Kopplung zu Datenbanken

Weiter enthält KDCROOT Bereiche für variable Daten sowie die Nachrichtenbereiche. Nähere Informationen zur Main Routine KDCROOT sind dem openUTM-Handbuch „Anwendungen programmieren mit KDCS“ zu entnehmen.

KDCDEF erzeugt das ROOT-Tabellenmodul als C/C++-Source, die Sie mit dem C/C++-Compiler übersetzen und mit Ihren Teilprogrammen, den UTM-Systemmodulen sowie eventuell weiteren Modulen zu einem ablauffähigen Programm binden, siehe unten.

2.1 UTM-Prozess binden auf Unix- und Linux-Systemen

Für eine UTM-Anwendung müssen Sie die unten aufgelisteten UTM-Systembibliotheken und UTM-Objekte in der angegebenen Reihenfolge zu einem Workprozess binden, damit alle Externverweise aufgelöst werden können. Das gebundene Anwendungsprogramm (Workprozess) muss im Verzeichnis *filebase* unter dem Namen *utmwork* abgelegt werden.

Die benötigten UTM-Systembibliotheken und UTM-Objekte finden Sie auf Ihrem Rechner unter *utmpfad/sys*. Auf der Plattform AIX werden die openUTM-Systembibliotheken als statische Bibliotheken zur Verfügung gestellt, auf allen anderen Plattformen als Shared Objects.

Wenn Sie die Funktion „Programmaustausch“ nutzen wollen, gibt es zwei Möglichkeiten:

- Wenn Sie Teile des Anwendungsprogramms austauschen möchten, müssen Sie diese Teile in einem Shared Object ablegen.
- Wenn nur der komplette Workprozess austauschbar sein soll, müssen Sie nichts beachten. Nach dem Binden müssen Sie *utmwork* mit Hilfe des Tools KDCPROG in das Dateigenerationsverzeichnis *filebase*/PROG bringen.

Details zum Übersetzen der Teilprogramme, zum Erzeugen von Shared Objects und zum Binden des Anwendungsprogramms finden Sie in der Dokumentation des verwendeten Compilers bzw. Laufzeitsystems.

Beispielprozeduren zum Übersetzen und Binden finden Sie bei der ausgelieferten Beispielanwendung. Sie können sich die Arbeit erleichtern, wenn Sie mit Hilfe der ausgelieferten Beispielanwendung ein Makefile erzeugen, das Sie als Grundlage für Ihre Bindeprozedur verwenden, siehe „Binden mit Makefile (Unix- und Linux-Systeme)“.

2.1.1 COBOL-Teilprogramme (Unix- und Linux-Systeme)

COBOL-Programme können Sie mit den Compilern von Micro Focus oder dem Compiler NetCOBOL von Fujitsu erstellen.

Bitte beachten Sie die COBOL-Compiler-spezifischen Programmierhinweise im openUTM-Handbuch „Anwendungen programmieren mit KDCS“ (Kapitel Ergänzungen für COBOL, Abschnitt „Plattform-spezifische Besonderheiten auf Unix- und Linux-Systemen“).

Umgebungsvariable für COBOL-Programme

Wenn Sie COBOL-Programme einsetzen müssen Sie Compiler-spezifische Umgebungsvariablen setzen.
Micro Focus COBOL

Wenn Sie COBOL-Teilprogramme mit Micro Focus COBOL verwenden, führen Sie folgende Schritte durch:

- Rufen Sie das Skript `<coboldir>/bin/cobsetenv` auf. Dieses Skript setzt die notwendigen Umgebungsvariablen für den Compiler.
- Erweitern Sie die Umgebungsvariable COBCPY um `$UTMPATH/copy-cobol85`.
- Falls Sie Programme auf Basis von CPIC, TX bzw. XATMI unter openUTM erstellen, erweitern Sie die Umgebungsvariable COBCPY um `$UTMPATH/<interface>/copy-cobol85`, wobei <interface> für `cpic`, `tx` bzw. `xatmi` steht.
- Falls Sie Client-Programme auf Basis von UPIC-L erstellen, erweitern Sie die Umgebungsvariable COBCPY um `$UTMPATH/upicl/copy-cobol85`.
- Setzen Sie die Umgebungsvariable COBMODE auf 64 um 64-Bit-Objekte zu erzeugen.

NetCOBOL

Wenn Sie NetCOBOL-Teilprogramme verwenden, führen Sie folgende Schritte durch:

- Rufen Sie das Skript `<COBOLDIR>/config/cobol.sh` auf. Dieses Skript setzt die notwendigen Umgebungsvariablen.
- Erweitern Sie die Umgebungsvariable COBCOPY um `$UTMPATH/netcobol`.
- Setzen Sie die Umgebungsvariable COB_LIBSUFFIX auf `None,CPY,cpy`.
- Falls Sie Programme auf Basis von CPIC, TX bzw. XATMI unter openUTM erstellen, erweitern Sie die Umgebungsvariable COBCOPY um `$UTMPATH/<interface>/netcobol`, wobei <interface> für `cpic`, `tx` bzw. `xatmi` steht.
- Falls Sie Client-Programme auf Basis von UPIC-L erstellen, erweitern Sie die Umgebungsvariable COBCOPY um `$UTMPATH/upicl/netcobol`.

Hinweis

Wenn Sie COBOL-Teilprogramme verwenden, dann beachten Sie bitte:

- Es gibt Compiler-spezifische Besonderheiten zu Schlüsselwörtern. Details finden Sie im openUTM-Handbuch „Anwendungen programmieren mit KDCS“ (Kapitel „Ergänzungen für COBOL“, Stichwort „Schlüsselwörter“).

- Wenn Sie Shared Objects erzeugen möchten, lesen Sie bitte den Abschnitt „SharedObjects (Unix- und Linux-Systeme)“.

2.1.2 Benötigte UTM-Systembibliotheken und UTM-Objekte (Unix- und Linux-Systeme)

Die im Folgenden aufgeführen Bibliotheken und Objekte können beim Binden einer Anwendung angegeben werden. Welche dieser Komponenten zum Ablauf notwendig sind (d.h. mit eingebunden werden müssen), ist abhängig von der Konfiguration der jeweiligen UTM-Anwendung.

- Die Main Routine von openUTM

  Hier müssen Sie entweder das Objekt `mainutm.o` einbinden, falls nur C- und COBOL-Programme verwendet werden oder das Objekt `mainutmCC.o`, falls mindestens ein C++-Programm verwendet wird.

  Auf den meisten Systemen darf die Anwendung nicht gleichzeitig C++- und COBOL-Programme enthalten.

  Die Objekte `mainutm.o` und `mainutmCC.o` stehen in *utmpfad* `/sys`. Das Objekt `mainutmCC.o` wird während der Installation von openUTM erzeugt, wenn der C++-Compiler bereits installiert ist.

  Wird der C++-Compiler erst nach openUTM installiert oder konnte openUTM den C++-Compiler bei der Installation aus irgendwelchen anderen Gründen nicht finden, dann können Sie `mainutmCC.o` mit Hilfe des UTM-Shellskripts `CCmainutm` selbst erzeugen.

  Dazu müssen Sie folgendes Kommando eingeben:

  `UTMPATH=` *utmpfad* `\`
  utmpfad `/shsc/CCmainutm`

  Näheres zum *utmpfad* siehe "UTM-Systemfunktionen auf Unix- und Linux-Systemen installieren".

- Die Main Routine KDCROOT (`rootname.o`)

  Dieses Modul wird in jeder Anwendung benötigt und aus dem ROOT-Quellprogramm erzeugt. Das ROOT-Quellprogramm (*filebase* `/` *rootname* `.c`) wird beim KDCDEF-Lauf erzeugt , wenn Sie in der OPTION-Anweisung GEN=ROOTSRC oder GEN=ALL angeben. *rootname* definieren Sie in der ROOT-Anweisung und *filebase* in der MAX-Anweisung mit dem Parameter KDCFILE=.

  Sie erzeugen KDCROOT wie folgt:

  - Übersetzen Sie das mit KDCDEF erstellte ROOT-Quellprogramm mit dem C-Compiler. Für die openUTM-System-Includes müssen Sie hier den Schalter `-I$UTMPATH/include` angeben.

    Wenn Sie COBOL-Teilprogramme verwenden, müssen Sie zusätzlich den Schalter `-I$COBDIR/include` angeben. `$COBDIR` ist der Installationspfad des COBOL-Compilers.

  - Speichern Sie die Ausgabe unter *filebase/rootname* `.o`.

Sie müssen *filebase*/*rootname*.o beim Binden der Anwendung **vor** den anwendungsspezifischen Modulen angeben.

- Ihre übersetzten Teilprogramme

  Ihre übersetzten Teilprogramme können Sie entweder als einzelne Objekte oder als Teile von Objektbibliotheken einbinden. Dabei kann es sich um statische Bibliotheken oder um dynamische Bibliotheken (genannt Shared Objects) handeln.

- Die UTM-Systembibliotheken einschließlich Administrationsprogramm und Code-Konvertierungstabellen

  Die Bibliothek `libwork` wird in jedem Anwendungsprogramm benötigt. Zusätzlich gilt:

  - Auf allen Unix-/Linux-Plattformen außer AIX wird die Bibliothek `libutmconvt` beim Starten der Anwendung automatisch nachgeladen. Dazu muss
    LD_LIBRARY_PATH bzw. LD_LIBRARY_PATH_64 auf *utmpfad* `/sys` gesetzt sein.
  - Auf der Plattform AIX muss die Bibliothek `libutmconvt.so` beim Binden von *utmwork* angegeben werden.

- OSI TP-Module (optional)

  Falls Kommunikation über OSI TP betrieben werden soll, muss die Bibliothek `libxaptp` eingebunden werden.

  Zusätzlich werden die OSS-Bibliotheken `libossutm` und `liboss` benötigt. Dabei müssen Sie `libossutm` immer **vor** `liboss` angeben, sonst kommt es zu folgendem Fehler beim Start der Anwendung:

  ```
  P001 Fehler beim OSS Aufruf (o_create() ·): - 1, 300, 199, 0
  K060 Der Anwendungslauf wurde abgebrochen; die Ursache ist XINI06.
  ```

- Die Laufzeitsysteme für C und evtl. COBOL

  Das Laufzeitsystem für die Programmiersprache C wird immer benötigt. Das COBOL-Laufzeitsystem wird nur benötigt, wenn Sie ein Teilprogramm in COBOL erstellt haben.

- Ein benutzereigenes Meldungsmodul (optional)

  Wenn Sie ein eigenes Meldungsmodul erstellt haben (siehe openUTM-Handbuch „Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen“), müssen Sie dieses Objektmodul zusätzlich einbinden.

- Das Laufzeitsystem für das Datenbanksystem (optional)

  Welche Bibliotheken und Module Sie für eine UTM-Datenbankkopplung einbinden müssen, ist in Abschnitt „UTM-Datenbank-Anwendung binden auf Unix- und Linux-Systemen“ beschrieben.

- Weitere Bibliotheken für XML (optional)

  Falls Sie in Teilprogrammen die XML-Funktionen nutzen möchten, benötigen Sie die entsprechende XML-Bibliothek.

  Näheres siehe Dokumentation zur XML-Schnittstelle.

- Weitere Bibliotheken für X/Open (optional)

  Falls Sie Teilprogramme mit den X/Open-Schnittstellen erstellt haben, benötigen Sie die UTM-Systembibliothek `libxopen`.

  Zusätzlich muss beim Binden die Option `-lm` angegeben werden.

- Auf Linux müssen Sie die Option `-lcrypt` angeben.
- Die Option `-ldl` müssen Sie immer angeben.
- Soll ihre Anwendung in einer 64-Bit-Umgebung laufen, beachten Sie beim Übersetzen ihrer Teilprogramme und des ROOT-Moduls, dass der Compiler 64-Bit-Objekte erzeugt. Das erreichen Sie, indem geeignete Compiler Schalter gesetzt werden (siehe Dokumentation zum jeweiligen Compiler).

2.1.3 Shared Objects (Unix- und Linux-Systeme)

Beim Binden von *utmwork* mit Shared Objects müssen Sie Folgendes beachten:

COBOL-Teilprogramme

Wenn COBOL-Teilprogramme als Shared Objects dynamisch nachgeladen werden, dann muss das COBOL-Laufzeitsystem ebenfalls als Shared Object in das Teilprogramm eingebunden werden, da das Teilprogramm sonst die Entries des Laufzeitsystems nicht finden kann.

Micro Focus COBOL

Shared Objects erzeugen Sie mit folgendem Aufruf:

`cob -z -o` *`shared-object Cobol-objects`*

Die Reihenfolge der Schalter und der angegebenen Objekte muss unbedingt eingehalten werden.

NetCOBOL

Shared Objects erzeugen Sie mit folgendem Aufruf:

`cobol -shared -dy -o` *`shared-object Cobol-objects`*

Die Reihenfolge der Schalter und der angegebenen Objekte muss unbedingt eingehalten werden.

Plattform AIX

Auf der Plattform AIX wird die Verwendung von Shared Objects von openUTM nicht unterstützt.

2.1.4 Binder aufrufen (Unix- und Linux-Systeme)

Je nachdem, in welcher Sprache die Teilprogramme erstellt wurden, müssen Sie folgenden Binder verwenden:

- den C-Binder `cc`, falls nur C-Programme enthalten sind,
- den COBOL-Binder von Micro Focus oder NetCOBOL, falls mindestens ein COBOL-Programm enthalten ist.
  - Der Aufruf des Micro Focus Binders lautet:

    `cob -o utmwork` *shared-object UTM-Systembibliotheken*
  - Das Binden mit NetCOBOL wird mit dem Kommando `cobol` ausgeführt. Dabei müssen folgende COBOL-Bibliotheken eingebunden werden:
    - `/opt/FJSVcbl64/lib/libFJBASE.so`
    - `/opt/FJSVcbl64/lib/libcobol.so`
    - `/opt/FJSVcbl64/lib/librcobflm64.so`
- Wenn Sie Shared Objects in Ihrer Anwendung einsetzen, müssen Sie immer dynamisch binden.
- den C++-Binder CC, falls mindestens ein C++-Programm enthalten ist. COBOL-Programme dürfen dann nicht vorhanden sein.
- Falls Sie eine Anwendung für einen 64 Bit Betrieb binden wollen, beachten Sie, dass alle Komponenten im 64 Bit Mode vorliegen, die Sie beim Binden angeben. Ein Mischbetrieb von Objekten im 32 Bit Mode und im 64 Bit Mode ist nicht möglich.

2.1.5 Binden mit Makefile (Unix- und Linux-Systeme)

Für Ihr Anwendungsprogramm können Sie ein Makefile erstellen, indem Sie das Makefile der Beispielanwendung, die zusammen mit openUTM ausgeliefert wird, Anwendungsspezifisch abändern.

Die Beispielanwendung finden Sie nach der Installation im CPIO-Archiv *utm-dateiverzeichnis*`/CPIO.utmsample`. Sie können die Beispielanwendung mit der Prozedur *utm-dateiverzeichnis*`/shsc/install.sample` in der Benutzerkennung installieren, unter der auch Ihre UTM-Anwendung ablaufen soll. Erstellen Sie mit der Prozedur `p /config` eine Konfiguration der Beispielanwendung, die Ihrer Anwendung entspricht. Das Makefile `makefile` dieser Beispielanwendung können Sie dann für Ihre Anwendung ändern und erweitern, siehe Online-Dokumentation der Beispielanwendung.

Wenn Sie mit einer Datenbank koppeln möchten, geben Sie beim `p/config`-Lauf das verwendete Datenbanksystem und die benötigte Konfiguration an. Das von `p/config` erzeugte Makefile enthält dann i.a. alle benötigten Module und Bibliotheken für die Datenbankkopplung. Die Bibliothekslisten werden unter `.liblists /ORACLE` bzw. `.liblists/ORACLECOB` erzeugt.

Näheres zum Binden einer UTM-Datenbank-Anwendung finden Sie in Kapitel „UTM-Datenbank-Anwendung“.

2.2 Anwendungsprogramm erzeugen auf Windows-Systemen

Auf Windows-Systemen können Sie Anwendungsprogramme in C, C++ oder in COBOL erstellen.

2.2.1 Anwendungsprogramme in C und C++ (Windows-Systeme)

Für Anwendungsprogramme in C und C++ müssen Sie mit dem Microsoft Visual Studio arbeiten. Dazu sind folgende Schritte nötig:

1. Optionen des Visual Studios einstellen
2. Projekt erzeugen für die Anwendung
3. Programme schreiben oder vorhandene Programme modifizieren
4. Anwendung übersetzen und binden
5. Anwendung als Dienst installieren, falls gewünscht.
   Dies hat u.a. den Vorteil, dass die Anwendung automatisch mit dem System gestartet und beendet werden kann.
   Der Dienst kann bei Bedarf auch manuell gestartet und beendet werden.

Die Abschnitte „Optionen des Visual Studios einstellen (Windows-Systeme)“ bis „Anwendung übersetzen und binden (Windows-Systeme)“ beschreiben, wie Sie ein statisch gebundenes Anwendungsprogramm erzeugen.

Wenn Sie Anwendungsprogramme dynamisch nachladen wollen, müssen Sie diese als DLLs erstellen, siehe Abschnitt „Anwendungsprogramme als DLLs erstellen (Windows-Systeme)“.

Was Sie für Anwendungsprogramm in COBOL beachten müssen, ist im Abschnitt„COBOL-Anwendungsprogramme auf Windows-Systemen“ beschrieben.

> ¡ Die folgende Beschreibung gilt für das Visual Studio Version 2010 (englische Variante).

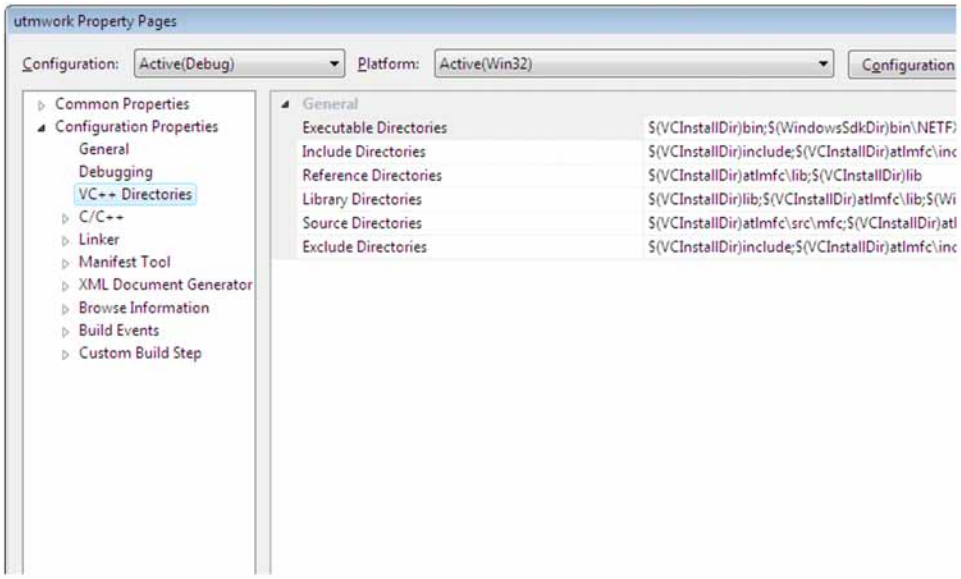
2.2.1.1 Optionen des Visual Studios einstellen (Windows-Systeme)

Bevor Sie UTM-Anwendungen zu entwickeln beginnen, müssen Sie die Verzeichnisse mit den openUTM Include Files (*utmpfad* `\include`), den openUTM Library Files (*utmpfad* `\sys`) und ggf. den Datenbank-Bibliotheken in die Entwicklungsumgebung aufnehmen.

Dies stellen Sie über bestimmte Optionen des Developer Studios ein. Diese Einstellungen sind projektunabhängig und gelten daher für die Erstellung unterschiedlicher UTM-Anwendungen.

Gehen Sie wie folgt vor:

1. Rufen Sie das Microsoft Visual Studio auf, wählen Sie *Tools - Options* und klicken Sie auf *Projects and Solutions.*
2. Markieren Sie *VC++ Directories* und setzen Sie wie folgt die Option für die Include-Dateien:
   - Wählen Sie in der Box *General* den Wert *Include Directories* aus und doppelklicken Sie in die leere Eingabezeile.



   - Klicken Sie auf die Schaltfläche

     New Line und geben Sie das Verzeichnis mit den Include-Dateien ein:
     utmpfad `\include`,
     Standard ist `C:\openUTM-Server\64\include.`

     Alternative: Klicken Sie auf die Schaltfläche „ ... “ und wählen Sie das Verzeichnis im nachfolgenden Zwischendialog aus.

   - Bringen Sie das Verzeichnis mit dem Pfeilbutton an die oberste Stelle.
3. Setzen Sie die Optionen für die UTM-Bibliotheken und die Datenbank-Bibliotheken.

   Dazu wählen Sie in der Box *Show Directories for* den Wert *Library files* aus und gehen analog wie bei den Include-Dateien vor:

   a. Verzeichnis mit den UTM-Bibliothekdateien angeben. Diese Option müssen Sie immer setzen:
      - Doppelklicken Sie auf die leere Eingabezeile

- Geben Sie das Verzeichnis mit den UTM-Bibliothekdateien ein (*utmpfad* `\sys`),
  Standard ist `C:\openUTM-Server\64\sys`.
  Alternative: Klicken Sie auf die Schaltfläche „ ... “ und wählen Sie das Verzeichnis im nachfolgenden Zwischendialog aus.
- Bringen Sie das Verzeichnis mit den Pfeil-Buttons an die erste Stelle,

b. Verzeichnis mit den Datenbanken-Bibliothekdateien angeben. Dies ist nur dann notwendig, wenn Sie eine Datenbank (z.B. Oracle) anschließen wollen:

- Doppelklicken Sie auf die leere Eingabezeile
- Geben Sie das Verzeichnis mit den Datenbanken-Bibliothekdateien direkt ein oder wählen Sie es über den Zwischendialog aus (doppelklicken auf „ ... “).

4. Drücken Sie jetzt im Fenster *Options* auf OK.
   Damit sind die Optionen für Include-Dateien und Bibliothekdateien gespeichert.

2.2.1.2 Projekt erzeugen (Windows-Systeme)

Gehen Sie wie folgt vor:

1. Rufen Sie das Visual Studio auf und wählen Sie in der Menüleiste den Punkt *File*, klicken Sie auf *New* und wählen das Blatt *Projects* aus.
2. Markieren Sie *Win32 Console Application* und schreiben Sie in das Feld *Project name* den Namen Ihres Projekts; der Name ist frei wählbar und wird im Folgenden mit *utmproject* bezeichnet.
3. Tragen Sie bei *Location* das Verzeichnis ein, in dem Ihr Projekt mitsamt allen Dateien abgelegt werden soll. Wenn Sie hier ein Verzeichnis eintragen, das noch nicht existiert, dann wird es neu angelegt.
4. Klicken Sie OK. Damit wird das Projekt erzeugt, es bleibt dabei geöffnet.

Projekt später öffnen

Wenn Sie das Projekt später öffnen möchten, dann haben Sie zwei Möglichkeiten:

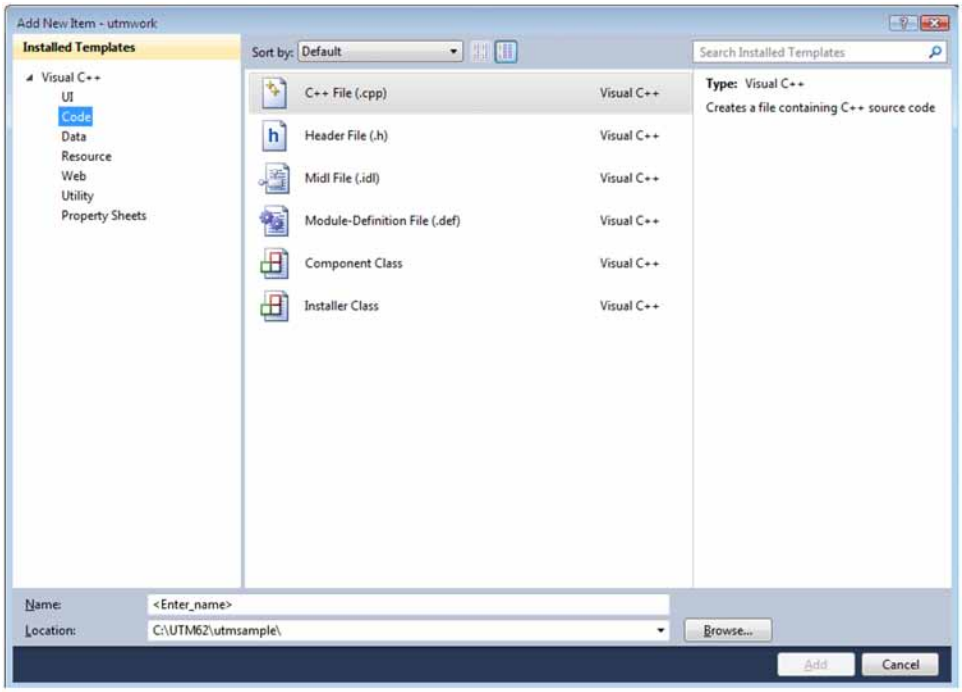
- Über das Visual Studio, indem Sie *File - Open - Project/Solution* klicken und - falls noch nötig - in das Anwendungsverzeichnis navigieren. Dort klicken Sie auf *utmproject*`.sln` (*utmproject* = Name Ihres Projekts).
- Über den Explorer, indem Sie auf die Datei *utmproject*`.sln` doppelklicken.

2.2.1.3 Quellprogramme schreiben (Windows-Systeme)

Wenn Sie neue Quellprogramme in C oder C++ erstellen möchten, dann verwenden Sie am besten den syntaxsensitiven Editor des Visual Studios. Sie können Ihre Programme natürlich auch mit einem beliebigem ASCII-Editor erstellen oder vorhandene Programme damit modifizieren.

Wenn Sie Programme mit Visual C++ erstellen, gehen Sie wie folgt vor:

1. Öffnen Sie Ihr Projekt, indem Sie z.B. im Explorer auf die Datei *utmproject*`.sln` doppelklicken. Damit wird das Visual Studios gestartet.
2. Wählen Sie in der Menüleiste den Punkt *Project* und klicken Sie auf *Add New Item*.



3. Markieren Sie im Bereich *Installed Templates* den Punkt *Code* und im mittleren Bereichden Punkt *C++ File (.cpp)* und tragen Sie unten bei *Name* den Namen Ihres Quellprogramms ein.
4. Klicken Sie *Add*. Damit wird diese Datei angelegt und automatisch in das Projekt aufgenommen. Es wird das Editier-Fenster geöffnet.
5. Erstellen Sie im Editier-Fenster den Quellcode.
6. Schließen Sie die Datei mit *File - Close* und speichern Sie dabei die Änderungen.

2.2.1.4 Anwendung übersetzen und binden (Windows-Systeme)

Voraussetzungen

Bevor Sie die Anwendung binden können, müssen Sie alle benötigten Quellprogramme und Objektdateien in Ihr Projekt aufnehmen und die Bindeoptionen einstellen, siehe unten.

Mit openUTM werden die Dateien `mainutm.c` (wenn keine C++-Programme einzubinden sind) und `mainutm.cpp` (wenn C++-Programme einzubinden sind) ausgeliefert. Diese Sourcen müssen übersetzt und die entstandenen Objekte mit in die Anwendung eingebunden werden.

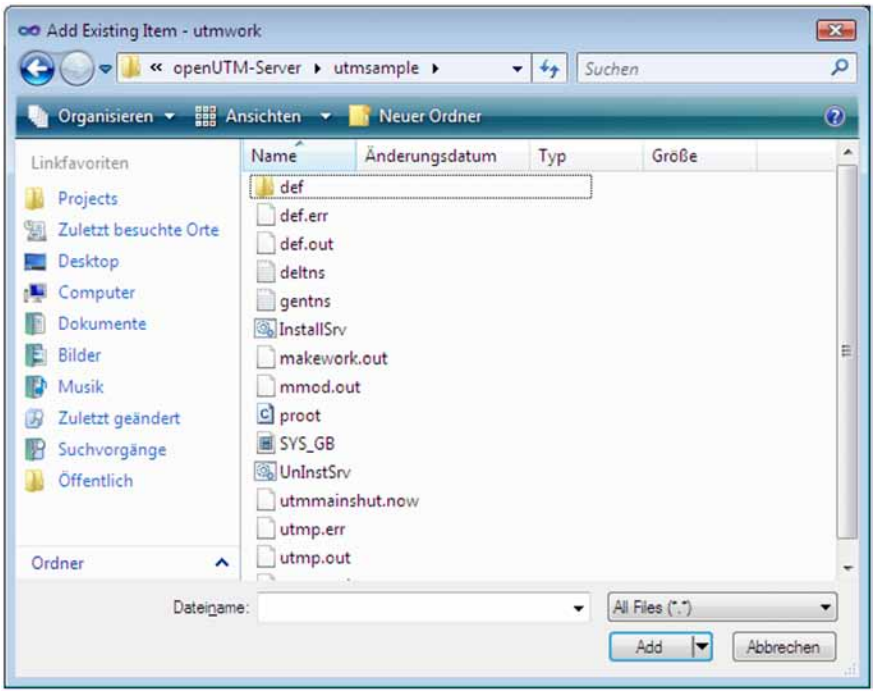
Wurde mit dem openUTM-Dienstprogramm KDCMMOD eine Source für ein anwendungsspezifisches Meldungsmodul erzeugt, so muss diese ebenfalls übersetzt und das so entstandene Objekt in die Anwendung eingebunden werden.

Zu den benötigten Quellprogrammen gehört auch die ROOT-Tabellen-Source, die Sie auf jeden Fall vor dem Binden mit KDCDEF erzeugen müssen. Es wird empfohlen, alle Quellprogramme einschließlich der ROOT-Tabellen-Source im Projektverzeichnis abzulegen.

Quellprogramme und Objektdateien in das Projekt aufnehmen

1. Öffnen Sie Ihr Projekt.
2. Fügen Sie die Quellprogramme wie folgt in Ihr Projekt ein:
   a. Klicken Sie in der Menüleiste auf *Project* und klicken Sie auf *Add Existing Item...* Damit wird das Fenster *Add Existing Item* geöffnet.
   b. Wählen Sie bei *Files of Type* den Punkt *Visual C++-Files (.c;.cpp;.cxx;.tli;.h;.tlh;.rc).*

      Navigieren Sie - falls noch nötig - in das Projektverzeichnis:



      Markieren Sie dort Folgendes:

      - Die mit KDCDEF erzeugte ROOT-Tabellen-Source, im Beispiel `proot.c`.

- Alle selbsterstellten Quellprogramme, die mit eingebunden werden sollen. Quellprogramme, die Sie innerhalb Ihres Projekts mit dem Visual Studio erstellt haben, sind automatisch im Projekt enthalten und brauchen nicht mehr markiert zu werden.

c. Klicken Sie *Add*. Damit haben Sie die Quellprogramme in das Projekt aufgenommen.

Sie können bei umfangreichen Projekten auch einzelne Quellprogramme schon vorher übersetzen (*Build - Compile ...*) und dann stattdessen die Objektdateien hinzufügen wie unten beschrieben.

3. Fügen Sie auf analoge Weise die UTM-Objektdateien zu Ihrem Projekt hinzu:
   a. Klicken Sie in der Menüleiste auf *Project* und klicken Sie auf *Add Existing Item...* Damit wird das Fenster *Add Existing Item* geöffnet.
   b. UTM-Objektdateien hinzufügen:
   Wählen Sie bei *Files of Type* den Punkt *All Files (*.*)*.
   Navigieren Sie in das Verzeichnis *utmpfad*\sys und markieren folgende Dateien:
      - Entweder, wenn keine C++-Programme enthalten sind, `mainutm.obj` bzw. das auf `mainutm.c` basierende Objekt.
      - oder, wenn C++-Programme enthalten sind, `mainutmCC.obj` bzw. das auf `mainutm.cpp` basierende Objekt.

      `mainutm.obj`/`mainutm.c` bzw. `mainutmCC.obj`/`mainutm.cpp` enthalten die Main-Funktion der Anwendung.
   c. Fügen Sie - falls vorhanden - das anwendungsspezifische Meldungsmodul hinzu - dies muss aber nicht in *utmpfad*\sys stehen.
   d. Klicken Sie jetzt auf *Add*. Damit haben Sie die UTM-Objektdateien eingefügt.
4. Weitere Objektdateien hinzufügen:
   Falls Sie schon Programme übersetzt haben, dann wiederholen Sie die Schritte a) bis c) aus 3. für die zugehörigen Objektdateien.

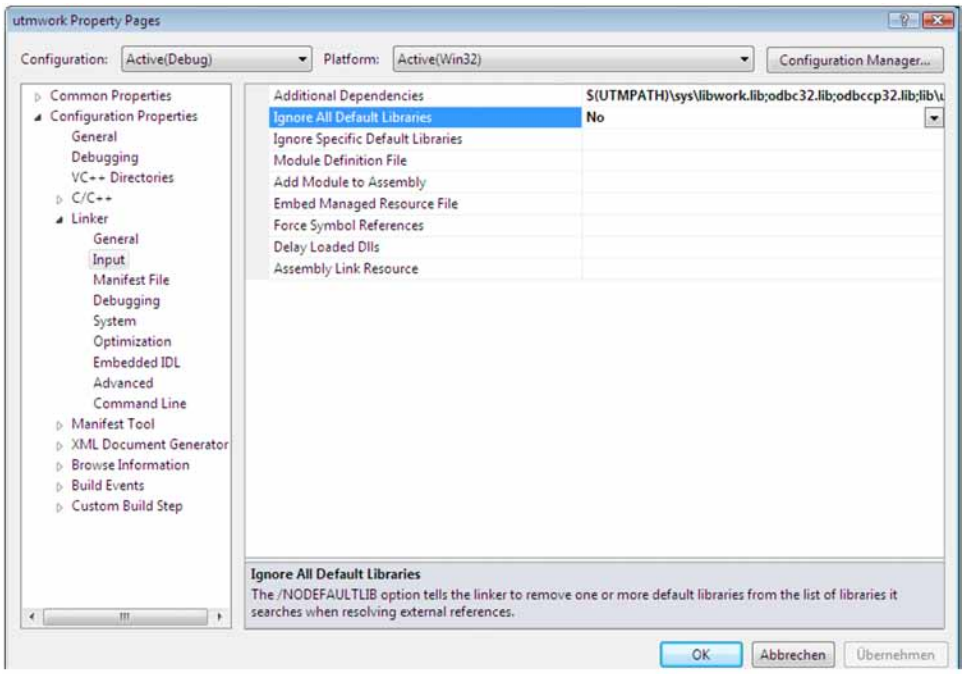
Dateien anzeigen

Sie können sich jederzeit alle in Ihrem Projekt enthaltenen Dateien anzeigen lassen, indem Sie im Navigationsbereich auf *Solution Explorer* klicken. Im Arbeitsfenster können Sie z.B. auch Quellprogramme per Doppelklick öffnen und anschließend editieren.

Bindeoptionen einstellen

Vor jedem Binden einer Anwendung müssen Sie die Bindeoptionen einstellen.

1. Öffnen Sie über die Menüleiste *Project* - *Properties* das Fenster *utmwork Property Pages* des Projektes (*utmproject*).
2. Wählen Sie im Navigationsbereich *Configuration Properties* - *Linker*.
3. Klicken Sie unter *Linker* auf *General* und tragen Sie bei *Output File* den Namen `utmwork.exe` ein. `utmwork.exe` ist der Name des gebundenen Workprozesses, dieser Name ist nicht frei wählbar.
4. Klicken Sie unter *Linker* auf *Input* und tragen bei *Additional Dependencies* die UTM-Bibliothek `libwork.lib` ein. Diese Bibliothek muss **vor** allen anderen Bibliotheken stehen.



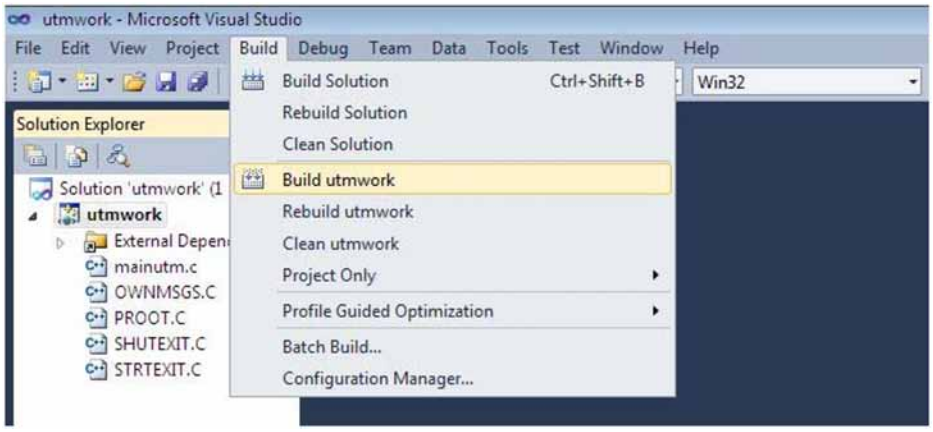
Falls Sie die XATMI-Schnittstelle oder Datenbanken verwenden, müssen Sie jeweils noch folgende Bibliotheken einfügen:

- Für XATMI die Bibliothek `libxtclt.lib`.
- Die Datenbank-Bibliothek(en). Welche Bibliothek(en) angegeben werden müssen, entnehmen Sie bitte der Dokumentation für die jeweilige Datenbank.

5. Klicken Sie jetzt OK. Damit sind Ihre gewählten Binderoptionen gültig.

Anwendungsprogramm binden

1. Öffnen Sie Ihr Projekt und wählen Sie für das Zielsystem x64 für ein 64-Bit-Programm aus.
2. Wählen Sie in der Menüleiste den Punkt *Build* und stellen den Cursor auf den Punkt *Build utmwork*, siehe Bildausschnitt:



Sobald Sie den Punkt anklicken, werden Ihre Programme übersetzt und Ihre Anwendung gebunden.

Meldungen beim Binden

Beim Binden erhalten Sie die Meldungen `LNK4075` und `LNK4056`. Diese Meldungen können Sie ignorieren; die Meldung `LNK4056` entsteht deshalb, weil die Anwendungsprogramme source-kompatibel zu Unix- und Linux-Systemen sind.

Wenn Sie die `exit()`-Funktion in einem Start-Exit verwenden, dann erhalten Sie noch die Meldung `LNK4006`. Ursache dafür ist, dass die `exit()`-Funktion durch die UTM-Bibliothek `libwork.lib` und nicht wie üblich durch eine Windows-Bibliothek abgehandelt wird. Dies ist einer der Gründe, warum `libwork.lib` immer an erster Stelle (s.o.) stehen muss.

Ergebnis des Binderlaufs

Am Ende des Binderlaufs wird das Anwendungsprogramm `utmwork.exe` in Ihrem Projektverzeichnis abgelegt. Da der Mainprozess von openUTM das Anwendungsprogramm immer unter diesem Namen sucht, dürfen Sie `utmwork.exe` nicht umbenennen.

2.2.2 Anwendungsprogramme als DLLs erstellen (Windows-Systeme)

Wenn Sie ein Anwendungsprogramm als DLL erzeugen möchten, dann gibt es gegenüber dem statischen Binden folgende Abweichungen:

1. Erstellen Sie das Projekt als *Dynamic-Link Library*, siehe Abschnitt „Projekterzeugen (Windows-Systeme)“.
2. Ergänzen Sie alle Teilprogramme um die Anweisung `void declspec( dllexport )`, siehe Abschnitt „Anwendung übersetzen und binden (Windows-Systeme)“. Dies kann z.B. so aussehen:

   ```
   void declspec( dllexport ) func (struct kc_ca *kb, struct work *spab)
   ```

Alle anderen Schritte sind analog zum statischen Binden.

DLLs werden wie Shared Objects auf Unix- und Linux-Systemen behandelt. Details zum Generieren und zum Einsatz von DLLs (alias Shared Objects) finden Sie in Abschnitt „Shared Objects austauschen“.

2.2.3 COBOL-Anwendungsprogramme auf Windows-Systemen

Wenn Sie Anwendungsprogramme in COBOL erstellen möchten, verwenden Sie den Compiler Visual Cobol von Micro Focus. Zum Ablauf von Programmen, die mit einem Micro Focus Compiler übersetzt wurden, werden die Cobol Runtime Lizenzen von Micro Focus benötigt.

Anwendungsprogramm übersetzen

Zum Übersetzen sind folgende Schritte notwendig:

1. Umgebungsvariablen setzen:
   - Rufen Sie das Befehlsskript `<visualcoboldir>\base\bin\CreateEnv.bat` auf.
   - Ergänzen Sie die Umgebungsvariable COBCPY um das Verzeichnis `%UTMPATH%\copy-cobol85`.
   - Erweitern Sie die Umgebungsvariable INCLUDE um `<Pfad>\include`, wobei `<Pfad>` das Installationsverzeichnis des COBOL-Compilers ist (notwendig für die Übersetzung des Root Sources).
   - Falls Sie Programme auf Basis von CPIC, TX bzw. XATMI unter openUTM erstellen, erweitern Sie die Umgebungsvariable COBCPY um `%UTMPATH%\<interface>\copy-cobol85`, wobei <interface> für `cpic`, `tx` bzw. `xatmi` steht.
   - Falls Sie Client-Programme auf Basis von UPIC-L erstellen, erweitern Sie die Umgebungsvariable COBCPY um `%UTMPATH%\upicl\copy-cobol85`.
2. Öffnen Sie die Eingabeaufforderung, z.B. über *Start - Programme - Eingabeaufforderung*, und geben Sie das Kommando `cobol` ein. Danach geben Sie die Source-Dateien interaktiv an.

   Übersetzen Sie auch `%UTMPATH%\src\mfcobol\MAINUTMCOB.cbl` und verwenden Sie das daraus entstandene Objekt.

   Falls Sie das Quick Start Kit installiert haben, können Sie auch das Makefile `workcob.mak` nach eigenen Bedürfnissen anpassen.

Anwendungsprogramm binden

Das Binden erfolgt in zwei Schritten:

1. Öffnen Sie ein Eingabeaufforderungs-Fenster und geben Sie das Kommando `cblnames` ein. Dabei geben Sie alle COBOL-Objekte und sonstigen Objekte einzeln an, z.B. `%UTMPATH%\sys\MAINUTMCOB.OBJ` und `root.obj`.
2. Binden Sie das Programm `utmwork.exe` mit Hilfe des Microsoft Binders `link`.

   Dabei müssen Sie folgende Objekte einbinden:
   - `@cbllds.lnk` (Output von `cblnames`)
   - weitere Anwendungsprogramm-Bibliotheken (falls vorhanden)
   - `%UTMPATH%\sys\libwork.lib` (Import-Bibiothek von UTM)
   - `cblrtss.lib` (Cobol-Laufzeitsystem)

- C-Laufzeitsystem, z.B. `msvcrt.lib kernel32.lib user32.lib gdi32.lib advapi32.lib`

  > **i**
  >
  > - Unter Umständen kann es erforderlich sein, die Umgebungsvariable LIB auf die Verzeichnisse mit diesen Bibliotheken zu setzen.
  > - Soll das Programm animiert werden, muss der Schalter `/BD` angegeben werden.
  > - Zum Definieren des COBOL-Main-Entries muss der Schalter `/mMainUtm` verwendet werden.

  Die Optionen für beide Schritte übernehmen Sie am einfachsten aus dem Quick Start Kit. Das Makefile für `nmake` wird im Verzeichnis *filebase* unter dem Namen `workcob.mak` abgelegt.

Bitte beachten Sie auch die Compiler-spezifischen Hinweise im Handbuch openUTM-Handbuch „Anwendungen programmieren mit KDCS“ (Kapitel Ergänzungen für COBOL, Abschnitt „Plattform-spezifische Besonderheiten auf Windows-Systemen“).

2.2.4 Anwendung als Dienst installieren (Windows-Systeme)

Sie können eine vorhandene UTM-Anwendung als Windows-Dienst (Service) einrichten, so dass die Anwendung beim Hochfahren des Rechners automatisch gestartet und beim Herunterfahren automatisch beendet wird. Sie installieren und deinstallieren einen Dienst, indem Sie das Programm `utmmains` aufrufen. `utmmains` ist Bestandteil von openUTM und befindet sich in *utmpfad* `\ex`.

Installieren eines UTM-Dienstes

Um eine UTM-Anwendung als Dienst zu installieren, müssen Sie sich unter einer Kennung mit Administrationsberechtigung auf dem Windows-System anmelden und mit *Start-Programme-Eingabeaufforderung* ein Kommandofenster öffnen. Dort rufen Sie `utmmains` wie folgt auf:

`utmmains [ -d] install` *servicename filebase* [*startparam-file* [*user*]]

Die Parameter müssen durch Leerzeichen getrennt sein und haben folgende Bedeutung:

-d

Es werden zusätzlich Diagnoseinformationen auf *stdout* ausgegeben; optional.

install

gibt an, dass `utmmains` einen Dienst installiert.

servicename

Variabler Namensteil des Dienstes; Pflichtparameter.
Der vollständige Name des Dienstes lautet `openUTM` *servicename*. Es wird empfohlen, für *servicename* den Anwendungsnamen aus der KDCDEF-Anweisung MAX APPLINAME zu nehmen.

filebase

Vollqualifizierter Basisname der UTM-Anwendung; Pflichtparameter.
Der Basisname ist der Name des Verzeichnisses, in dem sich Anwendungsprogramm und KDCFILE befinden.

startparam-file

Vollqualifizierter Name der Startparameterdatei.
Standard: Startp.std

user

Konto-Name für den Ablauf der Anwendung.

Hier können Sie einen lokalen Benutzer oder einen Domänen-Benutzer angeben. Der Name muss folgende Syntax besitzen:
Bei einem lokalen Benutzerkonto: `.\` *LocalUser*
Bei einem Domänen-Benutzerkonto: *DomainName**DomainUser*
Standard: Systemkonto

Wenn Sie hier einen Benutzer angeben, dann müssen Sie das zugehörige Passwort beim Konfigurieren des Dienstes eintragen, siehe unten.

`utmmains` richtet den Dienst sofort ein. Danach müssen Sie den Dienst konfigurieren.

Konfigurieren des UTM-Dienstes

Als Administrator können Sie die Startart und die Eigenschaften des Anmelde-Kontos festlegen. Dazu gehen Sie wie folgt vor:

1. Öffnen Sie das Fenster *Systemsteuerung*, indem Sie *Start - Systemsteuerung* klicken.
2. Klicken Sie auf *Verwaltung* und dann auf *Dienste*. Damit wird ein Fenster geöffnet, das alle auf dem Rechner vorhandenen Dienste auflistet.
3. Markieren Sie den gewünschten Dienst; UTM-Dienste haben immer das Präfix `openUTM`. Die Spalte *Status* zeigt an, ob der Dienst gestartet ist oder nicht.
4. Klicken Sie auf den Button *Startart...*. Damit wird ein weiteres Fenster geöffnet, in dem Sie Folgendes festlegen:
   - Die Startart (Manuell/Automatisch/Deaktiviert)
   - Das Konto, unter dem der Dienst angemeldet wird.
     Wenn Sie beim Installieren des Dienstes schon einen Benutzer angegeben haben (Parameter *user*), wird er hier angezeigt. In diesem Fall **müssen** Sie hier das zugehörige Passwort eingeben, sonst können Sie den Dienst nicht starten.

   > ¡ Wenn ein Dienstprogramm wie z.B. *KDCKAA.exe* nicht auf die Ressourcen der UTM-Anwendung (z. B. Shared Memory) zugreifen kann , dann kann das daran liegen, dass es sich um unterschiedliche Konten handelt. In diesem Fall sollten Sie den Dienst so konfigurieren, dass er mit den gleichen Kontodaten angemeldet wird wie der Benutzer, der das Dienstprogramm startet.

5. Klicken Sie OK und verlassen Sie die Systemsteuerung.

Deinstallieren eines UTM-Dienstes

Um einen UTM-Dienst zu deinstallieren, melden Sie sich unter einer Kennung mit Administrationsberechtigung an, öffnen ein Fenster *Eingabeaufforderung* und rufen `utmmains` mit folgenden Parametern auf:

`utmmains [-d] remove` *servicename*

Bedeutung der Parameter:

-d

Es werden zusätzlich Diagnoseinformationen auf *stdout* ausgegeben; optional.

remove

gibt an, dass `utmmains` einen Dienst deinstalliert.

servicename

Variabler Namensteil des UTM-Dienstes.

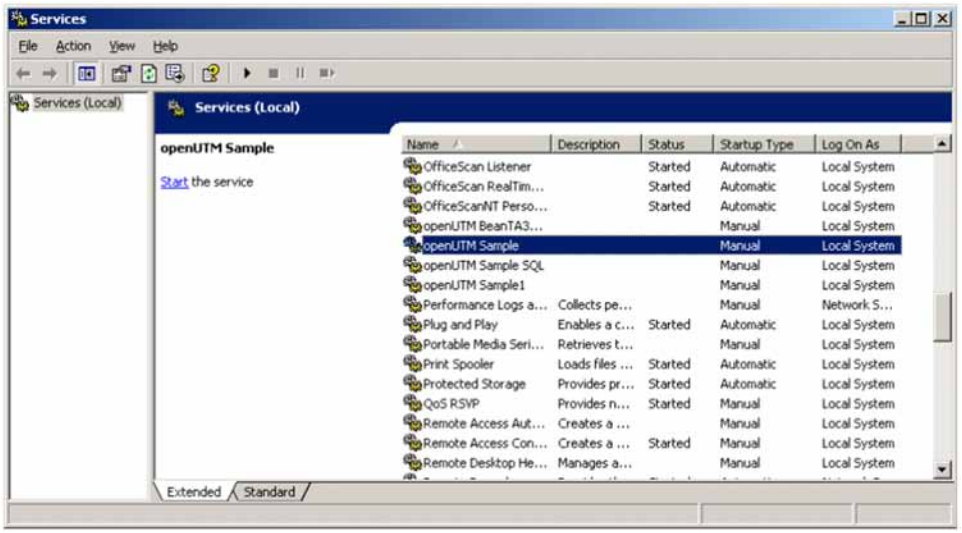
Beispiel

Sie möchten die Anwendung `Sample` als Dienst einrichten und auf das Systemkonto anmelden. KDCFILE, Startparameterdatei und Anwendungsprogramm befinden sich im Verzeichnis `C:\utmserv`. Die Startparameterdatei besitzt den Namen `startparameter`.

- Um den Dienst zu installieren, rufen Sie `utmmains` wie folgt auf:

  ```
  utmmains install Sample C:\utmserv C:\utmserv\startparameter
  ```

  Wenn Sie auf Windows jetzt nacheinander *Start* - *Systemsteuerung* - *Verwaltung* - *Dienste* (englisch: *Start* - *Control Panel* - *Administrative Tools* - *Services*) anklicken und nach unten blättern, siehe z.B. folgendes Bild:

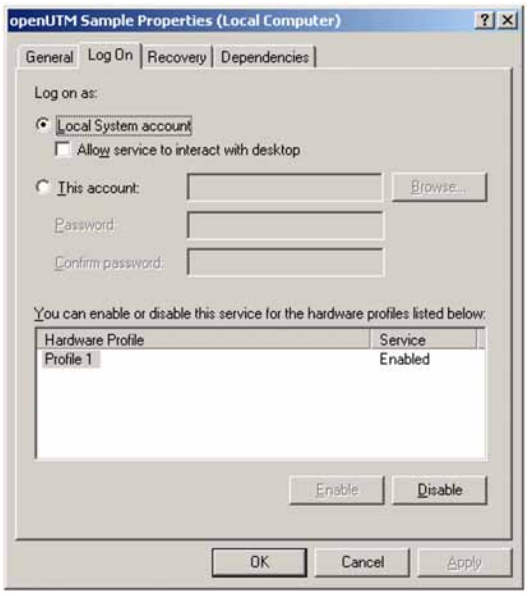


- Klicken Sie auf *Allgemein* (*General*) und konfigurieren Sie den Dienst in folgendem Fenster:

  Hier können Sie z.B. bei Starttyp auf *Automatisch* umschalten.

  Beachten Sie bitte Folgendes: Wenn Sie bei Starttyp *Automatisch* wählen, wird die UTM-Anwendung nur dann dauerhaft beendet, wenn Sie diesen Dienst beenden, siehe Abschnitt „Dienst beenden auf Windows-Systemen“.

- Klicken Sie auf *Anmelden* (*Log On*) und konfigurieren Sie den Dienst in folgendem Fenster:



  Hier können Sie bei *Anmelden an* (*Log on as*) an Stelle des lokalen Systemkontos bei *Dieses Konto* (*This Account*) ein anderes Konto angeben.

- Zum Deinstallieren rufen Sie `utmmains` wie folgt auf:

  ```
  utmmains remove Sample
  ```

3 Notwendige Dateien und systemglobale Betriebsmittel

Vor jedem Starten einer UTM-Anwendung müssen Sie dafür sorgen, dass die folgenden für den Betrieb einer UTM-Anwendung notwendigen Dateien vorhanden sind:

- die KDCFILE
- die Systemdateien stderr und stdout
- die System-Protokolldatei SYSLOG
- die Benutzer-Protokolldatei(en) USLOG (optional)
- das Dateiverzeichnis DUMP
- alle Programm- und Objektmodul-Bibliotheken, aus denen beim Starten und während des Betriebs der Anwendung Module dynamisch nachgeladen werden sollen.

KDCFILE, USLOG, SYSLOG und das Verzeichnis DUMP müssen in stand-alone Anwendungen im Verzeichnis *filebase* stehen (=Basisverzeichnis der UTM-Anwendung). *filebase* müssen Sie in den Startparametern angeben. Den Namen *filebase* legen Sie beim Erzeugen der KDCFILE mit dem Generierungstool KDCDEF fest, siehe openUTM-Handbuch „Anwendungen generieren“, Steueranweisung MAX...,KDCFILE=*filebase*.

Welche Dateien für den Betrieb einer UTM-Cluster-Anwendung notwendig sind, entnehmen Sie dem Kapitel „UTM-Cluster-Anwendung“.

3.1 Systemdateien stderr und stdout

openUTM protokolliert Meldungen auf die Systemdateien *stderr* und/oder *stdout*.

Diese Systemdateien können Sie im laufenden Betrieb umschalten. Nach dem Umschalten können die alten *stderr*- und *stdout*-Dateien ausgewertet und ggf. gelöscht werden, um den belegten Plattenspeicherplatz zu reduzieren.

Systemdateien umschalten

Die Systemdateien können im laufenden Betrieb per Administration oder über ein einstellbares Zeitintervall umgeschaltet werden. Die Systemdateien werden immer alle gemeinsam umgeschaltet.

- Per Administration schalten Sie die Systemdateien um
  - mit dem Kommando KDCAPPL SYSPROT=NEW
  - per Programmschnittstelle mit dem Feld *sysprot_switch* in der Datenstruktur *kc_diag_and_account_par_str* (siehe openUTM-Handbuch „Anwendungen administrieren“)
  - über WinAdmin/WebAdmin

  Die Systemdateien werden möglichst zeitnah zur Aufforderung umgeschaltet.
- Um die Systemdateien über ein Zeitintervall umzuschalten, geben Sie beim Starten der UTM-Anwendung den Startparameter SYSPROT an (siehe Abschnitt„Startparameterdatei der Anwendung“). Sie können hier ein Zeitintervall in Tagen angeben. Das Umschalten erfolgt jeweils um Mitternacht.

Namen der umgeschalteten Dateien

Beim Starten der UTM-Anwendung werden die Systemdateien mit den vom System oder vom Anwender festgelegten Namen eingerichtet. Erst beim ersten manuellen oder automatischen Umschalten werden Dateien gemäß folgender Formate erzeugt:

stdout: *präfix.out.YY-MM-DD.HHMMSS*

stderr: *präfix.err.YY-MM-DD.HHMMSS*.

präfix

Präfix, das Sie beim Starten der UTM-Anwendung für den Startparameter SYSPROT angegeben haben (siehe Abschnitt „Startparameter für openUTM“). Standardwert: utmp

YY-MM-DD.HHMMSS

Datum und Uhrzeit des Umschaltzeitpunkts

Wenn Sie die Umgebungsvariable UTM_REDIRECT_FILES auf YES setzen (siehe "Allgemeine Umgebungsvariablen für openUTM"), so wird schon nach dem Starten der UTM-Anwendung umgeschaltet und die Ausgaben in die Dateien mit den oben aufgeführten Formaten geschrieben. Es wird nicht in die bestehenden Systemdateien *stdout* / *stderr* oder deren Ausgabeumlenkung geschrieben.

3.2 System-Protokolldatei SYSLOG

openUTM protokolliert alle Ereignisse aus dem Lauf der Anwendung in die System-Protokolldatei SYSLOG (**SYS** TEM **LOG**GING), d.h. openUTM schreibt alle Meldungen mit Meldungsziel SYSLOG in diese Datei (zu „Meldungsziel“ siehe openUTM-Handbuch „Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen“).

Die System-Protokolldatei SYSLOG können Sie für die laufende Überwachung des Anwendungslaufs oder für spätere Kontrollen nutzen. Insbesondere für die Diagnose liefert die SYSLOG wichtige Information.

Die SYSLOG der Anwendung liegt immer unter dem Dateiverzeichnis *filebase,* wobei *filebase* das Dateiverzeichnis ist, unter dem die Anwendung installiert ist (Basisname der KDCFILE, definiert durch MAX KDCFILE).

openUTM bietet zwei Möglichkeiten, eine SYSLOG zu führen:

- als einfache Datei SYSLOG im Basisverzeichnis *filebase*
- als Dateigenerationsverzeichnis (**F**ile **G**eneration **G**roup, kurz FGG) SYSLOG im Basisverzeichnis *filebase*

Gegenüber einer einfachen SYSLOG-Datei hat eine SYSLOG-FGG folgende Vorteile:

- Sie können im laufenden Betrieb der Anwendung auf die jeweils folgende Dateigeneration umschalten (umschaltbare SYSLOG-Datei). Die SYSLOG können Sie z.B. mit dem Administrationskommando KDCSLOG administrieren. Sehen Sie dazu openUTM-Handbuch „Anwendungen administrieren“. Beim Umschalten schließt openUTM die zuvor beschriebene Dateigeneration.
- Sie können eine automatische Größenüberwachung für die SYSLOG einstellen. D.h. Sie können einen Schwellwert für die Größe der einzelnen Dateigenerationen der SYSLOG-FGG generieren bzw. per Administration festlegen, bei dem openUTM automatisch auf die folgende Dateigeneration der FGG umschaltet. Die
  Größenüberwachung kann während des Anwendungsbetriebs ein- und ausgeschaltet werden.

Meldungen von openUTM

openUTM gibt die folgenden Meldungen bezüglich der SYSLOG aus:

- Beim Anwendungsstart die Meldung K136:

  ```
  K136 (Erste) SYSLOG-Datei ist &FNAM
  ```

- Beim Anwendungsende die Meldung K138:

  ```
  K138 SYSLOG-Datei &FNAM geschlossen
  ```

- Beim Umschalten auf eine andere Dateigeneration die Meldung K137:

  ```
  K137 SYSLOG umgeschaltet auf Datei &FNAM
  ```

3.2.1 SYSLOG als einfache Datei

Sie können die Datei SYSLOG vor dem Start der Anwendung selbst als einfache Datei im Basisverzeichnis *filebase* anlegen. Existiert beim Start der Anwendung unter dem Namen SYSLOG weder eine Datei noch ein Dateigenerationsverzeichnis in *filebase*, dann legt openUTM eine einfache Datei mit dem Namen SYSLOG an.

Beim Start der Anwendung öffnet openUTM die Datei SYSLOG. Sie bleibt während des gesamten Anwendungslaufs geöffnet. In sie schreibt openUTM alle Ereignisse eines Anwendungslaufs.

Bei jedem Folgestart der Anwendung wird der Inhalt der SYSLOG-Datei von openUTM überschrieben. Die Protokollinformation aus dem vorherigen Anwendungslauf geht verloren. Nach dem Ende eines Anwendungslaufs sollten Sie deshalb, falls nötig, den Inhalt der SYSLOG-Datei sichern.

> **!** **ACHTUNG!**
>
> Wollen Sie die SYSLOG als einfache Datei führen, dann dürfen Sie die Größenüberwachung für die SYSLOG **nicht** generieren. Wenn Sie in diesem Fall die Größenüberwachung bei der UTM-Generierung einschalten mit MAX...,SYSLOG-SIZE=*size* (*size* > 0), dann bricht openUTM den Start der Anwendung mit Startfehlercode 58 ab.

3.2.2 SYSLOG als Dateigenerationsverzeichnis

openUTM führt die SYSLOG nur als Dateigenerationsverzeichnis, wenn beim Start der Anwendung im Basisverzeichnis *filebase* ein Dateigenerationsverzeichnis mit dem Namen SYSLOG existiert. Dieses Dateigenerationsverzeichnis müssen Sie mit dem Tool KDCSLOG anlegen, siehe Abschnitt „Das Tool KDCSLOG zum Anlegen der SYSLOG-FGG“.

Ein Dateigenerationsverzeichnis FGG (**F**ile **G**eneration **G**roup) ist ein Dateiverzeichnis mit Dateien, die über ihren Dateinamen durchnummeriert sind (z.B. 0001, 0002,...). Die Dateien heißen Dateigenerationen der FGG. Die Nummern heißen Dateigenerationsnummern.

Beim Einrichten der FGG mit KDCSLOG wird die Datei INFO erzeugt und in die FGG geschrieben. Beim ersten Start der Anwendung legt openUTM die erste Dateigeneration 0001 in der FGG an und öffnet sie als SYSLOG-Datei. Alle Prozesse der Anwendung schreiben die Meldungen mit Ziel SYSLOG zunächst in diese Dateigeneration.

Beim Umschalten legt openUTM die Folgegeneration selbst an.

Ist die Dateigeneration mit der Generationsnummer *n* die letzte Datei, in die openUTM vor dem Ende eines Anwendungslaufs geschrieben hat, dann erzeugt openUTM beim folgenden Anwendungsstart die Dateigeneration (*n*+1) und öffnet sie als SYSLOG-Datei.

In der FGG sind maximal *m* Dateigenerationen enthalten. Die Zahl *m* legen Sie beim Erzeugen der FGG mit dem Tool KDCSLOG fest. Sobald openUTM die (*m*+1)-te Dateigeneration anlegt, wird die älteste Dateigeneration gelöscht, d.h. die Dateigeneration mit der niedrigsten Generationsnummer.

3.2.3 Das Tool KDCSLOG zum Anlegen der SYSLOG-FGG

Die SYSLOG-FGG richten Sie mit dem Tool KDCSLOG ein. Sie finden es im Unterverzeichnis `ex` des Installationsverzeichnisses von openUTM. Es wird wie folgt gestartet:

Auf Unix- und Linux-Systemen aus der Shell mit
utmpfad`/ex/kdcslog` *filebase number* `[K]`

Auf Windows-Systemen in einem Eingabeaufforderungs-Fenster mit
utmpfad`\ex\kdcslog` *filebase number* `[K]`

Bedeutung der Parameter:

filebase	Name des Dateiverzeichnisses, unter dem die Anwendung installiert ist bzw. installiert werden soll (Basisname der KDCFILE).
number	Maximale Anzahl der Dateigenerationen in der FGG.
	In der FGG sind maximal *number* Dateigenerationen enthalten. Sobald openUTM die (*number*+1)-te Dateigeneration anlegt, wird die älteste Dateigeneration gelöscht (d.h. die Dateigeneration mit der niedrigsten Generationsnummer).
	Minimalwert: 1
	Maximalwert: 9999
K	(keep)
	Wenn dieser Parameter angegeben wird, werden alle Dateien aufgehoben, auch wenn *number* überschritten wird.

KDCSLOG legt zunächst das Basisverzeichnis *filebase* an, falls es noch nicht existiert. Anschließend wird unter *filebase* die FGG SYSLOG eingerichtet und innerhalb der FGG eine INFO-Datei angelegt. In der Datei INFO sind aktuelle Statusinformationen über Dateigenerationen der Gruppe abgelegt.

> **ACHTUNG!**
>
> Existiert im Verzeichnis *filebase* vor dem Aufruf von KDCSLOG bereits eine SYSLOG FGG, so wird diese FGG gelöscht und eine neue eingerichtet.

Meldungen von KDCSLOG

Das Tool KDCSLOG gibt seine Meldungen nach *stdout* und *stderr* aus. Die Meldungen von KDCSLOG finden Sie im openUTM-Handbuch „Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen“.

Beispiel

FGG SYSLOG erzeugen auf Unix- und Linux-Systemen

Der Basisname der Anwendung ist `/home/userutm/beispiel`. Das Dateigenerationsverzeichnis für die SYSLOG richten Sie wie folgt ein:

utmpfad`/ex/kdcslog /home/userutm/beispiel 10`

KDCSLOG erzeugt die FGG:

```
/home/userutm/beispiel/SYSLOG
```

und die Datei:

```
/home/userutm/beispiel/SYSLOG/INFO
```

FGG SYSLOG erzeugen auf Windows-Systemen

Der Basisname der Anwendung ist `C:\utmsample`. Das Dateigenerationsverzeichnis für die SYSLOG richten Sie wie folgt ein:

utmpfad`\ex\kdcslog C:\utmsample 10`

KDCSLOG erzeugt die FGG:

```
C:\utmsample\SYSLOG
```

und die Datei:

```
C:\utmsample\SYSLOG\INFO
```

Anmerkungen zu den Beispielen

Die UTM-Anwendung schreibt immer in die Datei mit der aktuell höchsten Generationsnummer. Wird die SYSLOG auf die nächste Dateigeneration umgeschaltet, dann legt openUTM diese Dateigeneration an. Es kann höchstens soviel nummerierte Protokolldateien geben, wie für den Parameter *number* angegeben wurde, d.h. maximal 10 Dateigenerationen. Ist die Anzahl erreicht und wird umgeschaltet, so wird die Datei mit der niedrigsten Nummer gelöscht, d.h wenn openUTM beim Umschalten die Dateigeneration 0011 anlegt, wird automatisch die Generation 0001 gelöscht usw.

> **ACHTUNG!**
>
> Achten Sie darauf, dass noch nicht ausgewertete Dateien nicht überschrieben bzw. gelöscht werden.

3.2.3.1 Automatische Größenüberwachung

Die automatische Größenüberwachung können Sie nur für FGGs verwenden. Wenn Sie die SYSLOG-Datei als einfache Datei anlegen und die automatische Größenüberwachung generieren, dann bricht openUTM den Start der Anwendung mit Startfehlercode 58 ab.

Sie können die automatische Größenüberwachung auf zwei Arten einstellen:

- bei der UTM-Generierung mit der KDCDEF-Anweisung MAX ...,SYSLOG-SIZE=*size*
- im laufenden Betrieb der Anwendung mit dem Administrationskommando KDCSLOG [SWITCH,]SIZE=*size* oder an der Programmschnittstelle zur Administration mit dem Operationscode KC_SYSLOG und Subopcode KC_CHANGE_SIZE (siehe openUTM-Handbuch „Anwendungen administrieren“)

In beiden Fällen müssen Sie für *size* einen Wert > 0 angeben.

Bei eingeschalteter Größenüberwachung überprüft openUTM vor jeder Meldungsausgabe in die SYSLOG-Datei, ob durch die Meldungsausgabe die vereinbarte Maximalgröße der Dateigeneration (*size* * Größe einer UTM-Seite) überschritten würde. Ist dies der Fall, dann wird vor der Meldungsausgabe versucht, auf die nächste Dateigeneration zu schalten. Bei Erfolg gibt openUTM die Meldung K137 aus. Die Meldung wird in die neue Dateigeneration geschrieben.

Tritt bei dem Versuch umzuschalten ein Fehler auf, dann arbeitet openUTM mit der alten Dateigeneration weiter, in die vor dem Umschaltversuch protokolliert wurde. openUTM schreibt die Meldung K139 auf *stdout* und auf die Konsole des Administrators. Außerdem wird wie bei allen DMS-Fehlern zusätzlich die Meldung K043 ausgegeben. Sie enthält einen DMS-Fehlercode, dem Sie den Grund für den Umschaltfehler entnehmen können.

Damit openUTM nicht bei jeder folgenden Meldung mit dem Ziel SYSLOG erneut erfolglos versucht, auf die nächste Dateigeneration umzuschalten, wird die automatische Größenüberwachung nach einem solchen Umschaltfehler deaktiviert.

Nachdem der Administrator den Grund für den Umschaltfehler gefunden und beseitigt hat, kann er die automatische Größenüberwachung wieder aktivieren, z.B. mit dem Kommando KDCSLOG SWITCH. Mit dem KDCSLOG SWITCH zwingt er openUTM, einen erneuten Umschaltversuch zu starten. Verläuft dieser Versuch ohne Fehler, so wird eine vorher deaktivierte Größenüberwachung automatisch wieder aktiviert.

Nach dem erfolgreichen Umschalten werden keine Meldungen mehr in die alte Dateigeneration geschrieben. Wird eine Dateigeneration geschlossen, so gibt openUTM die Meldung K138 aus.

3.2.4 Schutz vor zu großer SYSLOG-Datei

Sie können die Belegung von Speicherplatz durch die SYSLOG kontrollieren, wenn Sie die SYSLOG als FGG führen, für die FGG maximal *n* Dateigenerationen zulassen (Parameter *number* des Tools KDCSLOG) und für die SYSLOG die automatische Größenüberwachung einschalten. Sehen Sie dazu Abschnitt „Automatische Größenüberwachung“.

Die Dateigenerationen der SYSLOG werden zyklisch überschrieben, so dass die FGG maximal *n* Dateigenerationen enthält. Jede Generation hat durch die Größenüberwachung eine maximale Größe von *size* UTM-Seiten.

Damit ist der maximale Speicherverbrauch der SYSLOG-FGG:

n * *size* * (Größe einer UTM-Seite)

3.2.5 Verhalten bei Schreibfehlern

Tritt bei dem Versuch, eine Meldung in die SYSLOG zu schreiben, ein Fehler auf, dann gibt openUTM die Meldung K043 aus, die einen DMS-Fehlercode enthält. An diesem Fehlercode können Sie den Grund für den Fehler ablesen.

Das weitere Vorgehen von openUTM ist abhängig davon, ob die SYSLOG als einfache Datei oder als FGG geführt wird.

- Die SYSLOG wird als einfache Datei geführt

  Nach Ausgabe der Meldung K043 wird die Anwendung mit Grund SLOG09 abgebrochen.

- Die SYSLOG wird als FGG geführt

  openUTM versucht beim Auftreten eines Fehlers auf die nächste Dateigeneration zu schalten. openUTM schaltet auch um, wenn die Größenüberwachung ausgeschaltet bzw. nicht generiert ist. openUTM schaltet nicht um, wenn die Größenüberwachung auf Grund eines vorangegangenen Umschaltfehlers suspendiert ist.

  Schlägt der Umschaltversuch fehl, dann wird die Anwendung mit Grund SLOG09 abgebrochen.

  Kann openUTM auf die nächste Dateigeneration umschalten, dann versucht openUTM erneut die Meldung in die SYSLOG zu schreiben. Tritt dabei ein Fehler auf, wird die Anwendung mit SLOG09 abgebrochen. Tritt kein Fehler auf, läuft die Anwendung weiter, openUTM protokolliert in die neue SYSLOG-Dateigeneration.

3.3 Benutzer-Protokolldatei

In der Benutzer-Protokolldatei stehen die Sätze, die das Anwendungsprogramm mit LPUT-Aufrufen erzeugt hat. Die Benutzer-Protokolldateien einer Anwendung sind in einem Dateigenerationsverzeichnis FGG (**F**ile **G**eneration **G** roup) organisiert, d.h. einer Gruppe von Dateien, die über ihren Dateinamen durchnummeriert sind. Die Benutzer-Protokoll-Dateien stehen im Dateiverzeichnis USLA unter dem Basisverzeichnis *filebase*. Wenn Benutzer-Protokoll-Dateien benötigt werden (LPUT-Aufrufe), dann müssen diese vor dem Anwendungsstart mit dem Tool KDCUSLOG eingerichtet werden.

Aufruf von KDCUSLOG

Unix- und Linux-Systeme:
utmpfad`/ex/kdcuslog` *filebase number* `[ S | D ]`

Windows-Systeme in einem Eingabeaufforderungs-Fenster:
utmpfad`\ex\kdcuslog` *filebase number* `[ S | D ]`

Bedeutung der Parameter:

filebase	Name des Verzeichnisses, unter dem die Anwendung installiert ist oder installiert werden soll; Basisname der KDCFILE.
number	Zahl der Dateien pro Dateigenerationsverzeichnis; maximal 9999.
S	einfache Dateiführung; Standardeinstellung.
D	doppelte Dateiführung; es wird zusätzlich das Dateiverzeichnis USLB im Basisverzeichnis *filebase* erzeugt.

KDCUSLOG legt zunächst das Dateiverzeichnis *filebase* an, wenn es noch nicht existiert. Anschließend werden das Dateiverzeichnis USLA und bei doppelter Dateiführung zusätzlich USLB in *filebase* eingerichtet. Innerhalb des Dateiverzeichnisses USLA bzw. USLB wird eine INFO-Datei angelegt, in der aktuelle Statusinformationen über Dateien der FGG hinterlegt werden.

Das Dateiverzeichnis USLA enthält folgende Dateien:

/INFO: Verwaltungsdatei

0001: erste Datei der Dateigeneration (Generationsnummer 0001)

> **ACHTUNG!**
>
> Existiert vor dem Aufrufen der Prozedur bereits das Dateigenerationsverzeichnis, so wird die alte Gruppe gelöscht und eine neue eingerichtet.

Beispiel

Unix- und Linux-Systeme

Der Basisname der Anwendung ist `/home/userutm/beispiel`. Die Benutzer-Protokolldatei soll doppelt geführt werden.

Das Dateigenerationsverzeichnis für die Benutzer-Protokolldatei richten Sie wie folgt ein:

utmpfad/ex/kdcuslog /home/userutm/beispiel 2 D

KDCUSLOG erzeugt dann die Dateien:

```
/home/userutm/beispiel/USLA/INFO
/home/userutm/beispiel/USLA/0001
/home/userutm/beispiel/USLB/INFO
/home/userutm/beispiel/USLB/0001
```

Windows-Systeme

Der Basisname der Anwendung ist `C:\utmsample`. Die Benutzer-Protokolldatei soll doppelt geführt werden.

Das Dateigenerationsverzeichnis für die Benutzer-Protokolldatei richten Sie wie folgt ein:

utmpfad\kdcuslog C:\utmsample 2 D

KDCUSLOG erzeugt dann die Dateien:

```
C:\utmsample\USLA\INFO
C:\utmsample\USLA\0001
C:\utmsample\USLB\INFO
C:\utmsample\USLB\0001
```

Anmerkungen zum Beispiel

Die UTM-Anwendung schreibt immer in die Datei mit der aktuell höchsten Nummer. Bei jedem KDCLOG-Kommando durch den Administrator wird auf die folgende Dateigeneration weitergeschaltet. Es kann höchstens soviel nummerierte Benutzer-Protokolldateien geben, wie unter dem Parameter *number* (im Beispiel 2) beim Aufruf von KDCUSLOG angegeben wurden. Ist die Anzahl erreicht und wird mit KDCLOG weitergeschaltet, so wird die Datei mit der niedrigsten Nummer gelöscht.

Achten Sie darauf, dass noch nicht ausgewertete Dateien nicht überschrieben werden.

Die Benutzer-Protokoll-Sätze schreibt openUTM nicht sofort in die Protokolldatei, sondern speichert sie erst im Pagepool der KDCFILE. Sind im Pagepool so viele UTM-Seiten belegt, wie in MAX...,LPUTBUF=*number* generiert, kopiert openUTM die Sätze in die Benutzer-Protokolldatei. Das Kopieren erfolgt asynchron zu laufenden Transaktionen. Beim normalen Beenden der Anwendung kopiert openUTM die Sätze ebenfalls in die Benutzer-Protokolldatei.

Die Anzahl der bei LPUTBUF=*number* angegebenen UTM-Seiten ist bei der UTM-Generierung der Größe des Pagepools mit MAX...,PGPOOL=*number* zu berücksichtigen.

Mit MAX...,LPUTLTH=*length* beeinflussen Sie die Blocklänge der Benutzer-Protokolldatei. Sie wird von openUTM berechnet und kann größer sein als die Standardblockung 2KB.

openUTM kann LPUT-Sätze nur in die Benutzer-Protokolldatei kopieren, wenn diese eingerichtet ist und openUTM darauf zugreifen kann.

Bitte beachten Sie, dass die Benutzer-Protokolldatei nach einem KDCDEF-Lauf oder nach einem KDCUPD-Lauf von Anfang an überschrieben wird; ansonsten wird sie fortgeschrieben. Deshalb sollten Sie die Protokoll-Sätze vor einem KDCDEF- bzw. KDCUPD-Lauf auswerten.

Meldungen von KDCUSLOG

KDCUSLOG gibt seine Meldungen nach *stdout* und *stderr* aus.
Die Meldungen von KDCUSLOG finden Sie im openUTM-Handbuch „Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen“.

3.3.1 Verhalten bei Schreibfehlern

Tritt beim Schreiben von LPUT-Sätzen in die Benutzer-Protokolldatei ein DMS-Fehler (**D**ata **M**anagement **S**ystem) auf, so gibt openUTM die Meldung K043 aus, die einen DMS-Fehlercode enthält. An diesem Fehlercode können Sie den Grund für den Fehler ablesen. Gleichzeitig wird jeder weitere LPUT-Aufruf im Teilprogramm mit dem KDCS-Returncode 40Z (interner Returncode K903) abgewiesen.

Der Administrator der Anwendung kann dann die Benutzer-Protokolldatei bzw. ihre Generationen korrigieren, restaurieren oder neu einrichten.

Damit openUTM wieder LPUT-Sätze in die Benutzer-Protokolldatei schreibt, muss der Administrator der Anwendung nach der Fehlerbehebung das Administrationskommando KDCLOG oder einen KDCADMI-Aufruf mit Opcode KC_USLOG absetzen (siehe openUTM-Handbuch „Anwendungen administrieren“).

Die Dateigenerationsnummer wird erhöht. Die im Pagepool der KDCFILE gesicherten LPUT-Sätze werden anschließend in die Protokolldatei(en) geschrieben. Die Sperre für die LPUT-Aufrufe in den Teilprogrammen wird aufgehoben.

3.4 Dateiverzeichnis DUMP

In dem Dateiverzeichnis DUMP werden folgende Dateien abgelegt:

- Dump-Dateien, die evtl. während des Anwendungslaufs erzeugt wurden
- Temporäre Dateien, die für die Dump-Erstellung notwendig sind
- Bei Bedarf auch core-Dateien (Unix- und Linux-Systeme) bzw. Mini-Dumps (Windows-Systeme) für die Diagnose

Dieses Dateiverzeichnis sollten Sie daher stets vor dem Start einrichten, damit diese Dateien erzeugt werden können. Das Dateiverzeichnis DUMP müssen Sie im Basisverzeichnis *filebase* anlegen.

3.5 Systemglobale Betriebsmittel einer Anwendung

In diesem Abschnitt werden die systemglobalen Betriebsmittel aufgelistet, die eine UTM-Produktivanwendung benötigt. Sie erfahren, wie Sie die Größe des Shared Memory-Bereichs für die Inter-Prozess-Kommunikation (IPC) anpassen können, um die Performance Ihrer Anwendung bei der Kommunikation mit Netz-Partnern zu verbessern.

3.5.1 Von einer UTM-Anwendung benötigte Betriebsmittel

Eine UTM-Produktivanwendung benötigt die im Folgenden aufgeführten systemglobalen Betriebsmittel.

Shared Memory-Bereiche

Eine UTM-Anwendung benötigt drei Shared Memory-Bereiche für die Konfigurations- und Anwendungs-globalen Verwaltungsdaten (KAA), den Cache und die UTM-interne Prozesskommunikation (siehe auch Abschnitt „Performance-Verbesserung: Größe derDatenbereiche im IPC-Shared Memory anpassen“).

Für die Kommunikation über OSI TP werden zusätzlich ein OSS- und ein XAPTP Shared Memory benötigt.

Semaphore

Eine UTM-Anwendung benötigt Semaphore zur Steuerung und Synchronisation von Prozessen der UTM-Anwendung.

In openUTM sind die Semaphore als Semaphor-Arrays organisiert, wobei jedes Semaphor-Array genau 20 Semaphor-Einträge enthält. Eine UTM-Anwendung benötigt mindestens ein Semaphor-Array. Die maximale Anzahl der Semaphore im System ist beschränkt.

In der UTM-Umgebung werden die Semaphor-Einträge folgendermaßen belegt:

- neun Einträge für IPC-Shared Memory
- ein Eintrag für KAA-Shared Memory
- ein Eintrag für CACHE-Shared Memory
- ein Eintrag für CACHE-Zugriffssperre
- ein Eintrag für PCMM-Zugriffssperre
- je ein Eintrag für OSS und XAPTP Shared Memory
- je ein Eintrag für jeden Workprozess als Taskbörse
- je ein Eintrag für jeden angeschlossenen externen Prozess (*utmtimer*, *utmdtp*, *utmprint* und lokales UPIC-Client-Programm) zur Kommunikation zwischen Workprozess und externem Prozess.

- je zwei Einträge für jeden angeschlossenen Netzprozess vom Typ *utmnet,* utmnets oder utmnetssl, zur Kommunikation zwischen Workprozess und Netzprozess.

  Wie viele Netzprozesse gestartet werden, ist abhängig von der Art der Netz-Anbindung einer UTM-Anwendung:

  Anbindung über PCMX

  Für jede Listener-ID, die mit KDCDEF generiert wird (BCAMAPPL- oder ACCESS-POINT-Anweisung), wird je ein Netzprozess (*utmnet*) gestartet.

  Anbindung über die Socket-Schnittstelle (native TCP/IP)

  Für jede Socket-Listener-ID, die mit KDCDEF generiert wird (BCAMAPPL-Anweisung mit T-PROT=SOCKET und SECURE=NO), wird je ein Socket-Netzprozess (*utmnets*) gestartet.

  Für jede Socket-Listener-ID, die mit KDCDEF generiert wird (BCAMAPPL-Anweisung mit T-PROT=SOCKET und SECURE=YES), wird je ein TLS Socket-Netzprozess (*utmnetssl*) gestartet.
  Das bedeutet:
  Für eine minimale Produktivanwendung („Ein-Prozess-Anwendung“) und bei Generierung von einem Schlüssel für die Semaphore kann mindestens ein Dialog-Terminalprozess (*utmdtp*) angeschlossen werden.

  Mit MAX...,SEMARRAY= kann ein Bereich von bis zu 1000 aufeinanderfolgenden Schlüsseln definiert werden. Mit der KDCDEF-Generierung mit MAX...,SEMKEY= können bis zu 10 einzelne Schlüssel für Semaphore definiert werden.

> ¡ Wenn Sie bei der Generierung der UTM-Anwendung mit KDCDEF zu wenige Semaphor Schlüssel generiert haben, dann wird die Meldung U189 mit Engpass "SEMA USED" ausgegeben und die UTM-Anwendung wird abnormal beendet.

Filedeskriptoren

Ein Workprozess einer UTM-Anwendung belegt immer folgende Filedeskriptoren für:

- *stdin*
- *stdout*
- *stderr*
- die KDCFILE-Datei
- die SYSLOG-Datei
- das Shared Memory IPC
- das Shared Memory KAA
- das Shared Memory CACHE
- eine Named Pipe zum Mainprozess (*utmmain*)
- eine Named Pipe zum Loggingprozess (*utmlog*)

Zusätzliche Filedeskriptoren für die Kommunikation über OSI TP für:

- das Shared Memory OSS
- das Shared Memory XAPTP

Weitere Filedeskriptoren werden benötigt, wenn die KDCFILE doppelt geführt wird, wenn Pagepool-Dateien verwendet werden (MAX ...,PGPOOLFS=) oder wenn der Wiederanlaufbereich in mehrere Dateien aufgeteilt wird (MAX...,RECBUFFS=).

In UTM-Cluster-Anwendungen werden zusätzliche Filedeskriptoren für die Cluster-globalen Dateien benötigt:

- Cluster-Konfigurationsdatei
- Cluster-User-Datei
- Cluster-Pagepool Verwaltungsdatei
- Cluster-Pagepool Datei(en)
- Cluster-GSSB-Datei
- Cluster-ULS-Datei
- Cluster-Administrations-Journal
- Cluster-Lock-Datei
- Cluster-Startserialisierungs-Datei

Ein Workprozess der Anwendung belegt kurzzeitig weitere Filedeskriptoren für:

- die aktuelle Benutzer-Protokolldatei (USLOG)
- Dump-Dateien bei Fehlern
- den Start einer Knoten-Anwendung in einer UTM-Cluster-Anwendung

Die Datei applifile

Die Datei wird bei der openUTM-Installation im Installationsverzeichnis eingerichtet und enthält die Namen aller Anwendungen, die im System seither gestartet wurden, inklusive Statusinformationen der Anwendungen. Außerdem enthält die Datei die Schlüssel (Keys) des Semaphors und Shared Memory Segments, die zur Kommunikation zwischen den externen Prozessen (Dialogterminal-, Drucker-, Timerprozess, lokale UPIC-Client-Programme) und den Workprozessen dienen. Die Schlüssel sind systemweit eindeutig zu vergeben.

> **ACHTUNG!**
>
> Die applifile ist eine UTM-interne Verwaltungsdatei. Sie dürfen sie **nicht** mit einem Editor öffnen. Die applifile könnte dadurch zerstört werden.

3.5.2 Performance-Verbesserung: Größe der Datenbereiche im IPC-Shared Memory anpassen

Den Shared Memory-Bereich für die Inter-Prozess-Kommunikation (IPC-Shared Memory-Bereich) benötigt die UTM-Anwendung für den Austausch der Nachrichten zwischen ihren Prozessen. Er wird von openUTM angelegt. Ist dieser Bereich zu klein, dann kann es zu Performance-Engpässen und zum Abbau von Verbindungen kommen.

Der größte Teil des IPC-Shared Memory-Bereichs dient zur Ablage der Nachrichten, die zwischen den Prozessen einer Anwendung ausgetauscht werden. Dieser Bereich wird im folgenden Datenbereich genannt. Der restliche Teil dient zur Verwaltung der Prozesse und deren Verbindungen.

Der Datenbereich im IPC-Shared Memory ist in Einheiten von 4 KB organisiert.

Die Größe des IPC-Shared Memory wird von openUTM Anwendungs-spezifisch festgelegt. Sie ist im wesentlichen bestimmt durch die Anzahl der generierten Kommunikationspartner (MAX SEMARRAY) bzw. der generierten Semaphore (MAX SEMKEY). Siehe auch „Semaphore“ (Von einer UTM-Anwendung benötigte Betriebsmittel).

Pro generiertem Semaphorschlüssel legt openUTM einen Datenbereich von ca. 10 * 4 KB an. Zusätzlich legt openUTM noch pro generiertem Kommunikationspartner einen Datenbereich von ca. 4 KB an.

Bei hohem Daten-Aufkommen auf den Verbindungen zu Kommunikationspartnern kann der von openUTM angelegte Datenbereich zu klein sein. Es kann dann zu Performance-Engpässen und somit zum Abbau von Verbindungen kommen. Um das zu vermeiden, können Sie die Größe des IPC-Shared Memorys vergrößern. Dazu dienen die Umgebungsvariablen UTM_IPC_LETTER und UTM_IPC_EXTP_LETTER. Mit UTM_IPC_LETTER beeinflussen Sie die absolute Größe des IPC-Shared Memory, mit UTM_IPC_EXTP_LETTER den Datenbereich im IPC-Shared Memory, der einer einzelnen Verbindung maximal zur Verfügung steht.

Absolute Größe des Datenbereichs anpassen

Der Datenbereich im IPC-Shared Memory wird unter den existierenden Verbindungen im Sinne von „first come - first serve“ aufgeteilt. Ist der gesamte Datenbereich belegt, werden Verbindungen abgebaut.

openUTM gibt dann folgende Meldung aus:

```
U189 IPC Engpass &IPCOBJ &IPCREAS
```

mit den Inserts &IPCOBJ=LETT und &IPCREAS=IPC FULL oder EXTP FULL.

Um dies zu vermeiden, können Sie die absolute Größe des Datenbereichs mit Hilfe der Umgebungsvariablen UTM_IPC_LETTER vergrößern. In UTM_IPC_LETTER geben Sie die Anzahl der 4KB-Einheiten an, die das IPC-Shared Memory umfassen soll. Der kleinste erlaubte Wert ist 5 (entspricht 20KB).

Eine Änderung von UTM_IPC_LETTER wirkt sich erst nach dem nächsten Start der UTM-Anwendung aus. UTM_IPC_LETTER wird beim Anwendungsstart vom ersten Workprozess ausgewertet.

Bei sehr niedrigem Daten-Aufkommen können Sie andererseits mit UTM_IPC_LETTER den Datenbereich verkleinern. Damit können Sie den Overhead verringern, der durch die Verwaltung des IPC-Shared Memorys durch das Betriebssystem benötigt wird.

Maximalen Datenbereich für das Nachrichten-Aufkommen pro Verbindung anpassen

Grundsätzlich sind alle Verbindungen bei der Vergabe des Datenbereichs im IPC-Shared Memory gleichberechtigt. Damit der Datenbereich aber nicht von einer Verbindung allein belegt wird, ist standardmäßig ein maximaler Bereich von 64 KB (16 * 4KB) festgelegt, der von einer Verbindung zu einem Zeitpunkt belegt werden kann. Sind diese 64 KB von einer Verbindung bereits belegt, dann wird die Verbindung abgebaut.

openUTM gibt dann folgende Meldung aus:

```
U189 IPC Engpass &IPCOBJ &IPCREAS
```

mit den Inserts &IPCOBJ=LETT und &IPCREAS=MAX ILETT oder MAX OLETT.

Um dies zu vermeiden, können Sie den Datenbereich, der im IPC-Shared Memory pro Verbindung maximal zur Verfügung steht, mit der Umgebungsvariablen UTM_IPC_EXTP_LETTER vergrößern.

Mit UTM_IPC_EXTP_LETTER können Sie die maximale Größe des Datenbereichs pro Verbindung in 4KB-Einheiten angeben. Als Standardwert ist 16 eingestellt. Der kleinste erlaubte Wert ist 1 (entspricht 4KB).

Eine Änderung von UTM_IPC_EXTP_LETTER wirkt sich erst nach dem nächsten Start der UTM-Anwendung aus. UTM_IPC_EXTP_LETTER wird beim Anwendungsstart vom ersten Workprozess ausgewertet.

4 UTM-Anwendung starten

Eine UTM-Anwendung wird gestartet, indem das Programm *utmmain* aufgerufen wird. Dieses Programm ist die Main-Funktion der Anwendung und läuft als Hintergrundprozess ab. *utmmain* erzeugt die Workprozesse (*utmwork*), den Timerprozess (*utmtimer*) und bei Bedarf Netzprozesse (*utmnet*, *utmnets, utmnetssl*) und Druckerprozesse (*utmprint*, nur bei Unix- und Linux-Systemen). Sehen Sie dazu auch das openUTM-Handbuch „Konzepte und Funktionen“.

Zu Testzwecken kann eine UTM-Anwendung auch mit einem Debugger gestartet werden, siehe openUTM-Handbuch „Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen“, Abschnitt „Testen von UTM-Anwendungen“.

Bitte beachten Sie, dass Sie das Programm *utmmain* immer unter dem Anwendungsverzeichnis (*filebase*) gestartet werden muss.

Cluster-spezifische Besonderheiten beim Starten einer UTM-Cluster-Anwendung entnehmen Sie dem Abschnitt „Starten einer UTM-Cluster-Anwendung“.

4.1 Starten einer UTM-Anwendung auf Unix- und Linux- Systemen

Zum Starten einer Anwendung mit *utmmain* sind folgende Schritte notwendig:

1. Setzen Sie die Umgebungsvariable UTMPATH auf den *utmpfad*. Es ist sinnvoll, wenn Sie dafür folgende Kommandos in die *.profile* bzw. in die *login-file* der Benutzerkennung aufnehmen, unter der die Anwendung abläuft:

   `UTMPATH=`*utmpfad*

   `export UTMPATH`

   Näheres zum *utmpfad* siehe "UTM-Systemfunktionen auf Unix- und Linux-Systemen installieren".
2. Erstellen Sie die Startparameterdatei wie ab "Startparameterdatei der Anwendung" beschrieben.
3. Starten Sie *utmmain* im Verzeichnis *filebase* als Hintergrundprozess:

   utmpfad`/ex/utmmain` *filebase* [*startparam-file*] &

   filebase
   : ist der vollqualifizierte Basisname für die UTM-Anwendung (Name des Dateiverzeichnisses der Anwendung).

   startparam-file
   : ist der vollqualifizierte Name der Datei, in der die Startparameter definiert sind. Wird dieser Parameter weggelassen, dann müssen die Startparameter in der Datei *filebase*`/startparameter` stehen.

utmmain erzeugt sowohl beim Start als auch während des Anwendungslaufs Meldungen auf *stdout*; Fehlermeldungen beim Start werden auf *stderr* ausgegeben, siehe auch "Fehlermeldungen beim Start". Diese Meldungen können wie in folgendem Beispiel in eine Datei oder ein Filterprogramm umgelenkt werden.

Beispiel

Anwendung und Startparameterdatei befinden sich im Verzeichnis `/home/utmbsp`, die Startparameterdatei hat den Standardnamen `startparameter`.

Wenn alle Ausgaben in Dateien umgelenkt werden sollen, rufen Sie *utmmain* wie folgt auf:

utmpfad`/ex/utmmain /home/utmbsp 1>utmp.out 2>utmp.err &`

Wenn die Ausgabe stattdessen von einem Programm oder Shellskript namens `filter` weiterbehandelt werden soll, rufen Sie *utmmain* z.B. wie folgt auf:

utmpfad`/ex/utmmain /home/utmbsp 2>&1 | filter`

Mit dieser Methode kann z.B. auf Meldungen des Netzprozesses reagiert werden, was mit einem MSGTAC-Programm nicht möglich ist.

> ¡ Die Umlenkung zum `filter` Skript ist nur möglich, wenn das Umschalten der Protokolldateien nicht aktiviert ist (siehe SYSPROT im Abschnitt„Startparameterdatei der Anwendung“).

Netzprozesse

Beim Start erzeugt *utmmain* folgende Netzprozesse:

- Einen oder mehrere *utmnet*-Prozesse.

- Zusätzlich werden für die TCP/IP-Kommunikation ein oder mehrere Socket-Netzprozesse *utmnets* bzw. *utmnetss/* für TLS Kommunikation gestartet, siehe auch Abschnitt „Unterschiedliche Socket-Netzprozesse einsetzen (Unix- und Linux-Systeme)“.

4.2 Starten einer UTM-Anwendung auf Windows-Systemen

Auf Windows-Systemen können Sie eine UTM-Anwendung mit dem Programm *utmmain* oder als Dienst starten.

4.2.1 Starten mit utmmain (Windows-Systeme)

Zum Starten einer Anwendung mit *utmmain* sind folgende Schritte notwendig:

1. Ergänzen Sie für die Benutzerkennung, unter der die UTM-Anwendung laufen soll, die Variable PATH um *utmpfad* `\ex`.
2. Erstellen Sie die Startparameterdatei wie ab "Startparameterdatei der Anwendung" beschrieben.
3. Öffnen Sie mit *Start - Programme - Eingabeaufforderung* ein Kommandofenster, wechseln Sie in das Anwendungsverzeichnis und geben Sie dort Folgendes ein:

   `utmmain` *filebase* [*startparam-file*]

   filebase
   : ist der vollqualifizierte Basisname für die UTM-Anwendung (Name des Dateiverzeichnisses der Anwendung). Sie können *filebase* auch als relativen Pfadnamen eingeben, z.B. als „.“ (Punkt), wenn Sie *utmmain* aus dem Verzeichnis *filebase* aufrufen.

   startparam-file
   : ist der vollqualifizierte Name der Datei, in der die Startparameter definiert sind. Wird dieser Parameter weggelassen, dann müssen die Startparameter in der Datei *filebase* `\startparameter` stehen.

   Für den Aufruf von *utmmain* können Sie auch einen Kurzbefehl (Shortcut) erzeugen, so dass Sie die UTM-Anwendung per Mausklick oder Tastaturbefehl starten können, näheres siehe folgendes Beispiel.

utmmain erzeugt sowohl beim Start als auch während des Anwendungslaufs Meldungen auf *stdout*; Fehlermeldungen beim Start werden auf *stderr* ausgegeben, siehe auch "Fehlermeldungen beim Start". Diese Meldungen können wie in folgendem Beispiel in eine Datei umgelenkt werden.

Beispiel

Die Anwendung befindet sich im Verzeichnis `C:\utmtest\beispiel`, die PATH-Variable Ihrer Benutzerkennung haben Sie um *utmpfad* `\ex` ergänzt. Die Startparameterdatei besitzt den Standardnamen `startparameter` und befindet sich ebenfalls im Anwendungsverzeichnis.

Wenn Sie alle Meldungen in Datei umlenken möchten, öffnen Sie jetzt ein Kommandofenster und starten die UTM-Anwendung wie folgt:

```
cd C:\utmtest\beispiel
```

```
utmmain . 1>utmp.out 2>utmp.err
```

Wenn die Meldungen stattdessen von einem Programm oder einer .CMD-Datei namens `filter` weiterbehandelt werden sollen, rufen Sie *utmmain* wie folgt auf:

```
utmmain . 2>&1 | filter
```

Mit dieser Methode kann z.B. auf Meldungen des Netzprozesses reagiert werden, was mit einem MSGTAC-Programm nicht möglich ist.

> ¡ Die Umlenkung zum `filter` Skript ist nur möglich, wenn das Umschalten der Protokolldateien nicht aktiviert ist (siehe SYSPROT im Abschnitt „Startparameterdatei der Anwendung“) .

UTM-Anwendung über Shortcut starten

Für diesen Vorgang können Sie wie folgt ein Shortcut erzeugen, damit Sie die Anwendung per Mausklick oder über bestimmte Tastaturbefehle starten können.

Auf Windows-Systemen gehen Sie wie folgt vor:

1. Drücken Sie auf dem leeren Bildschirmhintergrund die rechte Maustaste, wählen Sie *Neu* und klicken Sie auf *Verknüpfung*. Das Fenster *Verknüpfung erstellen* wird geöffnet:
   - Geben Sie in das Feld *Speicherort des Elements* Folgendes ein:

     ```
     cmd.exe /c utmmain . ./startp.std >utmp.out 2>utmp.err <nul
     ```

     Durch `CMD` wird ein Kommandofenster geöffnet und die Befehle dort ausgeführt. `/C` heißt, dass das Fenster nach Beenden von *utmmain* geschlossen wird.
   - Klicken Sie auf *Weiter* und vergeben Sie im Fenster *Verknüpfung erstellen* einen sprechenden Namen, z.B. *start-utm*.
   - Klicken auf *Fertigstellen*. Das Fenster wird geschlossen, auf dem Bildschirm wird ein Symbol mit dem Namen `start-utm` angezeigt.
2. Klicken Sie mit der rechten Maustaste auf dieses Symbol und wählen Sie *Eigenschaften*. Klicken Sie das Registerblatt *Verknüpfung* an und führen dort folgende Schritte durch:
   - Tragen Sie in das Feld *Ausführen in* das Verzeichnis `C:\utmtest\beispiel` (= Anwendungsverzeichnis) ein. Damit sucht *utmmain* die Parameter in diesem Verzeichnis und legt die Dateien in diesem Verzeichnis ab.
   - Setzen Sie den Cursor in das Feld *Tastenkombination* und drücken Sie gleichzeitig `STRG`, `ALT` und `F3`. Wahlweise können Sie in diesem Fenster über den Button *Anderes Symbol...* auch ein anderes Symbol zuordnen.
   - Drücken Sie *OK*. Damit ist der Shortcut fertig, die Anwendung kann per Doppelklick oder `STRG+ALT+F3` gestartet werden.

4.2.2 Starten als Dienst (Windows-Systeme)

Eine UTM-Anwendung muss wie in Abschnitt „Anwendung als Dienst installieren (Windows-Systeme)“ beschrieben als Dienst installiert und konfiguriert werden. Dabei kann als Startart *Automatisch* eingestellt werden, so dass der Dienst bei jedem Hochfahren des Systems gestartet wird. Bei Startart *Manuell* muss der Dienst immer manuell gestartet werden.

Auf Windows starten Sie einen Dienst z.B. wie folgt:

1. Melden Sie sich unter einer Windows-Kennung an, die Administrationsberechtigung besitzt.
2. Rufen Sie die Systemsteuerung auf mit *Start - Systemsteuerung.*
3. Klicken Sie auf *System und Sicherheit - Verwaltung* und dann auf *Dienste.* Markieren Sie den gewünschten UTM-Dienst mit der rechten Maustaste; dieser hat immer den Namen `openUTM` *servicename*. *servicename* wird beim Installieren des Dienstes vergeben.
4. Wählen Sie im Kontextmenü den Befehl *Starten*. Der Dienst wird gestartet.

Beim Start und während des Anwendungslaufs erzeugt eine als Dienst gestartete Anwendung die gleichen Meldungen wie eine per *utmmain* gestartete Anwendung. Diese Meldungen werden standardmäßig in folgende Dateien geschrieben:

- Meldungen nach *stdout* in die Datei *filebase*`\utmp.out`
- Meldungen nach *stderr* in die Datei *filebase*`\utmp.err`, siehe auch "Fehlermeldungen beim Start".

Der Dateiname ist abhängig vom Startparameter für SYSPROT und dem Umschalten der Systemdateien (siehe Abschnitt „Startparameterdatei der Anwendung“).

> ¡ Wenn die Anwendung als Dienst gestartet und dabei das Systemkonto verwendet wird (Standardeinstellung), dann kann es vorkommen, dass einige Diagnose-Dateien im Systemverzeichnis abgelegt werden.

4.3 Starten einer UTM-Anwendung mit OpenSSL

In UTM-Anwendungen muss vom Anwender eine openSSL Bibliothek für die Verschlüsselungsfunktionalität zur Verfügung gestellt werden.

Sowohl beim Starten eines *utmwork* Prozesses als auch beim Starten des Dientsprogramms KDCDEF wird versucht diese openSSL Bibliothek zu laden.

Erst nach erfolgreichem Laden einer passenden openSSL Bibliothek steht die Verschlüsselungsfunktionalität in der UTM-Anwendung bzw. im Dienstprogramm KDCDEF zur Verfügung.

Das Ergebnis dieses Ladevorgangs der openSSL Bibliothek wird mittels einer K078 Meldung nach stderr ausgegeben.

Die Zuordnung der openSSL Bibliothek erfolgt über die Umgebungsvariable UTM_SSL_LIBRARY. Falls Sie die Umgebungsvariable nicht explizit setzen, wird ein Default Name verwendet. Details dazu finden Sie im Abschnitt "Umgebungsvariablen für die Nutzung der openSSL Bibliothek".

> **Verhalten im Fehlerfall**
>
> Kann die openSSL Bibliothek nicht erfolgreich geladen werden, steht in der UTM-Anwendung die Verschlüsselungsfunktionalität nicht zur Verfügung. Das hat aber keinen Einfluss auf die restliche Funktionalität von openUTM.

> **Version openSSL Bibliothek**
>
> Details zur Version der openSSL Bibliothek entnehmen Sie bitte der Freigabemitteilung für openUTM.

4.4 Starten einer UTM-Anwendung mit TLS Verbindungen

Neben der Verschlüsselungsfunktionalität muss vom Anwender auch für TLS Verbindungen eine openSSL Bibliothek zur Verfügung gestellt werden.

Sind für eine UTM-Anwendung BCAMAPPLs mit T-PROT=(SOCKET,...,SECURE) generiert, wird beim Start der Anwendung eine Anzahl von Netzprozessen vom Typ *utmnetss/* gestartet.

In jedem *utmnetss/* Prozess erfolgt die Kommunikation über den Transport Layer Security (TLS).

Beim Starten eines *utmnetss/* Prozesses wird dafür die TLS Umgebung initialisiert. Dabei werden neben dem Laden der openSSL Bibliothek auch die für die Kommunikation benötigten Zertifikate und privaten Schlüssel geladen.

Voraussetzung für einen erfolgreichen Start eines Prozesses vom Typ *utmnetss/* sind folgende Aktionen des Anwenders:

1. Zur Verfügung stellen einer passenden openSSL Bibliothek
2. Setzen der Umgebungsvariablen UTM_SSL_LIBRARY
   Details siehe Kapitel Umgebungsvariable für die Nutzung der openSSL Bibliothek.
3. Erzeugen einer UTM TLS Konfigurationsdatei unter dem <filebase> Verzeichnis

> **Verhalten im Fehlerfall**
>
> Ist die Initialisierung der TLS Umgebung nicht erfolgreich, wird der utmnetssl Prozess beendet und im Anschluss daran auch die UTM Anwendung abnormal beendet.

> **Version openSSL Bibliothek**
>
> Details zur Version der openSSL Bibliothek entnehmen Sie bitte der Freigabemitteilung für openUTM.

UTM TLS Konfigurationsdatei

In der UTM TLS Konfigurationsdatei geben Sie Optionen für die TLS Kommunikation an.

Dies sind die Dateinamen für die Datei mit dem SSL Server-Zertifikat und die Datei mit dem privaten SSL Server-Schlüssel.

Der Inhalt der UTM TLS Konfigurationsdatei wird beim Starten eines jeden *utmnetss/* Prozesses ausgewertet und die dort angegeben Zertifikate/Schlüssel geladen und geprüft.

Im Fehlerfall wird der *utmnetss*l Prozess beendet.

Dateiname

Der Name für UTM TLS Konfigurationsdatei ist fest vorgegeben:

`utm.ssl.conf`

Dateiformat

Die Datei ist zeilenorientiert. Als erste Zeile enthält sie einen Header, danach folgen die Angabe der Optionen. Zusätzlich sind Leerzeilen und Kommentarzeilen erlaubt.

Der Header – erste Zeile – hat folgendes Format:

```
#@(#)  openUTM SSL Config File
```

Kommentarzeilen beginnen mit # oder *.

Die Angabe für das Zertifikat und den Schlüssel erfolgt in folgender Form

```
RSACertificateFile=<dateiname>
```

bzw.

```
RSAKeyFile=<dateiname>
```

RSACertificateFile

Mit dem Parameter *RSACertificateFile* wird eine Datei angegeben, die das RSA-basierte X.509-Server-Zertifikat im PEM-Format enthält.

Diese Datei kann auch den privaten RSA-Server-Schlüssel enthalten. In der Regel werden aber Zertifikat und Schlüssel in getrennten Dateien abgelegt. In diesem Fall wird die Schlüsseldatei mithilfe des Parameters *RSAKeyFile* spezifiziert.

RSAKeyFile

Mit dem Parameter *RSAKeyFile* wird eine Datei angegeben, die den privaten RSA-Server-Schlüssel im PEM-Format enthält.

Für die UTM TLS Konfigurationsdatei gelten folgende Maximalwerte und Regeln:

- Die erste Zeile muss der Header sein
- Maximale Zeilenlänge: 300 Zeichen
- Leerzeilen und Kommentarzeilen werden ignoriert.

> **Muster für UTM TLS Konfigurationsdatei**
>
> Unter dem Verzeichnis utmpfad/ssl wird eine Musterdatei für eine UTM TLS Konfigurationsdatei ausgeliefert.

4.5 Startparameterdatei der Anwendung

Die Startparameterdatei wird vom Administrator der Anwendung mit einem beliebigen Editor erstellt.

In der Startparameterdatei werden aktuelle Ablaufparameter der Anwendung festgelegt. Dazu gehören z.B. die Anzahl der Workprozesse, mit der die Anwendung arbeiten soll, oder Parameter für einen angeschlossenen Resource Manager.

Die Startparameter können in einer oder mehreren Zeilen angegeben werden. Ein Präfix bestimmt, für wen die Startparameter bestimmt sind:

- Startparameter mit dem Präfix "`.UTM`" oder ohne Präfix werden von openUTM selbst interpretiert.
- Startparameter mit dem Präfix "`.RMXA`" leitet openUTM an den angeschlossenen Resource Manager (z.B. ein Datenbanksystem) zur Auswertung weiter (siehe "Startparameter für eine UTM-Datenbank-Anwendung").

Die Reihenfolge der Startparameter für openUTM und das Datenbanksystem ist beliebig, das END-Kommando muss jedoch die Eingabe aller Startparameter abschließen.

Kommentare

Alle Zeilen mit einem „*“ (Stern) oder „#“ (Nummernzeichen) in Spalte 1 werden als Kommentar interpretiert. Kommentare können an beliebiger Stelle in der Startparameterdatei stehen. Damit können Sie z.B. einzelne Startparameter je nach Anwendungslauf aktivieren oder deaktivieren.

4.5.1 Startparameter für openUTM

Die Syntax der UTM-Startparameter ist im Folgenden dargestellt.

```
[.UTM] START   {FILEBASE= filebase | CLUSTER-FILEBASE= cluster_filebase }
               [ ,ADMI-TRACE=  ON | OFF ]
               [ ,ASYNTASKS= number ]
               [ ,BTRACE={{ ON | OFF } | ({ ON | OFF }, length )}]
               [ ,CPIC-TRACE = { TRACE | BUFFER | DUMP | ALL | OFF }]
               [ ,DB-CONNECT-TIME= sec ]
               [ ,DUMP-CONTENT={ ST AN D ARD | EXTENDED } ]
               [ ,DUMP-MESSAGE=( event-typ , event ) ]
               [ ,NODE-TO-RECOVER= node-name ]
               [ ,OTRACE={ ON | (SPI, INT, OSS, SERV, PROT) | OFF } ]

               [ ,RESET-PTC ={ YES | NO } ]

               [ ,STXIT={ ON | OFF } ]
               [ ,SYSPROT=( interval , filename-prefix ) ]
               [ ,TASKS= number ]
               [ ,TASKS-IN-PGWT= number ]
               [ ,TESTMODE={ ON | OFF | FILE } ]

               [ ,TX-TRACE={ ERROR | INTERFACE | FULL | DEBUG | OFF }]

               [ ,XATMI-TRACE={ ERROR | INTERFACE | FULL | DEBUG | OFF }]

[.UTM] END
```

In obiger Syntax werden die Parameter in einer Zeile **ohne Zeilenumbruch** angegeben und jeweils durch ein Komma getrennt.

Sie können die Parameter beim START-Kommando jedoch auch auf mehrere Zeilen verteilen. In diesem Fall müssen Sie in jeder Zeile das Kommando START vor den Parameter stellen.

Syntax-Prüfung beim Start der Anwendung

- Erkennt openUTM bei der Überprüfung der Startparameter einen Syntaxfehler, dann gibt openUTM die Meldung K038 aus, setzt für den betroffenen Startparameter den Standardwert, sofern vorhanden, und startet die Anwendung.
- Bei einem Syntaxfehler im Parameter FILEBASE oder CLUSTER-FILEBASE kann die Anwendung **nicht** gestartet werden, da kein Standardwert für diesen Parameter existiert.

Bedeutung der Kommandos

START — Mit diesem Kommando gibt man die für den Lauf einer UTM-Anwendung erforderlichen UTM-Startparameter an. Die Anwendung wird nach der Eingabe aller Startparameter sofort gestartet.

END — Dieses Kommando schließt die Eingabe der Startparameter ab.

Bedeutung der Operanden

FILEBASE=filebase

Basisname für die KDCFILE und die Benutzer-Protokolldatei in stand-alone Anwendungen. Hier müssen Sie den Namen angeben, unter dem die KDCFILE zum Startzeitpunkt angelegt ist. *filebase* darf maximal 29 Zeichen lang sein, unabhängig davon, ob der Name voll- oder teilqualifiziert angegeben wird.

Bei Angabe eines ungültigen Namens wird der Start der Anwendung abgebrochen.

Wenn Sie den Startparameter FILEBASE angeben, dürfen Sie den Startparameter CLUSTER-FILEBASE nicht angeben.

Für UTM-Cluster-Anwendungen verwenden Sie anstelle des Startparameters FILEBASE den Startparameter CLUSTER-FILEBASE. Der Basisname einer einzelnen Knoten-Anwendung wird bei der UTM-Generierung in der Anweisung CLUSTER-NODE festgelegt.

CLUSTER-FILEBASE=cluster_filebase

Wenn Sie eine UTM-Anwendung als Knoten-Anwendung einer UTM-Cluster-Anwendung starten wollen, geben Sie mit diesem Startparameter den Basisnamen für die Cluster-Dateien an. Hier müssen Sie den Namen angeben, unter dem die Cluster-globalen Dateien zum Startzeitpunkt angelegt sind.

CLUSTER-FILEBASE wirkt Knoten-lokal.

Wenn Sie hier einen ungültigen Namen angeben, wird der Start der Anwendung abgebrochen. Welche Namen gültig sind, entnehmen Sie dem openUTM-Handbuch „Anwendungen generieren “.

Wenn Sie den Startparameter CLUSTER-FILEBASE angeben, dürfen Sie den Startparameter FILEBASE nicht angeben.

ADMI-TRACE=

Ein-/Ausschalten der ADMI-Tracefunktion (= Tracefunktion der Programmschnittstelle zur Administration KDCADMI), siehe auch openUTM-Handbuch „Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen“.

In UTM-Cluster-Anwendungen wirkt ADMI-TRACE Knoten-lokal.

Zum Namen der Trace-Dateien siehe „Trace-Dateien“.

ON — Beim Start der Anwendung wird die ADMI-Tracefunktion eingeschaltet.

OFF — Beim Start der Anwendung bleibt die ADMI-Tracefunktion ausgeschaltet.

Standard: OFF

ASYNTASKS=number

Anzahl der Workprozesse, die max. für asynchrone Vorgänge arbeiten sollen.

In UTM-Cluster-Anwendungen wirkt ASYNTASKS Knoten-lokal.

Standard: in MAX...,ASYNTASKS=*number* festgelegte Anzahl
Minimalwert: 0
Maximalwert: in MAX...,ASYNTASKS=*number* festgelegte Anzahl

BTRACE= Ein-/Ausschalten der BCAM-Tracefunktion.

In UTM-Cluster-Anwendungen wirkt BTRACE Cluster-global.

ON Beim Start der Anwendung wird die BCAM-Tracefunktion eingeschaltet.
Es werden alle Verbindungs-spezifischen Ereignisse in der BCAM-Tracedatei aufgezeichnet.
Die Beschreibung der BCAM-Tracedatei und die Auswertung mit dem Dienstprogramm KDCBTRC finden Sie im openUTM-Handbuch „Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen“.

OFF Beim Start der Anwendung bleibt die BCAM-Tracefunktion ausgeschaltet.

Standard: OFF

length gibt die maximale Länge der Daten an, die bei eingeschalteter BCAM-Tracefunktion aufgezeichnet werden. Wenn die aufzuzeichnenden Daten länger sind, werden die ersten */length* /2 Zeichen und die letzten */length*/2 Zeichen der Daten aufgezeichnet. Diese Länge kann nur über Startparameter festgelegt werden.

Standard: 256
Minimalwert: 32
Maximalwert: 32767

Wenn Sie den BCAM-Trace für die Funktion UPIC Capture einsetzen (siehe auch Abschnitt „UPIC-Conversation mitschneiden (UPIC Capture)“), dann wird empfohlen, den Maximalwert zu verwenden.

CPIC-TRACE=

Ein-/Ausschalten der CPI-C-Tracefunktion (= Tracefunktion der X/Open-Schnittstelle CPI-C), siehe auch openUTM-Handbuch „Anwendungen erstellen mit X/Open-Schnittstellen“.

In UTM-Cluster-Anwendungen wirkt CPIC-TRACE Knoten-lokal.

Zum Namen der Trace-Dateien siehe „Trace-Dateien“.

TRACE Beim Start der Anwendung wird die CPI-C-Tracefunktion mit Level TRACE eingeschaltet. Zu jedem CPI-C-Funktionsaufruf wird der Inhalt der Input- und Output-Parameter ausgegeben. Von den Datenpuffern werden nur die ersten 16 Byte ausgegeben. Die Returncodes der KDCS-Aufrufe, auf die die CPI-C-Aufrufe abgebildet werden, werden ausgegeben.

BUFFER Beim Start der Anwendung wird die CPI-C-Tracefunktion mit Level BUFFER eingeschaltet. Dieser Trace-Level beinhaltet den Level TRACE, die Datenpuffer werden jedoch in voller Länge protokolliert.

DUMP Beim Start der Anwendung wird die CPI-C-Tracefunktion mit Level DUMP eingeschaltet. Dieser Trace-Level beinhaltet den Level TRACE, zusätzlich werden Diagnose-Informationen in die Trace-Datei geschrieben.

ALL Beim Start der Anwendung wird die CPI-C-Tracefunktion mit Level ALL eingeschaltet. Dieser Trace-Level beinhaltet die Level BUFFER, DUMP und TRACE.

OFF Beim Start der Anwendung bleibt die CPI-C-Tracefunktion ausgeschaltet.

Standard: OFF

DB-CONNECT-TIME=sec

Maximale Wartezeit in Sekunden für den Verbindungsaufbau zur Datenbank. Erfolgt während der Wartezeit kein Verbindungsaufbau zur Datenbank, wird die Meldung K078 ausgegeben und der utmwork-Prozess beendet.

In UTM-Cluster-Anwendungen wirkt DB-CONNECT-TIME Knoten-lokal.

Standard: 0 (keine Zeitüberwachung)
Minimalwert: 60
Maximalwert: 3600

DUMP-CONTENT=

gibt an, ob openUTM die Prozess-übergreifenden Speicherbereiche in allen Dumps einer Dump-Dateigeneration, d.h. für alle Prozesse, abzieht oder nur im Dump des Prozesses, der den Anwendungsabbruch verursacht hat.

In UTM-Cluster-Anwendungen wirkt DUMP-CONTENT Knoten-lokal.

STAN**D**ARD

Wenn openUTM eine Dump-Dateigeneration erzeugt, dann sind Prozess übergreifende Speicherbereiche nur im Dump des ersten Prozesses (Verursacher) enthalten. Für die Diagnose ist das normalerweise ausreichend.

Standard: STANDARD

EXTENDED

Die Prozess-übergreifenden Speicherbereiche sind in allen Dumps einer DUMP-Dateigeneration enthalten.

> ¡ Diesen Wert sollten Sie nur auf besondere Anforderung des Service einstellen.

DUMP-MESSAGE= (event-type,event)

Ereignis, bei dem UTM bei eingeschaltetem Testmodus einen UTM-Dump erzeugt. Ein Dump wird nur von dem Prozess erstellt, in der das Ereignis eingetreten ist; die Anwendung wird dabei nicht beendet.

In UTM-Cluster-Anwendungen wirkt DUMP-MESSAGE Cluster-global.

Das Kennzeichen des Dumps ist abhängig vom Ereignis:

Ereignis	Präfix	Beispiel
K- oder P-Meldung	ME gefolgt von der Meldungnummer	`MEP012`
Primärer KDCS-Returncode	CC- gefolgt vom Returncode	`CC-71Z`
Sekundärer KDCS-Returncode	DC gefolgt vom Returncode	`DCK303`
SIGN-Status	SG- gefolgt vom Status	`SG-U01`

Sie können für *event-type*,*event* Folgendes angeben:

- *event-type*=MSG,*event*=K*nnn* (K-Meldung)Der UTM-Dump wird erzeugt, wenn die Meldung K *nnn* auftritt.

  Bei den Meldungsnummern K023, K043, K061, K062 wird nur einmal ein Dump erzeugt, danach wird *event* automatisch zurückgesetzt.
  Bei allen anderen Meldungen wird solange bei jedem Auftreten der Meldungsnummer ein Dump erzeugt, bis der Wert per Administration zurückgesetzt wird.

  Der Wert von DUMP-MESSAGE kann per Administration zurückgesetzt werden, z.B. durch WinAdmin/WebAdmin oder das Kommando KDCDIAG DUMP-MESSAGE=*NONE.

- *event-type*=RCCC,*event*=rccc (Kompatibler KDCS-Returncode)

  Für rccc geben Sie einen KDCS-Returncode (KCRCCC, z.B. 40Z) an.
  Beim Auftreten dieses Returncodes bei einem KDCS-Aufruf wird ein UTM-Dump mit Kennzeichen `CC-40Z` von dem Prozess erzeugt, in dem der Returncode aufgetreten ist. Anschließend wird der Message-Dump für dieses Ereignis automatisch ausgeschaltet.

- *event-type*=RCDC,*event*=rcdc (interner KDCS-Returncode)

  Für rcdc geben Sie einen inkompatiblen KDCS-Returncode (KCRCDC, z.B. KD10) an.
  Beim Auftreten dieses Returncodes bei einem KDCS-Aufruf wird ein UTM-Dump von dem Prozess erzeugt, in dem der Returncode aufgetreten ist. Anschließend wird der Message-Dump für dieses Ereignis automatisch ausgeschaltet.

- *event-type*=SIGN,*event*=sign (SIGN-Statuscode)

  Für sign geben Sie einen SIGNON-Statuscode (KCRSIGN1/2, z.B. U05) an, wobei KCRSIGN1 den Wert U, I, A oder R haben muss. Beim Auftreten dieses Codes beim Anmelden eines Benutzers wird ein UTM-Dump mit Kennzeichen `SG-U05` von dem Prozess erzeugt, in dem der SIGNON-Status aufgetreten ist. Das passiert unabhängig davon, ob in der Anwendung ein Anmelde-Vorgang generiert ist oder nicht.
  Anschließend wird der Message-Dump für dieses Ereignis automatisch ausgeschaltet.

Hinweise:
Bei allen KDCS-Returncodes >=70Z und den zugehörigen inkompatiblen KDCS-Returncodes, bei denen kein PENDER-Dump geschrieben wird (z.B. 70Z/K316), wird auch kein DUMP erzeugt.

Im Administrationskommando KDCDIAG können mit den Parametern DUMP-MESSAGE1, DUMP-MESSAGE2 und DUMP-MESSAGE3 bis zu drei verschiedene Ereignisse angegeben werden, über den Startparameter *DUMP-MESSAGE* dagegen nur ein Ereignis. Außerdem können im Startparameter bei *event-type*=MSG keine Meldungs-Inserts angegeben werden, im Kommando KDCDIAG dagegen bis zu drei Inserts als zusätzliche Bedingungen.

NODE-TO-RECOVER=node-name

Der Parameter ist nur für UTM-Cluster-Anwendungen relevant.

node-name ist der Name der Knoten-Anwendung, für die eine Knoten-Recovery durchgeführt werden soll.
Der Name ergibt sich aus der UTM-Generierung, siehe openUTM-Handbuch „Anwendungen generieren“, Anweisung CLUSTER-NODE, Operand NODE-NAME. Bei jedem Starten, Beenden oder Ausfall einer Knoten-Anwendung gibt die K169-Meldung den *node-name* zusammen mit dem Rechnernamen aus. Auch WinAdmin/WebAdmin zeigen den *nodename* in der Liste Cluster-Nodes an.

Die Knoten-Recovery einer abnormal beendeten Knoten-Anwendung sollte nur durchgeführt werden, wenn ein normaler Knoten-Warmstart nicht bzw. nicht zeitnah möglich ist, weil der Knoten-Rechner ausgefallen ist und kein virtueller Host definiert wurde. Daher ist eine Knoten-Recovery für eine Knoten-Anwendung nur auf einem Knoten-Rechner möglich, auf dem die abnormal beendete Knoten-Anwendung nicht abgelaufen ist.
Die Voraussetzungen zur Nutzung der Knoten-Recovery für UTM-Cluster-Anwendungen, sowie Zweck und Funktion der Knoten-Recovery entnehmen Sie dem Abschnitt „Knoten-Recovery“.

Wenn ein Datenbanksystem die Knoten-Recovery nicht unterstützt, so beendet sich die Knoten-Recovery immer abnormal.

Standard: Leerzeichen, d.h. normaler Anwendungsstart.

OTRACE=

Ein-/Ausschalten der Tracefunktion von OSS.
Der OSS-Trace wird zur Diagnose bei Problemen mit OSI TP-Verbindungen der Anwendung benötigt. Sehen Sie dazu auch das openUTM-Handbuch „Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen“.

In UTM-Cluster-Anwendungen wirkt OTRACE Cluster-global.

ON — Schaltet die OSS-Tracefunktion beim Start der Anwendung ein.
Es werden die Trace-Records der Typen SPI, INT, OSS, SERV und PROT protokolliert. Beim Einschalten der OSS-Tracefunktion erzeugt jeder Prozess der Anwendung seine eigene Trace-Datei.

(SPI, INT, OSS, SERV, PROT)

Schaltet die OSS-Trace-Funktion beim Start der Anwendung ein. Tracesätze des angegebenen Typs werden protokolliert. Die Tracesätze werden in einer beliebigen Reihenfolge angegeben.

SPI
Das XAP-TP System Programming Interface wird protokolliert.

INT
Der interne Ablauf im XAP-TP-Baustein wird protokolliert.

OSS
Die OSS-Aufrufe werden protokolliert.

SERV
Die OSS-internen Trace Records vom Typ O_TR_SERV werden protokolliert.

PROT
Die OSS-internen Trace Records vom Typ O_TR_PROT werden protokolliert.

OFF — Die OSS-Tracefunktion bleibt beim Start der Anwendung ausgeschaltet.

Standard: OFF

RESET-PTC =

Der Parameter ist nur für UTM-Cluster-Anwendungen relevant, wenn NODE-TO-RECOVER ungleich Leerzeichen angegeben wurde.

RESET-PTC legt fest, ob Transaktionen im Zustand PTC („prepare to commit“) bei der Knoten-Recovery zurückgesetzt werden.

Eine Transaktion im Zustand PTC kann Cluster-globale Sperren auf globale UTM-Speicherbereiche halten, die möglicherweise die laufende UTM-Cluster-Anwendung beeinträchtigen.

Transaktionen im Zustand PTC können bei einer Knoten-Recovery nicht vorgesetzt werden, da keine Verbindungen zu Partner-Anwendungen aufgebaut werden. Wenn Transaktionen im Zustand PTC bleiben, beendet sich die Knoten-Recovery abnormal, d.h. ein Online-Import oder ein KDCUPD mit der KDCFILE des ausgefallenen Knoten-Anwendung ist nicht erlaubt und etwaige Cluster-globale Sperren auf UTM-Speicherbereiche bleiben bestehen.

Wenn Sie mögliche Dateninkonsistenzen in Kauf nehmen können, wiederholen Sie daher bei existierenden Transaktionen im Zustand PTC die Knoten-Recovery mit RESET-PTC=YES.

YES — Transaktionen im Zustand PTC werden bei der Knoten-Recovery zurückgesetzt.

NO — Transaktionen im Zustand PTC bleiben bei der Knoten-Recovery bestehen.

Standard: NO

STXIT=

Signalbehandlung bei openUTM aktivieren.

In UTM-Cluster-Anwendungen wirkt STXIT Knoten-lokal.

ON — Signalbehandlung bei openUTM ist eingeschaltet.

Mit STXIT=ON kann auf Unix- und Linux-Systemen auch die Funktionalität „User Signal Routine“ genutzt werden.

Standard: ON

OFF — Die Standard-Fehlerbehandlung für Signale bleibt ausgeschaltet.

> *Unix- und Linux-Systeme*
>
> - STXIT=OFF bewirkt auf Unix- und Linux-Systemen, dass bei jedem Beenden eines Workprozesses zusätzliche Diagnose-Speicherauszüge (gcore-Dumps) im Basis-Dateiverzeichnis der Anwendung erzeugt werden. Die gcore-Dumps werden nur geschrieben, wenn im Dateiverzeichnis `/bin` das Programm *gcore* zur Verfügung steht. Die gcore-Dumps werden durch eine K078 Meldung angekündigt:
>
>  ```
>  K078 STXIT OFF in utmwork: termination of utmwork process
>  creates gcore-dump
>  ```
>
> - Bei STXIT=OFF kann die Funktionalität „User Signal Routine“ nicht genutzt werden.

SYSPROT= — Systemdateien *stderr* und *stdout* umschalten

interval — Umschaltintervall in Tagen

In UTM-Cluster-Anwendungen wirkt *interval* Cluster-global.

Standard: 0 (kein Intervall, das Umschalten erfolgt administrativ)
Maximalwert: 364

flename-prefix

Präfix für den neuen Dateinamen der umgeschalteten Systemdateien. Der Präfix kann ein voll- oder teilqualifizierter Teil eines Dateinamens sein.

In UTM-Cluster-Anwendungen wirkt *filename-prefix* Knoten-lokal.

Standard: utmp

Maximale Länge: 31 Zeichen

Eine vollständige Beschreibung zur Umschaltung der System-Protokolldateien finden Sie im Abschnitt „Systemdateien stderr und stdout“.

TASKS=number

Anzahl der Workprozesse, die für diese Anwendung arbeiten sollen.

In UTM-Cluster-Anwendungen wirkt TASKS Knoten-lokal.

Standard: in MAX...,TASKS=*number* festgelegte Anzahl
Minimalwert: 1 *)
Maximalwert: in MAX...,TASKS=*number* festgelegte Anzahl

*) Falls die Anwendung mit Program Wait generiert ist (d.h. wenn entweder eine TAC-Klasse oder ein TAC mit PGWT=YES generiert ist), oder wenn die Anwendung als UTM-Cluster-Anwendung generiert ist, dann muss für den Startparameter TASKS mindestens 2 angegeben werden.

> **i** Zusätzlich zu der bei TASKS festgelegten Anzahl von Workprozessen werden von UTM für eine Anwendung weitere Workprozesse gestartet, die als UTM-System-Prozesse bezeichnet werden. Die UTM-System-Prozesse sollen Anwendungen, die unter Last laufen, reaktionsfähig halten. Die UTM-System-Prozesse bearbeiten nur ausgewählte Aufträge, die in erster Linie durch kurze Laufzeiten gekennzeichnet sind. Beim Start einer Anwendung werden - abhängig von der Anzahl gestarteter Prozesse (TASKS= number) - bis zu drei zusätzliche UTM-System-Prozesse für die Anwendung gestartet.

TASKS-IN-PGWT=number

Maximale Anzahl der Prozesse, in denen gleichzeitig Teilprogramme mit blockierenden Aufrufen wie z.B. der KDCS-Aufruf PGWT ablaufen dürfen (Operand PGWT= in den KDCDEF-Anweisungen TAC und TACCLASS).

In UTM-Cluster-Anwendungen wirkt TASKS-IN-PGWT Knoten-lokal.

Standard: in MAX...,TASKS-IN-PGWT=*number* festgelegte Anzahl.
Minimalwert: 1 falls MAX...,TASKS-IN-PGWT > 0 generiert wurde, sonst 0.

Maximalwert: in MAX...,TASKS-IN-PGWT=*number* festgelegte Anzahl.

TESTMODE= Testmodus einschalten.

Sehen Sie hierzu auch openUTM-Handbuch „Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen“, Fehlerdiagnose.

In UTM-Cluster-Anwendungen wirkt TESTMODE Cluster-global.

ON Beim Start der Anwendung soll der Testmodus eingeschaltet werden. Im Testmodus werden zusätzliche UTM-interne Plausibilitätsprüfungen durchgeführt und Trace-Informationen im internen Trace-Bereich aufgezeichnet. Der Testmodus sollte nur zur Diagnose von UTM-Fehlern auf Empfehlung des Systemberaters eingeschaltet werden.

> Mit MAX...,IPCTRACE= (siehe openUTM-Handbuch „Anwendungen generieren“, MAX-Anweisung) können Sie bei der KDCDEF-Generierung die Anzahl Trace-Informations-Einträge angeben, die bei TESTMODE=ON geschrieben werden. IPCTRACE sollte nur zur Diagnose von schwerwiegenden UTM-Fehlern auf Empfehlung des Systemberaters versorgt werden.

OFF — Beim Start der Anwendung soll der Testmodus ausgeschaltet bleiben.

Standard: OFF

FILE — Beim Start der Anwendung wird der Testmodus eingeschaltet. Zusätzlich werden bei jedem Überlauf des internen Trace-Bereichs die Diagnosedaten auf Datei geschrieben, um einen evtl. Diagnosedatenverlust zu vermeiden.

Der Dateiname setzt sich zusammen aus dem Basisnamen *filebase* und der PID des jeweiligen Workprozesses, d.h. es wird bei einer UTM-Produktivanwendung pro Workprozess folgende Datei angelegt:

filebase.KTATRC.*pid* (*pid* max. 4-stellig)

TX-TRACE=

Ein-/Ausschalten der TX-Tracefunktion (= Tracefunktion der X/Open-Schnittstelle TX), siehe auch openUTM-Handbuch „Anwendungen erstellen mit X/Open-Schnittstellen“.

In UTM-Cluster-Anwendungen wirkt TX-TRACE Knoten-lokal.

Zum Namen der Trace-Dateien siehe „Trace-Dateien“.

ERROR — Beim Start der Anwendung wird die TX-Tracefunktion mit Level ERROR eingeschaltet. Es werden nur Fehler protokolliert.

INTERFACE — Beim Start der Anwendung wird die TX-Tracefunktion mit Level INTERFACE eingeschaltet. Der Level INTERFACE umfasst den Level ERROR, zusätzlich werden alle TX-Aufrufe protokolliert.

FULL — Beim Start der Anwendung wird die TX-Tracefunktion mit Level FULL eingeschaltet. Der Level FULL umfasst den Level INTERFACE, zusätzlich werden alle KDCS-Aufrufe, auf die die TX-Aufrufe abgebildet werden, protokolliert.

DEBUG — Beim Start der Anwendung wird die TX-Tracefunktion mit Level DEBUG eingeschaltet. Der Level DEBUG umfasst den Level FULL, zusätzlich werden Diagnose-Informationen protokolliert.

OFF — Beim Start der Anwendung bleibt die Tracefunktion der XATMI-Schnittstelle ausgeschaltet.

Standard: OFF

XATMI-TRACE=

Ein-/Ausschalten der XATMI-Tracefunktion (= Tracefunktion der X/Open-Schnittstelle XATMI), siehe auch openUTM-Handbuch „Anwendungen erstellen mit X/Open-Schnittstellen“.

In UTM-Cluster-Anwendungen wirkt XATMI-TRACE Knoten-lokal.

Zum Namen der Trace-Dateien siehe „Trace-Dateien“.

ERROR	Beim Start der Anwendung wird die XATMI-Tracefunktion mit Level ERROR eingeschaltet. Es werden nur Fehler protokolliert.
INTERFACE	Beim Start der Anwendung wird die XATMI-Tracefunktion mit Level INTERFACE eingeschaltet. Der Level INTERFACE umfasst den Level ERROR, zusätzlich werden alle XATMI-Aufrufe protokolliert.
FULL	Beim Start der Anwendung wird die XATMI-Tracefunktion mit Level FULL eingeschaltet. Der Level FULL umfasst den Level INTERFACE, zusätzlich werden alle KDCS-Aufrufe, auf die die XATMI-Aufrufe abgebildet werden, protokolliert.
DEBUG	Beim Start der Anwendung wird die XATMI-Tracefunktion mit Level DEBUG eingeschaltet. Der Level DEBUG umfasst den Level FULL, zusätzlich werden Diagnose-Informationen protokolliert.
OFF	Beim Start der Anwendung bleibt die Tracefunktion der XATMI-Schnittstelle ausgeschaltet. Standard: OFF

Trace-Dateien

Die Trace-Sätze der ADMI-, CPI-C-, TX, und XATMI-Tracefunktion werden standardmäßig in folgende Datei im Verzeichnis *filebase* geschrieben.

KDC.TRC.*trace-type*.*appliname*.*pid* (stand-alone Anwendung) bzw.

KDC.TRC.*trace-type*.*appliname*.*nodename*.*pid* (UTM-Cluster-Anwendung)

trace-type

Kennzeichnet den Trace-Typ:

ADMI ADMI-Trace

CPIC CPI-C-Trace

TX TX-Trace

XATMI XATMI-Trace

appliname
Name der Anwendung

nodename
Name des Cluster-Knotens, auf dem die Knoten-Anwendung läuft.

pid
PID des Prozesses

Beispiel für den Inhalt einer Startparameterdatei einer stand-alone Anwendung

```
.UTM START FILEBASE=/home/utmsmp (Unix- und Linux-Systeme)
.UTM START FILEBASE=C:\utmtest\utmsmp (Windows-Systeme)
.UTM START TASKS=2
.UTM START TASKS-IN-PGWT=1
.UTM START ASYNTASKS=1
.UTM START TESTMODE=OFF
.UTM START ADMI-TRACE=ON
.UTM START BTRACE=OFF
.UTM START OTRACE=OFF
.UTM START STXIT=ON
.UTM END
```

4.6 Kaltstart und Warmstart

Man versteht bei openUTM unter:

- Kaltstart: Starten nach normaler Beendigung der UTM-Anwendung oder nach Neugenerierung.
- Warmstart: Starten nach abnormaler Beendigung der UTM-Anwendung.

Kaltstart bei openUTM

Vor dem ersten Start einer Anwendung haben Sie die KDCFILE mit dem Generierungstool KDCDEF erzeugt. Nach einer Neu-Generierung der KDCFILE oder wenn eine UTM-Anwendung zuvor normal beendet wurde, führt openUTM beim nächsten Start der Anwendung einen Kaltstart durch. Nach erfolgreichem Start meldet openUTM:

`K051 Kaltstart für Anwendung` *`appliname`* `mit UTM V07.0A00 / <Typ des Betriebsystems> / <Bit-Modus des Systems> erfolgreich.`

Warmstart bei openUTM

Wenn eine UTM-Anwendung abnormal beendet wurde, führt openUTM beim nächsten Start dieser Anwendung einen Warmstart durch. Beim Warmstart bringt openUTM die KDCFILE in einen konsistenten Zustand. Nach erfolgreichem Start meldet openUTM:

`K050 Warmstart für Anwendung` *`appliname`* `mit UTM V07.0A00 / <Typ des Betriebsystems> / <Bit-Modus des Systems> erfolgreich.`

Dabei ist zu beachten, dass sich UTM-S und UTM-F im Umfang der Wiederanlauf-Funktionen unterscheiden. Sehen Sie dazu auch im openUTM-Handbuch „Konzepte und Funktionen“, Abschnitt „Fehlertoleranz und Wiederanlauf“.

Hat sich eine UTM-Datenbank-Anwendung abnormal beendet (Systemabbruch oder UTM-Anwendungsabbruch), dann muss der Administrator des Datenbanksystems die Datenbank vor dem Warmstart in einen ordnungsgemäßen Zustand bringen. Beim Warmstart der UTM-Datenbank-Anwendung führt openUTM eine gemeinsame Recovery Phase durch.

4.7 Fehlermeldungen beim Start

Wird der Start einer UTM-Anwendung oder eines Prozesses wegen eines Fehlers abgebrochen, gibt openUTM in der Regel die Meldungen K049 und/oder K078 aus. Die Meldung K078 kann in mehreren Varianten auftreten. Die Bedeutung dieser Meldungen und der darin enthaltenen Returncodes sind ausführlich im openUTM-Handbuch „Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen“ erklärt.

Startfehler können beim Start eines jeden Workprozesses auftreten.

5 UTM-Anwendung beenden

Eine UTM-Anwendung kann

- normal beendet werden durch Administration oder durch das Tool KDCSHUT
- abnormal beendet werden aufgrund von Betriebsmittelengpässen, internen Fehlern in openUTM oder durch Administration

Nach einem Anwendungsende sind evtl. noch systemglobale Betriebsmittel zu löschen, bevor die Anwendung erneut gestartet wird.
Sehen Sie dazu Abschnitt „Das Tool KDCREM“.

Beim Beenden einer UTM-Cluster-Anwendung sind einige Besonderheiten zu beachten, siehe Abschnitt „Shutdown einer UTM-Cluster-Anwendung“.

5.1 UTM-Anwendung per Administration normal beenden

Eine UTM-Anwendung beendet der UTM-Administrator normal, indem er z.B. folgendes UTM-Administrationskommando an einem Terminal eingibt:

`KDCSHUT GRACE,TIME=`*time*

oder

`KDCSHUT WARN,TIME=`*time*

oder

`KDCSHUT NORMAL`

In Anwendungen, die verteilte Transaktionsverarbeitung verwenden, sollte eine Anwendung immer mit KDCSHUT GRACE oder WARN beendet werden, denn dies erlaubt eine geordnete Beendigung der offenen verteilten Transaktionen.

Statt des Kommandos KDCSHUT kann auch die entsprechende Funktion bei WinAdmin/WebAdmin oder KDCADMI verwendet werden.

Beim Beenden der Anwendung führt openUTM die folgenden Aktionen durch:

- Es werden alle Aufträge abgearbeitet, die noch in der UTM-Warteschlange sind.
- Die Verbindungen zu allen Kommunikationspartnern der Anwendung werden abgebaut.
- KDCFILE, System-Protokolldatei und Benutzer-Protokolldatei werden in einen konsistenten Zustand gebracht und ordnungsgemäß geschlossen.
- Alle Prozesse der Anwendung werden beendet.

Um eine UTM-Anwendung normal zu beenden, können Sie an Stelle des Kommandos KDCSHUT auch die entsprechende Funktion bei WinAdmin/WebAdmin oder an der Administrations-Programmschnittstelle verwenden.

5.2 Das Tool KDCSHUT - UTM-Anwendung auf Shell-Ebene normal beenden

Das Tool KDCSHUT ist eine einfache Methode, die Anwendung zu beenden, ohne sich als Administrator bei der Anwendung anmelden zu müssen. Das Tool KDCSHUT wirkt wie das Administrationskommando KDCSHUT N oder KDCSHUT G,TIME= bzw. das Kommando KDCSHUT W, TIME=.

Das Tool KDCSHUT wird wie folgt aufgerufen:

Unix- und Linux-Systeme: *utmpfad*`/ex/kdcshut` *filebase* [*time* `[G]`]

Windows-Systeme: *utmpfad*`\ex\kdcshut` *filebase* [*time* `[G]`]

filebase

Basisname der Anwendung;

time

Wartezeit in Minuten, bis sich die Anwendung beendet.
Maximale Wartezeit: 60 Minuten

G

die Anwendung wird mit einem Graceful Shutdown beendet (siehe openUTM-Handbuch „Anwendungen administrieren“).

5.3 Dienst beenden auf Windows-Systemen

Wenn eine UTM-Anwendung als Dienst gestartet wurde, dann kann sie entweder so beendet werden, als ob sie nicht als Dienst gestartet wurde (d.h. z.B. mit dem Tool KDCSHUT), oder sie kann als Dienst beendet werden. Dazu gehen Sie ähnlich wie beim Starten vor:

1. Melden Sie sich unter einer Windows-Kennung an, die Administrationsberechtigung besitzt.
2. Rufen Sie die Systemsteuerung auf mit *Start - Systemsteuerung* .
3. Klicken Sie auf *Verwaltung*, dann auf *Dienste* und markieren Sie den gewünschten UTM-Dienst; dieser hat immer den Namen `openUTM` *servicename*.
4. Drücken den Button *Beenden*; der Dienst und damit die Anwendung werden normal beendet.

Wird das Windows-System heruntergefahren, dann werden der Dienst und damit die Anwendung ebenfalls normal beendet.

5.4 UTM-Anwendung abnormal beenden

Eine UTM-Anwendung kann durch folgende Ereignisse abnormal beendet werden:

- UTM-interner Fehler
- Fehler in der Systemumgebung und Shutdown des Unix- und Linux-Systems
- UTM-Administrationskommando KDCSHUT KILL (oder durch die entsprechende Funktion bei WinAdmin /WebAdmin oder KDCADMI)
- Anwenderfehler

Bei abnormaler Beendigung werden folgende Aktionen ausgeführt:

- Alle Transaktionen, die gerade von den einzelnen Workprozessen bearbeitet werden, werden abgebrochen.
- Die Verbindungen zu allen Kommunikationspartnern der Anwendung werden abgebaut.
- Für jeden Workprozess der Anwendung wird ein UTM-spezifischer Speicherauszug (UTM-Dump) erzeugt. Siehe dazu auch openUTM-Handbuch „Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen“.
- Alle Prozesse der Anwendung werden beendet und alle Dateien werden geschlossen. Es wird nicht versucht, die KDCFILE in einen konsistenten Zustand zu bringen. Dies geschieht erst bei einem erneuten Start der Anwendung.

Nach einem abnormalen Beenden der Anwendung sollten Sie als erstes die Ursache für den Abbruch feststellen. Dazu suchen Sie im Protokoll der Workprozesse auf *stdout* die Meldung K060. Diese Meldung enthält als Insert den Dump-Fehlercode, der genaue Auskunft über die Abbruch-Ursache gibt. Die Dump-Ursache können Sie auch als Teil des Dateinamens der UTM-Dumps finden. Die Bedeutung des Dump-Fehlercodes ist im openUTM-Handbuch „Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen“ bei der Meldung K060 beschrieben. Dabei gibt es drei Möglichkeiten:

- Der Dump-Fehlercode sagt aus, dass ein KDCDEF-Operand verändert werden muss.Dann ist die KDCFILE neu zu generieren. Wenn Sie die Benutzerdaten im Pagepool erhalten wollen, gehen Sie wie folgt vor:
  - Warmstart mit ASYNTASKS=0, TASKS=1
  - Anwendung normal beenden mit KDCSHUT NORMAL
  - alte KDCFILE sichern
  - neue KDCDEF-Generierung mit dem veränderten Operanden
  - Übertragung der Benutzerdaten mit KDCUPD aus der alten in die neue KDCFILE
  - Start der Anwendung mit der neuen aktualisierten KDCFILE
- Der Dump-Fehlercode nennt als Ursache
  - Speicherengpass
  - Datenbanksystem ist z.Zt. nicht verfügbar

  Ist der Fehler behoben, können Sie die Anwendung erneut starten, openUTM führt dann automatisch einen Warmstart durch.

- Es liegt ein Systemfehler vor. In diesem Fall erstellen Sie Diagnoseunterlagen und schreiben eine Fehlermeldung an den Systembetreuer. Dazu bereiten Sie die UTM-Dumps aller Workprozesse der Anwendung mit dem Tool KDCDUMP auf. Weitere Unterlagen sind die *stdout*- und *stderr*-Systemdateien, die *gcores* (bei Unix- und Linux-Systemen), das *utmwork*-Programm, die KDCDEF-Steueranweisungen und eine Auswertung der System-Protokolldatei.

  Ein Warmstart mit derselben KDCFILE ist in diesem Fall nicht immer erfolgreich. Kann kein Warmstart durchgeführt werden, müssen Sie die KDCFILE mit KDCDEF neu generieren.

Bei einer abnormalen Beendigung ist vor dem Neustart der Anwendung das Tool KDCREM aufzurufen (siehe folgenden Abschnitt).

5.5 Das Tool KDCREM

Mit dem Tool KDCREM werden nach einem Anwendungsende evtl. noch vorhandene Semaphore und Shared Memories sowie Statusinformationen der Anwendung, die in der Datei `applifile` im *utmpfad* enthalten sind, gelöscht bzw. zurückgesetzt. Sehen Sie dazu auch den Abschnitt „Systemglobale Betriebsmittel einer Anwendung“.

> **ACHTUNG!**
>
> - Nach der abnormalen Beendigung des Mainprozesses einer UTM-Anwendung (z.B. durch einen Fehler im Betriebssystem, System-Shutdown oder das Signal SIGKILL) **muss** KDCREM vor einem Anwendungsstart aufgerufen werden.
> - Das Tool KDCREM beendet eine laufende UTM-Anwendung abnormal und ohne Warnung!

Aufruf von KDCREM

Unix- und Linux-Systeme: *utmpfad*`/ex/kdcrem` *filebase*

Windows-Systeme: *utmpfad*`\ex\kdcrem` *filebase*

filebase ist der Basisname der Anwendung, deren Semaphore, Shared Memories und Statusinformationen in der Datei `applifile` in *utmpfad* gelöscht werden sollen.

6 UTM-Datenbank-Anwendung

Dieses Kapitel gibt einen zusammenhängenden Überblick über das, was für den Einsatz von Datenbanken (= Resource Managern) unter openUTM zu beachten ist. openUTM verwendet für die Kopplung die von X/Open genormte XA-Schnittstelle.

openUTM unterstützt auf Unix- , Linux- und Windows-Systemen die Koordination mit folgendem Datenbank-System:

- Oracle

Nähere Information über das Konzept der koordinierten Zusammenarbeit finden Sie im openUTM-Handbuch „Konzepte und Funktionen“.

6.1 UTM-Datenbank-Anschluss generieren

Sie müssen den UTM-Datenbank-Anschluss in der KDCDEF-Anweisung RMXA generieren. Dort geben Sie an:

- den Namen der *xa_switch_t*-Struktur wie er von der eingesetzten Datenbank vorgegeben ist.
- Bei Windows-Systemen: ob die *xa_switch_t*-Struktur mit *dllimport* adressiert wird. Bei der Kopplung mit Oracle muss immer über *dllimport* adressiert werden.
- Zugangsdaten für die Datenbank (Username, Passwort).

  Diese Angaben sind optional. Wenn Sie die Zugangsdaten in der UTM-Generierung hinterlegen wollen, dann müssen Sie im Openstring für den Benutzernamen und das Passwort Platzhalter vorsehen.

Es gibt in der Regel einen statischen und einen dynamischen XA Switch. Eine Datenbank kann eine oder auch beide Varianten anbieten. Wenn die Datenbank einen dynamischen XA Switch anbietet, sollte man diesen nutzen, da hierdurch die Ressourcenbelegung im Datenbanksystem minimiert wird.

Näheres zur RMXA-Anweisung finden Sie im openUTM-Handbuch „Anwendungen generieren“ bei der Beschreibung der RMXA-Anweisung.

Konfiguration des dynamischen XA-Switches von Oracle auf Windows-Systemen

Wenn auf Windows-Systemen für die XA-Kopplung mit Oracle der dynamische XA-Switch *xaoswd* verwendet wird, müssen Sie zuerst die Windows Registry um den Eintrag ORA_XA_REG_DLL erweitern.

Dazu führen Sie nach der Installation von Oracle Client und openUTM folgende Schritte aus:

1. Starten Sie den Windows Registry Editor in der Windows-Eingabeaufforderung mit dem Kommando `regedit`.
2. Erlauben Sie im Fenster *Benutzerkonten* der Systemsteuerung den System-Zugriff.
3. Wählen Sie unter `HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE` den Schlüssel `KEY_OraClientnnHome1`, z. B. `KEY_OraClient12Home1`.
4. Wählen Sie im Menü *Bearbeiten* den Befehl *Neu* -> *Wert der erweiterbaren Zeichenfolge* und tragen Folgendes ein:
   - Bei *Name* (Voreinstellung `Neuer Wert #1`) den Wert `ORA_XA_REG_DLL`
   - Bei *Typ* den Wert `REG_EXPAND_SZ`.
   - Bei *Daten* den vollqualifizierten Namen der installierten dynamischen openUTM-Bibliothek (*libwork.dll*), z:B. `C:\openUTM-Server\ex\libwork.dll`.
5. Bestätigen Sie mit OK und beenden Sie den Registry Editor.

Damit ist das Windows System mit openUTM und Oracle für den Einsatz mit der dynamischen Oracle XA-Kopplung konfiguriert.

Wichtig!
Um sicherzustellen, dass die XA-Kopplung zwischen openUTM und Oracle korrekt konfiguriert wurde, muss einmalig Folgendes überprüft werden: *xa_reg*-Aufrufe werden bei der Protokollierung der XA-Schnittstelle von openUTM bei der Ausführung von Updates in der Oracle-Datenbank nach folgendem Muster protokolliert:

```
kdcrtxa-<pid>: AX_CALL-1: ax_reg, XID.GTRID=<utm-gtrid-value>
```

Wenn die openUTM XA-Debug-Informationen keine Einträge obiger Art enthalten, oder wenn die Oracle XA-Debug-Informationen *xa_reg*-Aufrufe enthalten, die NULL-XIDs anzeigen, liegt ein schwerwiegendes Konfigurations-Problem vor!

Sie aktivieren die Protokollierung der XA-Schnittstelle über die Funktion „XA-Debug“ (z.B. mit KDCDIAG), siehe openUTM-Handbuch „Anwendungen administrieren“ und Abschnitt „Debug-Parameter“.

6.2 UTM-Datenbank-Anwendung binden auf Unix- und Linux- Systemen

Für die UTM-Datenbankkopplung müssen Sie zusätzliche Module in den UTM-Workprozess einbinden. Im Folgenden sind diese Module für die einzelnen Datenbanksysteme aufgelistet. Die Richtigkeit der Namen von Modulen, die Bestandteil des jeweiligen Datenbanksystems sind, sollten Sie anhand der Benutzerhandbücher zu diesem Datenbanksystem überprüfen.

> **i** Sie können sich die Arbeit erleichtern, indem Sie die Beispielanwendung verwenden, die mit openUTM ausgeliefert wird, siehe auch "Beispielanwendung für Unix- und Linux-Systeme". Mit Hilfe dieser Beispielanwendung lässt sich auf einfache Art und Weise eine UTM-Datenbank-Anwendung erstellen, die alle jeweils benötigten Datenbankbibliotheken enthält. Diese Datenbank-Anwendung kann als Vorlage für Ihre eigene Anwendung dienen; Sie können z.B. das dabei erzeugte Makefile anpassen, siehe Abschnitt „Binden mit Makefile (Unix- und Linux-Systeme)“.

Kopplung mit Oracle

Für für Kopplung mit Oracle müssen zusätzlich eine Reihe von Oracle-Modulen eingebunden werden. Welche dies sind, können Sie mit Hilfe der mit Oracle ausgelieferten Beispielprogramme und -Prozeduren ermitteln. Die Oracle Client-Bibliothek lautet *$ORACLE_HOME/lib/libclntsh.so*. Zusätzlich müssen Sie bei ESQL-COBOL Anwendungen das Objekt-Modul $*ORACLE_HOME/precomp/lib/cobsqlintf.o* einbinden.

Die Liste der zusätzlich von Oracle benötigten Systembibliotheken steht in der Datei *$ORACLE_HOME/lib/sysliblists* .

Das Präprozessor Flag `release_cursor=yes` ist unbedingt zu setzen. Siehe dazu das Benutzerhandbuch zu Oracle.

6.3 UTM-Datenbank-Anwendung binden auf Windows- Systemen

Auf Windows-Systemen erstellen Sie eine UTM-Datenbank-Anwendung auf die gleiche Art und Weise wie eine UTM-Anwendung, siehe "Anwendungsprogramm erzeugen auf Windows-Systemen". Sie müssen nur die folgenden zusätzlichen Optionen einstellen:

- Bei den Optionen des Visual Studios müssen Sie das Verzeichnis angeben, in denen die Datenbank-Bibliotheken stehen, siehe "Optionen des Visual Studios einstellen (Windows-Systeme)".
- Bei den Binderoptionen müssen Sie die Namen der benötigten Datenbank-Bibliotheken angeben, siehe "Anwendung übersetzen und binden (Windows-Systeme)".

Damit werden beim Aufruf des Binders die richtigen Datenbank-Bibliotheken dazugebunden, siehe "Anwendung übersetzen und binden (Windows-Systeme)".

> **i** Sie können sich mit Hilfe des Quick Start Kit eine UTM-Datenbank-Anwendung erzeugen, siehe "openUTM Quick Start Kit für Windows-Systeme". Das Quick Start Kit wird mit openUTM ausgeliefert.

6.4 UTM-Datenbank-Anwendung starten und beenden

Eine UTM-Datenbank-Anwendung wird wie eine UTM-Anwendung gestartet und beendet, d.h. durch Starten und Beenden des UTM-Anwendungsprogramms.

6.4.1 Startparameter für eine UTM-Datenbank-Anwendung

Für den Start einer UTM-DB-Anwendung müssen neben den Startparametern für openUTM auch die Startparameter für die Datenbank angegeben werden. Dabei gilt folgendes Schema:

`.UTM ...`

Startparameter für openUTM, siehe Abschnitt „Startparameterdatei der Anwendung“.

`...`

`.RMXA [xa-inst-name]...`

Startparameter für das Datenbanksystem (XA Resource Manager). Diese sind im Handbuch für das DB-System beschrieben. Beispiele finden Sie in der Unix- bzw. Linux-Beispielanwendung und für Windows im Quick Start Kit, sowie in den folgenden Abschnitten.

`END`

Startparameter für das Datenbanksystem haben das Präfix `.RMXA`. openUTM leitet dann beim Start der Anwendung diese Startparameter an den Resource Manager weiter. Der Resource Manager wird während der Startphase des UTM-Workprozesses von openUTM geöffnet.

6.4.1.1 Openstring und Closestring

In der Startparameterdatei definieren Sie die Datenbank (Instanz des Resource Managers) durch einen Openstring und, falls der Resource Manager diesen benötigt, einen Closestring. Das Datenbanksystem Oracle benötigt **keinen** Closestring.

Die Angaben für Openstring und Closestring sind vom jeweiligen Resource Manager zur Verfügung zu stellen. Damit hängt auch die Syntax dieser Angaben vom jeweiligen Resource Manager ab und ist dem Handbuch zu dem verwendeten Resource Manager zu entnehmen. openUTM übergibt die Strings aus der Startparameterdatei an den Resource Manager, ohne sie zu überprüfen. Jeder String muss durch doppelte Hochkommata begrenzt werden und darf maximal 255 Zeichen lang sein.

Openstring und Closestring werden in **einer** Zeile der Startparameterdatei angegeben, durch ein Leerzeichen voneinander getrennt (Closestring nur, wenn vom Resource Manager benötigt):

`.RMXA RM="`*name*`", OS="`*openstring*`" [CS="`*closestring*`"]`

Die Zeile darf insgesamt maximal 560 Zeichen lang sein.

Zugangsdaten für die Oracle-Datenbank angeben

Sie können die Zugangsdaten für die Datenbank (Benutzername, Passwort) wahlweise in den Startparametern oder in der UTM-Generierung angeben, wobei Folgendes gilt:

- Die Angabe in den Startparametern hat Vorrang vor der UTM-Generierung. Sie können sowohl beides Benutzernamen, Passwort als auch nur das Passwort oder nur den Benutzernamen angeben. In allen drei Fällen wird beim Start der UTM-Anwendung die Meldung K238 ausgegeben.
- Wenn die Zugangsdaten aus der UTM-Generierung verwendet werden sollen (siehe "Beispiel für Oracle-Startparameter"), dann müssen Sie in den Startparametern die Platzhalter *UTMUSER (für den Benutzernamen) und *UTMPASS (für das Passwort) angeben. Beim Verbindungsaufbau zu Oracle ersetzt openUTM automatisch die generischen Platzhalter *UTMUSER und *UTMPASS durch den Benutzernamen und das Passwort, die in der RMXA-Anweisung generiert wurden.

Aus Sicherheitsgründen wird empfohlen, die Zugangsdaten in der UTM-Generierung zu hinterlegen.

Die Zugangsdaten lassen sich später per Administration ändern.

6.4.1.2 Mehrere Instanzen

Die UTM-Anwendung kann mehrere Instanzen (Datenbanken) des Resource Managers betreiben, sofern der Resource Manager den Multi-Instanzen-Betrieb unterstützt. Dazu müssen Sie für jede Instanz einen eigenen Openstring angeben. Jeder Openstring ist in einer eigenen Zeile in der Startparameterdatei anzugeben. Der Name des Resource Managers *name* muss in den einzelnen Startparameter-Anweisungen übereinstimmen. Für die Openstrings (Datenbanken) sind verschiedene Namen anzugeben (angegeben im Parameter DB= innerhalb der Strings, z.B +DB=DBNAME1 und +DB=DBNAME2).

Die allgemeine Syntax für Startparameter von mehreren XA-Instanzen eines Resource Managers ist:

```
.RMXA[xa-inst-name1] RM="name",OS="openstring1"
.RMXAxa-inst-name2 RM="name",OS="openstring2"
.RMXAxa-inst-name3 RM="name",OS="openstring3"
...
```

Hierbei muss der Wert von *xa-inst-name1*, *xa-inst-name2*, ... jeweils zum Generierungswert des Parameters XA-INST-NAME einer RMXA-Anweisung der KDCDEF-Generierung passen.

Um in jedem Fall die Eindeutigkeit der Zuordnung von XA-Instanzen zur UTM-Generierungsinformation sicherzustellen, beachten Sie folgende drei Regeln:

- Verwenden Sie pro UTM-Anwendung maximal einen leeren XA-Instanznamen und maximal einen leeren Oracle-DB-Namen.
- Alle XA-Instanz-Namen müssen in der UTM-Anwendung eindeutig sein.
- Alle Oracle-DB-Namen müssen in der UTM-Anwendung eindeutig sein.

Benötigt der Resource Manager einen Closestring, dann ist für jede Instanz zusätzlich ein Closestring anzugeben.

> **ACHTUNG!**
>
> Neben einem gekoppelten Datenbankanschluss (bei openUTM generiert) darf es keinen ungekoppelten Datenbankanschluss geben.

Beispiele für Startanweisungen für das Datenbanksystem, mit denen openUTM gekoppelt werden kann, entnehmen Sie dem Abschnitt „Beispiel für Oracle-Startparameter“.

Beispiel

Dieses Beispiel zeigt den Anschluss zweier Oracle-XA-Datenbanken an openUTM (Unix/Linux/Windows).

Startparameter:

```
.RMXA
RM="Oracle_XA",OS="Oracle_XA+Acc=P/*UTMUSER/*UTMPASS+SqlNet=RAC2+SesTm=60+
LogDir=.+DbgFl=0"
.RMXAEVE
RM="Oracle_XA",OS="Oracle_XA+DB=EVESDB+Acc=P/*UTMUSER/*UTMPASS+SqlNet=RAC1
+SesTm=60+LogDir=.+DbgFl=0"
```

Der erste Startparameter enthält nur das Präfix „.RMXA", d.h. einen leeren XA-Instanznamen, und wird deshalb dem ersten bei openUTM generierten XA-Resource Manager ohne XA-INST-NAME-Parameter zugeordnet. D.h. die erste XA-Instanz baut eine Verbindung mit den entsprechend generierten Zugangsdaten ("MaxTheSuperman", ...) auf.

Der zweite Startparameter enthält das Präfix „.RMXAEVE ", d.h. die zweite XA-Instanz wird dem generierten XA-Resource Manager mit XA-INST-NAME=EVE zugeordnet, und die dafür generierten Zugangsdaten („Eve", ...) werden von dieser XA-Instanz verwendet.

KDCDEF-Generierung:

```
RMXA XASWITCH=xaoswd,USERID='MaxTheSuperman',PASSWORD='PasswordOfMax'
RMXA XASWITCH=xaoswd,USERID='Eve',PASSWORD='PasswordOfEve',XA-INST-
NAME=EVE
```

6.4.1.3 Beispiel für Oracle-Startparameter

Oracle benötigt nur einen Openstring, keinen Closestring.

Ein Multi-Instanzen-Betrieb ist möglich, d.h in der Startparameterdatei einer UTM-Anwendung dürfen mehrere Openstrings für den Resource Manager angegeben werden.

In der Startparameterdatei können Sie für eine Oracle-Datenbank z.B. die folgenden Startparameter angeben:

```
.RMXA Oracle_XA OS="Oracle_XA+Acc=P//+SesTm=60"
```

Diese Anweisung müssen Sie in eine Zeile ohne Zeilenvorschub schreiben.

Wie Sie DEBUG-Informationen für den Anschluss an die Datenbank bekommen, ist in Abschnitt „Debug-Parameter“ beschrieben.

Im Openstring sind nur die Pflichtparameter aufgeführt. Zusätzlich können Sie weitere optionale Parameter angeben. Diese sind weiter unten aufgelistet.
Die Parameter im Openstring werden durch das Zeichen "+" getrennt.

Bedeutung der Pflichtparameter:

Oracle_XA	Von Oracle vorgegebener Name des Resource Managers, wie in der *xa_switch*-Struktur enthalten.
Acc=P//	Informationen für die Zugriffskontrolle der Datenbank (User Access Information). Wird wie im Beispiel `Acc=P//` angegeben, so werden weder Benutzerkennung noch Kennwort zur Zugriffskontrolle übergeben.
	Eine Oracle-Datenbank kann auch Datenbank-spezifische Informationen (Benutzer *user* und Kennwort *pwd*) anfordern, die mit `Acc=P/` *user* `/` *pwd* übergeben werden müssen. Informationen dazu entnehmen Sie bitte dem Handbuch zu Oracle.
SesTm=	Maximale Zeit in Sekunden, die für eine Transaktion zur Verfügung steht (im Beispiel 60 s). Mögliche Angaben für *SesTm* entnehmen Sie bitte dem Handbuch zu Oracle.
	Die Angabe von `SesTM=0` bedeutet, dass die Transaktionsdauer nicht beschränkt wird. Es wird deshalb empfohlen, für *SesTm* einen Wert > 0 anzugeben.

Die im Folgenden aufgelisteten Parameter sind optional. Ihre Bedeutung und die möglichen Angaben entnehmen Sie bitte dem Handbuch zu Oracle.

DB=	Name der Oracle-Datenbank
	Den Parameter müssen Sie angeben, wenn die UTM-Anwendung mit mehreren Oracle-Datenbanken gekoppelt werden soll (Multi-Instanzen-Betrieb).
GPwd=P/	Gruppenkennwort, wird in der Form `GPwd=P/` *kennwort* angegeben
LogDir=	Pfadname des Logging-Dateiverzeichnisses
MaxCur=	Maximale Anzahl offener Cursor
SqlNet=	Network Connection String

Bei abnormalem Anwendungsende sind keine Maßnahmen notwendig, denn vor dem erneuten Start der UTM-Anwendung führt openUTM automatisch eine gemeinsame Recovery Phase durch.
Beachten Sie auf Windows-Systemen, dass die RMXA-Anweisung bei der Generierung mit KDCDEF den Operand DLLIMPORT=YES benötigt.

Informationen über die einzubindenden Objekte bzw. Bibliotheken und Startparameter (Openstring) entnehmen Sie bitte der Dokumentation zu Oracle.

Oracle-Benutzername und Oracle-Passwort aus der UTM-Generierung verwenden

Die Zugangsberechtigung für eine Oracle Datenbank sollte aus Sicherheitsgründen per KDCDEF-Generierung festgelegt werden.

Beachten Sie bitte Folgendes:

- Der Oracle-Benutzername für die Verbindung zu Oracle und das dazugehörige Oracle-Passwort müssen in KDCDEF generiert sein (KDCDEF-Anweisung RMXA, Operanden USERID und PASSWORD).

  Das Oracle-Passwort wird als Hashcode in den UTM-Systemtabellen abgespeichert (maskiert) und ist im UTM-Dump deshalb nicht im Klartext enthalten.
- Im Openstring des Startparameters geben Sie anstelle des Oracle-Benutzernamens den Platzhalter *UTMUSER und anstelle des Oracle-Passworts den Platzhalter *UTMPASS an. Diese Platzhalter werden nach folgenden Regeln ersetzt:
  - Enthält der Openstring mindestens einen der Platzhalter *UTMUSER oder *UTMPASS, so ersetzt UTM beim xa_open() Aufruf die Platzhalter durch die Werte, die für dieses Datenbanksystem generiert sind. D.h. im Openstring wird *UTMUSER durch den generierten Oracle-Benutzernamen und *UTMPASS durch das generierte Oracle-Passwort ersetzt.
    Aus Sicherheitsgründen wird das Oracle-Passwort erst unmittelbar vor der Verwendung beim xa_open() Aufruf in Klartext umgewandelt und sofort nach dem xa_open() Aufruf wieder im Prozessspeicher gelöscht.
  - Wenn der Openstring des Startparameters weder *UTMUSER noch *UTMPASS enthält, wird der Openstring unverändert an den xa_open() Aufruf übergeben.

Beachten Sie bitte, dass die Groß-/Kleinschreibung signifikant ist!

Beispiel

Sie möchten den Oracle-Benutzernamen und das Oracle-Passwort aus der UTM-Generierung verwenden:

```
OS="Oracle_XA+SqlNet=O11+ACC=P/*UTMUSER/*UTMPASS+DbgFl=15"
```

Verhalten bei nicht generierten Oracle-Zugangsdaten

- Wurden die Operanden USERID und PASSWORD bei der UTM-Generierung nicht angegeben und verlangt die Oracle-Datenbank einen Benutzernamen und/oder ein Passwort, dann wird der Verbindungsaufbau zur Datenbank abgelehnt.
- Falls Sie im Startparameter *UTMUSER bzw. *UTMPASS angeben, obwohl die Operanden USERID und PASSWORD bei der UTM-Generierung nicht angegeben wurden, dann verwendet UTM einen leeren Oracle-Usernamen bzw. ein leeres Oracle-Passwort. D.h. der Verbindungsaufbau zur Datenbank wird in der Regel nicht erfolgreich sein.

6.4.2 Startparameter für Failover mit Oracle® Real Application Clusters

Eine UTM-Anwendung kommuniziert mit Oracle Real Application Clusters über die XA-Schnittstelle. Kann in einem Failover-Fall ein XA-Aufruf von Oracle nicht mehr ordnungsgemäß durchgeführt werden, quittiert dies Oracle mit dem Rückgabewert „XAER_RMFAIL“.

Im Normalfall, d.h. wenn die Failover-Unterstützung nicht eingeschaltet ist, schließt openUTM aus dieser Meldung, dass eine weitere Zusammenarbeit mit dieser Datenbank nicht mehr möglich ist und bricht den Anwendungslauf ab.

Um diesen Abbruch zu verhindern, geben Sie bei den Startparametern unter .RMXA zusätzlich den Wert RAC=Y an und steuern das Failover-Verhalten mit den optionalen Parametern RAC_retry und RAC_recover_down:

```
.RMXA RM="Oracle_XA",OS="openstring" ,RAC=Y
                                    [,RAC_retry=nnn ] [,RAC_recover_down={Y|N}]
```

RAC=Y

schaltet die Failover-Unterstützung bei der Kopplung der UTM-Anwendung mit Oracle® Real Application Clusters ein.

RAC=N

schaltet die Failover-Unterstützung aus.
Standard: N

RAC_retry=nnn

nnn gibt die Anzahl der Versuche an, die openUTM unternehmen soll, um sich erneut mit der Datenbank zu verbinden und einen Recover-Auftrag auszuführen.

Konnte für eine Transaktion im Zustand „Prepare-to-Commit“ der Commit-Auftrag wegen eines Failover nicht ausgeführt werden, verbindet sich openUTM erneut mit der Datenbank und führt einen Recover-Auftrag aus. Ist die aktuelle XID in der Liste der gelieferten XIDs enthalten, wird von openUTM ein Commit-Auftrag für die XID, also für die aktuelle Transaktion, ausgeführt. Ist die XID nicht in der Liste enthalten, wird von openUTM ein *xa_close* durchgeführt. Anschließend versucht openUTM erneut, sich mit der Datenbank zu verbinden und einen Recover-Auftrag auszuführen.

Standard: 1

RAC_recover_down=

Legt fest, wie sich openUTM verhält, wenn die Transaktion nach der mit RAC_retry= festgelegten Anzahl von Versuchen nicht endgültig abgeschlossen werden konnte, d.h. nicht in den Zustand „Commit“ versetzt werden konnte.

N N openUTM geht davon aus, dass die Transaktion bei Oracle Real Application Clusters nicht mehr bekannt. ist. Die Transaktion wird als „Commit“ angenommen und openUTM setzt den Anwendungslauf fort.

Standard: N

Y openUTM beendet den Anwendungslauf abnormal und erzwingt damit einen Warmstart, um für Datenkonsistenz zu sorgen.

Verhalten von openUTM im Failover-Fall

Wenn Sie Failover-Unterstützung eingeschaltet haben, dann verhalten sich openUTM und das Datenbanksystem wie folgt:

- Der Anwendungsabbruch wird vermieden, wenn ein Failover zu einem noch aktiven Knoten des Oracle® Real Application Clusters möglich ist.
- Bei Verbindungsverlust am Transaktionsende zwischen „Prepare“ und „Commit“ wird ein „Reconnect“ mit Recovery durchgeführt und im Erfolgsfall der „Commit“ über diese neue Verbindung wiederholt.
- Wenn zum Failover-Zeitpunkt noch Transaktionen offen sind, dann kann dies trotz eingeschalteter Failover-Unterstützung zu Problemsituationen und entsprechenden Fehlermeldungen führen (z.B. den Returncode ORA-25402 - transaction must rollback). Dies liegt daran, dass Oracle® Real Application Clusters beim Failover-Fall keine offenen Transaktionen migrieren kann. Diese Transaktionen müssen vom UTM-Anwendungsprogramm zurückgesetzt werden, siehe auch „UnterbrocheneTransaktionen“.
  Offene Mehrschritt-Transaktionen (d.h. nach PEND KP) werden im Failover-Fall durch das Datenbank-System zurückgesetzt, ohne dass openUTM dies beeinflussen kann.
  Das Datenbank-System wird nach dem Rollback automatisch neu verbunden; anschließend können wieder neue Transaktionen gestartet werden.
- Wenn der Failover-Fall während des Warmstarts der Anwendung oder der Beendigung eines UTM-Prozesses eintritt, wird die Fehlerbehandlung wie gewohnt durchgeführt, es wird dann kein „Reconnect“ versucht.
- Die Datenbank-Funktionalität „prepared Statements“ kann im Failover-Fall zu Fehlern führen.
- Der „Reconnect“ zum Datenbank-System kann durch Meldungen verfolgt werden.
  - xa_close im Reconnect-Fall:

    In der Meldung K202 wird im Insert &RMSTAT anstelle von „closed“ für die Instanz von Oracle® Real Application Clusters der String „RAC closed“ ausgegeben.
  - xa_open im Reconnect-Fall:

    In der Meldung K224 wird im Insert &XACALL der String „RAC: xa_open“ ausgegeben.

  Debug-Meldungen
  In den Debug-Meldungen wird vermerkt, ob sich die Meldung auf eine Instanz von Oracle® Real Application Clusters bezieht. Wie Sie XA-DEBUG-Informationen für den Anschluss an die Datenbank bekommen, ist in Abschnitt „Debug-Parameter“ beschrieben.

Unterbrochene Transaktionen

Unterbrochene Transaktionen können nur von dem Knoten fortgesetzt werden, der die Transaktion gestartet hat. Daher müssen alle UTM-Prozesse stets mit dem gleichen Knoten des Oracle® Real Application Clusters verbunden sein. Deshalb ist es am einfachsten,

- die UTM-Anwendung nach dem Failover des Oracle® Real Application Clusters zu beenden, bevor der ausgefallene Knoten neu gestartet wird,
- nach dem Neustart des ausgefallenen Knotens die UTM-Anwendung neu zu starten.

Dadurch ist sichergestellt, dass alle UTM-Prozesse mit dem gleichen Knoten des Oracle® Real Application Clusters verbunden sind und alle Transaktionen der Anwendung vom neu gestarteten Knoten des Oracle® Real Application Clusters bearbeitet werden.

Falls ein Beenden und Neustart der UTM-Anwendung nicht möglich ist, d.h. falls Knoten des Oracle® Real Application Clusters bei laufender openUTM-Anwendung umgeschaltet werden, dann kann sich folgende Situation ergeben, in der nicht mehr alle UTM-Prozesse mit demselben Knoten verbunden sind:

- Eine Transaktion wird durch das Failover unterbrochen, zu diesem Zeitpunkt ist der UTM-Prozess noch mit dem alten Knoten verbunden.
- Durch Nachstarten des Prozesses oder nach einem PEND ER im UTM-Anwendungsprogramm wird die unterbrochene Transaktion von einem anderen UTM-Prozess weitergeführt. Dieser Prozess ist jetzt mit dem neuen Knoten verbunden.
- Die Datenbank-Instanz lehnt die Wiederaufnahme der unterbrochenen Transaktion ab (xa-start mit RESUME) und meldet, dass die Transaktion nicht bekannt ist.
- openUTM führt einen „Reconnect“ zur Datenbank-Instanz aus. Auf der neuen Verbindung (d.h. mit dem neuen Knoten) versucht openUTM, die Transaktion wieder aufzunehmen.
- Das Datenbank-System lehnt dies erneut ab, da die Datenbank-Transaktion auf dem alten Knoten des Oracle® Real Application Clusters begonnen wurde und auf dem neuen nicht fortgesetzt werden kann.
- openUTM setzt die globale Transaktion zurück und gibt eine K160-Meldung aus, im Insert des internen Returncodes KCRCDC wird „NOTA“ ausgegeben.

Eine solche Situation lässt sich wie folgt mit Hilfe eines MSGTAC-Programms behandeln.

Steuerung über ein MSGTAC-Programm

Für die K160-Meldung wird als Meldungsziel der Event-Service MSGTAC definiert. MSGTAC reagiert auf den Insert der Meldung und stößt über die Programmschnittstelle zur Administration (KC_CHANGE_APPLICATION) ein Neustarten an. Damit werden alle Prozesse ausgetauscht, neu gestartet und sind anschließend mit dem neuen Knoten verbunden.

Durch dieses Verfahren wird die Zeitspanne minimiert, in der die UTM-Prozesse mit unterschiedlichen Knoten verbunden sind. Die Zahl der zurückgesetzten Transaktionen beschränkt sich auf diejenigen, die auf dem alten Knoten begonnen wurden und auf dem neuen nicht fortgesetzt werden konnten. Die Transaktionen, die auf dem neuen Knoten vor dem Neustarten begonnen wurden, können weitergeführt werden.

6.4.2.1 Besonderheiten bei einem Oracle®-Anschluss

Die Verbindung zu einer Oracle®-Datenbank wird über einen sogenannten „Service“ aufgebaut. In einer Oracle® Real Application Clusters-Umgebung können Sie zusätzlich "DTP services" einrichten.

Daraus ergeben sich für den laufenden Betrieb folgende Möglichkeiten:

- automatische Fehlererkennung
- automatisches Failover.
  Nach Ausfall einer Instanz wird eine neue Transaktion auf eine andere Instanz des „service“ umgeleitet. Es muss nicht administrativ eingegriffen werden.
- Lastverteilung bereits beim Verbindungsaufbau

DTP service anlegen (Oracle ®)

1. Richten Sie mit dem Kommando “srvctl add service“ einen neuen "service" für die Datenbank ein und ordnen Sie ihn einer Instanz der Datenbank zu.

   Beispiel:

   Für die RAC-Datenbank `dbracutm` mit den Instanzen `racutm1` und `racutm2` sollen zwei "DTP services" mit folgenden Optionen angelegt werden:

-d	Name der Datenbank
-s	Name des (DTP-)Services
-r	Name der Erst-Instanz
-a	Name der Zweit-Instanz
-P	Methode des Failovers

   ```
   "srvctl add service -d dbracutm -s racutmS12 -r racutm1
                                                -a racutm2
                                                -P BASIC"
   ```

   und

   ```
   "srvctl add service -d dbracutm -s racutmS21 -r racutm2
                                                -a racutm1
                                                -P BASIC"
   ```

   Der Service `racutmS12` verbindet sich mit der Instanz `racutm1` und im Failover-Fall mit Instanz `racutm2`. Umgekehrt verbindet sich Service `racutmS21` mit der Instanz `racutm2` und im Failover-Fall mit Instanz `racutm1`.
2. Wandeln Sie mit SQLPLUS die Services in "DTP services“ um:

```
SQL> connect ....
SQL> execute dbms_service.modify_service
             ( service_name => 'racutmS12', dtp => true );
SQL> execute dbms_service.modify_service
             ( service_name => 'racutmS21', dtp => true );
SQL> exit
```

Sie können die (DTP-)Services mit "srvctl-Kommandos" starten, stoppen und administrieren, siehe auch Oracle ®-Handbuch "Administration and Deployment Guide".

> ℹ Der DTP-Service muss auf dem Knoten gestartet werden, auf dem die ihm primär zugewiesene Instanz des RAC-DB-Systems läuft, d.h. der DTP-Service `racutmS21`, der primär der Instanz `racutm2` zugeordnet ist, muss auf dem Knoten gestartet werden, auf dem diese Instanz läuft.

3. Tragen Sie den "service" in der Datei tnsnames.ora mit einem net_service_name ein:
   Beispiel

```
RACUTMS1 =
  (DESCRIPTION =
     (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP) (HOST=server1) (PORT=1521))
        (ADDRESS = (PROTOCOL = TCP) (HOST=server2) (PORT=1521))
     )
     (CONNECT_DATA =
        (SERVICE_NAME = racutmS12.domain_name )
     )
     (FAIL_OVER = ON)
  )
```

4. Weisen Sie in den Startparametern im Open-String dem Operanden "SqlNet" diesen net_service_name (hier RACUTMS1) zu.

6.4.3 Debug-Parameter

Sie haben die Möglichkeit, zu Testzwecken die XA-Schnittstelle in openUTM zu protokollieren. Dazu steht Ihnen der RMXA-Startparameter DEBUG= zur Verfügung.

Der Parameter DEBUG= hat folgendes Format und sollte als erster RMXA-Startparameter verwendet werden.

```
.RMXA DEBUG={ YES | ALL },OUTPUT={ SYSOUT | FILE }
```

Erläuterung

DEBUG=

Einschalten der Debug-Funktion

YES

protokolliert werden die einzelnen XA-Aufrufe, sowie für jeden Aufruf

- die Vorgangsnummer
- der Transaktionszähler
- der Rückgabewert

ALL

protokolliert werden zusätzlich zu den Werten bei DEBUG=YES jeweils die Statuswerte und die XID.

OUTPUT=

bestimmt das Ziel der Ausgabe

SYSOUT

Ausgabe auf *stderr*

FILE

die Ausgabe erfolgt in eine Datei.
Die Datei hat eines der folgenden Formate:
`KDC.TRC.XA.`*`appliname.pid`* (stand-alone Anwendung)
`KDC.TRC.XA.`*`appliname.nodename.pid`* (UTM-Cluster-Anwendung)

appliname

Name der Anwendung

nodename

Name des Knotens, auf dem die Knoten-Anwendung läuft.

pid

PID des Prozesses

Die Protokollierung der XA-Schnittstelle können Sie während des Anwendungslaufs auch per Administration ein- oder ausschalten. Verwenden Sie dazu die Programmschnittstelle, die Administrationstools WinAdmin/WebAdmin oder das Administrationskommando KDCDIAG XA-DEBUG=. Details finden Sie im openUTM-Handbuch „Anwendungen administrieren“.

6.4.4 UTM-Datenbank-Anwendung normal beenden

Eine UTM-Datenbank-Anwendung beenden Sie mit Hilfe der UTM-Administration, siehe Abschnitt „UTM-Anwendung per Administration normal beenden“. openUTM schließt den Resource Manager, während sich der UTM-Workprozess der Anwendung beendet.

Wenn eine Anwendung auf Windows-Systemen als Dienst gestartet wurde, dann kann sie als Dienst beendet werden, siehe Abschnitt „Dienst beenden auf Windows-Systemen“, oder mit dem Dienstprogramm KDCSHUT, siehe Abschnitt „Das Tool KDCSHUT- UTM-Anwendung auf Shell-Ebene normal beenden“.

6.4.5 UTM-Datenbank-Anwendung abnormal beenden

Eine UTM-Datenbank-Anwendung kann per Administration oder aufgrund von Fehlern abnormal beendet werden, siehe Abschnitt „UTM-Anwendung abnormal beenden“. Nach einem abnormalen Anwendungsende kann es vorkommen, dass die Datenbank oder die KDCFILE in einem nicht-konsistentem Zustand sind.

In diesem Fall wird die Konsistenz der Daten beim anschließendem Warmstart der UTM-Datenbank-Anwendung überprüft und ggf. wieder hergestellt. Dabei führt openUTM eine gemeinsame Recovery Phase mit den beteiligten Datenbanksystemen durch.

6.5 Betrieb einer UTM-Datenbank-Anwendung

Der Betrieb einer UTM-Datenbank-Anwendung gehorcht den selben Prinzipien wie der Betrieb einer UTM-Anwendung. Die Besonderheiten, die dabei zu beachten sind, sind in den nachfolgenden Abschnitten beschrieben.

6.5.1 Anmelden und Abmelden eines Benutzers

Ein Benutzer, der mit einer UTM-Datenbank-Anwendung arbeiten möchte, meldet sich über die Client-spezifischen Anmeldeverfahren bei openUTM an. Entsprechendes gilt für das Abmelden.

Beim Anmelden stehen alle Möglichkeiten offen, die UTM für das Anmelden bietet. Insbesondere kann der Benutzer die SIGNON-Services von UTM nutzen. Dabei ist Folgendes zu beachten:

- Meldet sich der Benutzer über ein Terminal an, dann sind Datenbank-Aufrufe im ersten Teil des SIGNON-Services aus Sicherheitsgründen nicht erlaubt, es sei denn, man lässt dies per UTM-Generierung explizit zu mit der KDCDEF-Anweisung SIGNON, ...RESTRICTED=NO.
- Im zweiten Teil des SIGNON-Services kann das Berechtigungsprofil für den Benutzer aus der Datenbank gelesen werden. Damit lässt sich ein DB/DC-übergreifendes Berechtigungskonzept realisieren.

Näheres zum An- und Abmelden finden Sie im Kapitel „Arbeiten mit einer UTM-Anwendung“.

6.5.2 Diagnose

UTM bietet zur Fehlerdiagnose bei einer UTM-Datenbank-Anwendung die gleichen Informationsquellen wie bei einer reinen UTM-Anwendung, das sind UTM-Meldungen, Fehlercodes und Dumps. Einige dieser Quellen enthalten auch Datenbank-spezifische Daten. Diese sollten als erstes herangezogen werden, wenn der Fehler mit der Datenbank-Kopplung zusammenhängen könnte. Dabei handelt es sich um folgende UTM-Diagnose-Informationen:

- Die Datenbank-spezifischen UTM-Meldungen K068 und K071
- Die Start-Fehlercodes der Meldung K049
- Meldungen des XA-Datenbankanschlusses K201 bis K233
- Den inkompatiblen Returncode KCRCDC
- Die DB Diagarea des UTM-Dumps, falls ein UTM-Dump erzeugt wurde

Einzelheiten dazu finden Sie im openUTM-Handbuch „Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen“.

7 UTM-Cluster-Anwendung

Ein Cluster ist eine Anzahl von Rechnern (Knoten), die über ein schnelles Netzwerk verbunden sind und sich eine gemeinsame Peripherie teilen.

Auf einem Cluster kann eine UTM-Anwendung in Form einer UTM-Cluster-Anwendung ablaufen. Eine UTM-Cluster-Anwendung kann weitgehend wie eine einzelne UTM-Anwendung (stand-alone Anwendung) betrieben werden. Eine UTM-Cluster-Anwendung besteht aus mehreren identisch generierten UTM-Anwendungen, den Knoten-Anwendungen, die auf den einzelnen Knoten zum Ablauf kommen.

Auf Unix-, Linux- und Windows-Systemen kann eine UTM-Cluster-Anwendung auf bis zu 32 Knoten laufen.

7.1 Eigenschaften einer UTM-Cluster-Anwendung

Eine UTM-Cluster-Anwendung ist für den Ablauf auf mehr als einem Rechner vorgesehen und zeichnet sich durch folgende Eigenschaften aus:

- Die UTM-Cluster-Anwendung muss wegen einheitlicher Zugriffsrechte für die verwendeten Dateien auf allen Rechnern unter der gleichen Benutzerkennung laufen.
- Die Konfiguration der UTM-Cluster-Anwendung einschließlich der KDCFILE für alle Knoten wird in einem gemeinsamen Generierungslauf erstellt und ist daher für alle Knoten identisch. Dies gilt insbesondere auch für den Anwendungsnamen der UTM-Cluster-Anwendung.
- Die Rechner eines Clusters müssen einen kompatiblen Stand bezüglich Hardware- und Software-Konfiguration haben. Abweichungen bei Korrekturständen, Betriebssystemversionen und Updates sind möglich, Details siehe Freigabemitteilung.
- Die Knoten-Anwendungen einer UTM-Cluster-Anwendung müssen alle auf dem gleichen Betriebssystem (z.B. Solaris) mit dem gleichen Bit-Modus (einheitlich 32- oder 64-Bit in allen Knoten) laufen. Eine Mischkonfiguration, z.B. Unix- und BS2000-Rechner oder auch Solaris- und Linux-Rechner, ist nicht möglich.
- Für den Ablauf einer UTM-Cluster-Anwendung wird eine Anzahl von Dateien benötigt, auf die von allen Knoten gemeinsam zugegriffen werden kann, die UTM-Cluster-Dateien. Detaillierte Information zu den UTM-Cluster-Dateien entnehmen Sie dem Abschnitt „Ablaufumgebung“.
- Je Knoten gibt es Knoten-lokale Dateien. Diese müssen Sie mit einem knotenspezifischen Dateinamenspräfix anlegen. Die KDCFILE einer Knoten-Anwendung muss von allen Knoten-Anwendungen aus zugreifbar sein. Detaillierte Information zu den Knoten-lokalen Dateien entnehmen Sie dem Abschnitt „Ablaufumgebung“.

Besondere Eigenschaften eines Clusters auf Unix- und Linux-Systemen

- Für den Ablauf müssen die verwendeten Benutzerkennungen auf allen Rechnern nicht nur den gleichen Namen haben, sondern auch intern vom Betriebssystem mit der gleichen Benutzer-Nummer verwaltet werden, die beim Einrichten der Benutzerkennung vergeben wird.
- Für den Ablauf von Rechner-übergreifenden Skripten müssen die Knoten für die Ablaufkennungen gegenseitig ssh-Zugriffe erlauben.
- Auf Unix- und Linux-Systemen wird über **N**etwork **F**ile **S**ystem/**S**ervice (NFS4) auf die gemeinsamen Dateien zugegriffen. Als NFS4-Serversystem können Sie beispielsweise NetApp FAS verwenden.

Besondere Eigenschaften eines Clusters auf Windows-Systemen

- Die Windows-Rechner müssen Mitglieder einer gemeinsamen Windows-Domäne sein.
- Als Ablaufkennung muss auf allen Knoten eine identische Windows-Domänenkennung verwendet werden.
- Auf Windows werden Windows-Netzlaufwerke mit dem auf Windows üblichen CIFS-Protokoll verwendet.

7.2 Installation und Einsatzvorbereitung einer UTM-Cluster-Anwendung

7.2.1 Installation

Bevor Sie eine UTM-Cluster-Anwendung erzeugen und betreiben können, müssen Sie das Produkt openUTM auf allen Rechnern installieren, die für den Cluster verwendet werden sollen. Das Vorgehen bei der Installation von openUTM ist unabhängig davon, ob Sie später stand-alone oder UTM-Cluster-Anwendungen betreiben wollen, siehe auch Abschnitte „openUTM installieren auf Unix- und Linux-Systemen“ und „openUTM installieren auf Windows-Systemen".

Informationen zu den Software-Voraussetzungen für UTM-Cluster-Anwendungen finden Sie in der Freigabemitteilung.

Die Ablaufumgebung von openUTM (z.B. die Systemzeit) muss auf allen Knoten gleich sein, siehe Abschnitt „Eigenschaften einer UTM-Cluster-Anwendung“.

Für den Ablauf werden Dateien benötigt, auf die alle Knoten-Anwendungen gemeinsam zugreifen können, siehe Abschnitt „Ablaufumgebung“.

Im laufenden Betrieb einer UTM-Cluster-Anwendung ist der Einsatz von openUTM-Korrekturversionen möglich. Details dazu entnehmen Sie dem Abschnitt „Einsatz von openUTM-Korrekturstufen in der UTM-Cluster-Anwendung“ .

Wenn die Anwendungen auf Windows-Systemen als Dienst ausgeführt werden sollen, müssen Sie diesen Dienst auf allen Knoten entsprechend installieren und konfigurieren.

7.2.1.1 Installation der UTM-Laufzeitkomponenten für Unix- und Linux-Systeme

Da die Anwendungen über ihren Anwendungsnamen in der Datei `applifile` identifiziert werden, muss jeder Knoten eigene Installationsdateien verwenden, d.h. openUTM muss auf jedem Knoten installiert werden.

- Die UTM-Installation schlägt als Installationsverzeichnis */opt/lib/<utmversion>* vor, z.B. */opt/lib/utm70a00*.
- Wenn die Installation unter */opt/lib/<utmversion>* nicht möglich ist, empfehlen wir dringend, für die openUTM-Installation auf allen Knoten jeweils ein gleichnamiges Verzeichnis zu wählen, das auf einem Festplatten-Speicher liegt, der diesem Knoten exklusiv zugeordnet ist.
- Wenn auch diese Einheitlichkeit nicht möglich ist, beachten Sie folgende Punkte:

  1. Beim Binden der Anwendung mit dem Binder *ld*:

     - Legen Sie die Verzeichnisse mit den Shared Objects mit dem Flag *-L* fest, z.B. *-L/opt/lib/utm70a00/64/sys*.
     - Geben Sie die Namen der Shared Objects mit dem Flag *-l* ohne das Präfix *lib* und ohne Suffix an, z. B. *-lwork* für das Einbinden von *libwork.so*.
     - Halten Sie die Reihenfolge dieser Flags ein: Geben Sie das Flag *-L* vor *-l* an.

       Regel: „Groß-L vor Klein-l“, da Verwechslungsgefahr mit „i“ in der Großschreibung besteht.

  2. Setzen Sie beim Start der Anwendung die Umgebungsvariable $LD_LIBRARY_PATH und ggf. $LD_LIBRARY_PATH64 entsprechend den Verzeichnissen mit den verwendeten Shared Objects.

7.2.1.2 Installation weiterer Laufzeitkomponenten für Unix- und Linux-Systeme

> Installieren Sie auch weitere Laufzeitkomponenten, die von der UTM-Anwendung verwendet werden (z.B. Cobol-Laufzeitsystem oder Datenbank-Software), auf allen Knoten möglichst einheitlich, d.h. in den gleichen Verzeichnissen.

  Dadurch ist sichergestellt, dass der Zugriff auf diese Laufzeitkomponenten über die gleichen Pfade (z.B. auch bei der Knoten-Recovery) von jedem Knoten aus konsistent möglich ist.

> Falls keine einheitliche Installation möglich ist, halten Sie die Hinweise zum Binden und Starten entsprechend der Beschreibung in Abschnitt „Installation der UTM-Laufzeitkomponenten für Unix- und Linux-Systeme“ ein.

7.2.2 UTM-Generierung

Die Konfiguration der UTM-Cluster-Anwendung einschließlich der initialen KDCFILE wird in einem gemeinsamen Generierungslauf erstellt.

Die initiale KDCFILE für eine UTM-Cluster-Anwendung legen Sie im Basis-Generierungslauf an. Sie wird unter dem Basisnamen abgelegt, den Sie im Operanden KDCFILE der Anweisung MAX angeben.

7.2.2.1 Spezielle Generierungsanweisungen für UTM-Cluster-Anwendungen

Für die UTM-Generierung einer UTM-Cluster-Anwendung sind spezifische Generierungsanweisungen nötig:

- Die CLUSTER-Anweisung definiert gemeinsame Eigenschaften der UTM-Cluster-Anwendung.
- Die CLUSTER-NODE-Anweisungen definieren die Rechner, auf denen die Knoten-Anwendungen laufen, und legen zu jeder Knoten-Anwendung die knotenspezifischen Eigenschaften fest. Für jede Knoten-Anwendung müssen Sie eine eigene CLUSTER-NODE-Anweisung angeben.

> **i** Über die Anzahl der CLUSTER-NODE-Anweisungen legen Sie Anzahl der Knoten-Anwendungen für den Cluster fest. Im laufenden Betrieb können Sie später keine zusätzlichen Knoten-Anwendungen in den Cluster aufnehmen.
> Sie können aber bei der UTM-Generierung „Reserve“-Knoten anlegen, die Sie später per Administration modifizieren und mit tatsächlichen Werten für weitere Knoten füllen können, siehe unten.

openUTM-Handbuch „Anwendungen generieren“

CLUSTER-Anweisung

Mit dem Operanden CLUSTER-FILEBASE legen Sie das Cluster-globale Namenspräfix für die Cluster-globalen Dateien der UTM-Cluster-Anwendung fest.

CLUSTER-NODE-Anweisung

Mit dem Operanden FILEBASE legen Sie den Knoten-lokalen Basisnamen für die Knoten-Anwendung fest.

7.2.2.2 UTM-Generierung von Reserve-Knoten

Sie haben die Möglichkeit, bei der Generierung mit KDCDEF Reserve-Knoten mit vorläufigen Werten anzulegen. Deren Rechnernamen und den Basisnamen der KDCFILE der Knoten-Anwendung können Sie später per Administration ändern. Dabei darf diese Knoten-Anwendung nicht aktiv sein.

Diese Möglichkeit ist insbesondere in folgenden Fällen hilfreich:

- Sie generieren als Reserve mehr Knoten als Sie zunächst betreiben wollen, da beispielsweise noch nicht genügend Rechner zur Verfügung stehen.

  Zu einem späteren Zeitpunkt möchten Sie einen Knoten zum bestehenden Cluster hinzunehmen, weil die Anzahl der bisher existierenden Knoten nicht ausreicht. Mit den Daten des neuen Knotens, die Ihnen nun bekannt sind, können Sie per Administration die Konfiguration für einen Reserve-Knoten modifizieren.
- Die Hardware, auf der eine Knoten-Anwendung läuft, ist defekt oder soll durch eine leistungsfähigere Hardware ausgetauscht werden. Dazu gehen Sie so vor:
  - Beenden Sie die Knoten-Anwendung.
  - Transferieren Sie die UTM-Anwendungsdaten auf den neuen Rechner.
  - Ändern Sie per Administration in einer laufenden Knoten-Anwendung den Rechnernamen des beendeten Knotens ab, d.h. tragen Sie dort statt dem alten Rechnernamen den neuen Namen des Knotens ein.

  Nachdem Sie die Änderung vorgenommen haben, können Sie die Knoten-Anwendung auf dem neuen Rechner starten.

Detaillierte Information zur UTM-Generierung von Reserve-Knoten und zur Änderung ihrer vorläufigen Eigenschaften mittels Administration entnehmen Sie dem openUTM-Handbuch „Anwendungen generieren “ sowie dem openUTM-Handbuch „Anwendungen administrieren“.

7.2.3 Nutzung globaler Speicherbereiche

In UTM-Cluster-Anwendungen werden die globalen UTM-Speicherbereiche GSSB und ULS Cluster-global unterstützt. Die zugehörigen Anwenderdaten werden im Cluster-Pagepool gespeichert.

openUTM-Handbuch „Anwendungen generieren“, CLUSTER-Anweisung
Mit den Operanden PGPOOL und PGPOOLFS legen Sie die Eigenschaften des Cluster-Pagepools fest (Größe, Warnstufe und Anzahl der Dateien). Mit dem Operanden DEADLOCK-PREVENTION steuern Sie das Verhalten bezüglich gesperrter globaler Speicherbereiche (zusätzliche Prüfung oder Steuerung über Timeout).

TACs für Zugriffe auf GSSB und ULS

In UTM-Cluster-Anwendungen sollten Sie den TACs zu Programmen, die auf die Speicherbereiche GSSB oder ULS zugreifen, TAC-Klassen zuweisen. Durch Beschränkung der Prozesse dieser TAC-Klassen kann verhindert werden, dass gleichzeitig alle Prozesse einer Knoten-Anwendung auf die Speicherbereiche GSSB oder ULS zugreifen. openUTM weist Zugriffe auf Speicherbereiche ab, wenn alle Prozesse einer Knoten-Anwendung warten müssten.

Es wird empfohlen, die TACs, die auf GSSB oder ULS zugreifen, möglichst in die gleiche TAC-Klasse zu legen. Falls TACs PGWT nutzen, sollten diese in derselben TAC-Klasse sein, da die PGWT-Wartesituationen auch berücksichtigt werden müssen.

Nachdem Sie die TACs einzelnen TAC-Klassen zugeordnet haben, können Sie die Anzahl der Prozesse entweder über die Anweisung TACCLASS oder die Anweisung TAC-PRIORITIES beschränken:

- Anweisung TACCLASS:
  Die Anzahl der Prozesse, die gestartet werden, muss um mindestens eins höher sein als die maximale Anzahl der Prozesse, die für die TAC-Klassen, in denen die TACs mit GSSB-/ULS-Zugriff sind, laufen dürfen.
- Anweisung TAC-PRIORITIES:
  Die Anzahl der Prozesse, die gestartet werden, muss um mindestens eins höher sein, als die Summe aus FREE-DIAL-TASKS und MAX ASYNTASKS.

Beispiele

Im folgenden Beispiel wird in der MAX-Anweisung TASKS=10 und ASYNTASKS=2 generiert. Die TACs mit GSSB-/ULS-Zugriff sollen in der TAC-Klasse 2 laufen (TAC.... TACCLASS=2). Damit gilt:

- Wenn die Prozess-Beschränkung über die TACCLASS-Anweisung gesteuert wird und die TAC-Klasse 2 maximal 5 Prozesse verwenden darf, lautet die TACCLASS-Anweisung:

  ```
  TACCLASS 2,TASKS=5,PGWT=YES
  ```

  Es müssen mindestens 6 Prozesse gestartet werden.
- Wenn die Prozess-Beschränkung über die TAC-PRIORITIES-Anweisung gesteuert wird und mindestens ein Prozess für Aufträge freigehalten werden soll, deren TACs keiner Dialog-TAC-Klasse angehören, dann lautet die TAC-PRIORITIES-Anweisung:

  ```
  TAC-PRIORITIES FREE-DIAL-TASKS=1
  ```

  Es müssen mindestens 4 Prozesse gestartet werden (wegen MAX ... ASYNTASKS=2).

7.2.4 Vorgangswiederanlauf

In UTM-Cluster-Anwendungen wird der Vorgangswiederanlauf für alle echten Benutzerkennungen, die mit RESTART=YES generiert sind, Cluster-global unterstützt.
D.h. ein Benutzer kann - nach dem Abmelden von der Knoten-Anwendung - einen offenen Dialog-Vorgang an einer anderen Knoten-Anwendung fortsetzen, sofern es sich nicht um einen knotengebundenen Vorgang handelt.

Knotengebundene Vorgänge

Folgende Vorgänge sind knotengebunden:

- Vorgänge, die eine Kommunikation mit einem Auftragnehmer über LU6.1 oder OSI TP begonnen haben und der Auftragnehmervorgang noch nicht beendet wurde
- eingeschobene Vorgänge einer Vorgangskellerung

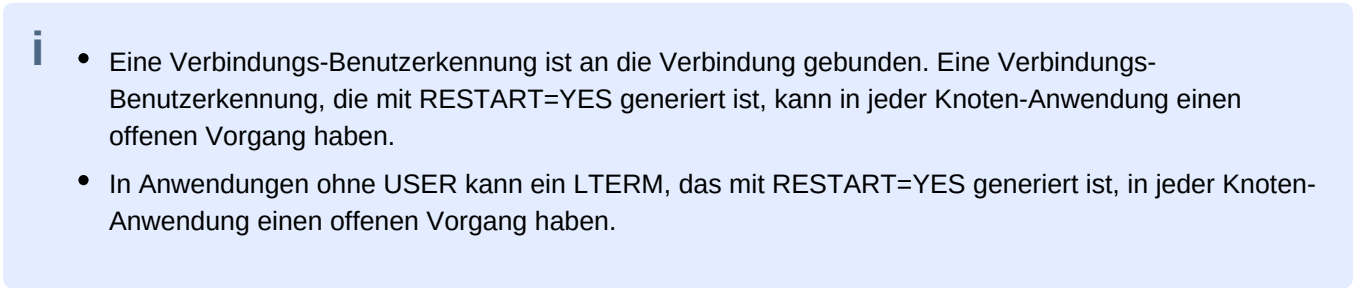
Außerdem ist der Vorgang eines Benutzers knotengebunden, solange der Benutzer an eine Knoten-Anwendung angemeldet ist. Daher ist ein offener Vorgang nach einem abnormalen Anwendungsende an eine Knoten-Anwendung gebunden, wenn der Benutzer zum Zeitpunkt des Anwendungsendes an die Knoten-Anwendung angemeldet war.

Knotengebundene Vorgänge können nur an dem Knoten fortgesetzt werden, an den sie gebunden sind.

Will sich ein Benutzer, der einen knotengebundenen Vorgang hat, an eine andere Knoten-Anwendung anmelden, so wird die Anmeldung abgelehnt, wenn

- die Knoten-Anwendung, an die der Vorgang gebunden ist, läuft, oder
- der gebundene Vorgang eine Transaktion im Zustand PTC hat oder
- die UTM-Cluster-Anwendung mit ABORT-BOUND-SERVICE = NO generiert ist.

Wird die Anmeldung eines Benutzers mit knoten-gebundenem Vorgang an einer anderen Knoten-Anwendung akzeptiert, so wird der offene Vorgang nicht fortgesetzt, sondern beim Start der Knoten-Anwendung, an die er gebunden ist, abnormal beendet.



- Eine Verbindungs-Benutzerkennung ist an die Verbindung gebunden. Eine Verbindungs-Benutzerkennung, die mit RESTART=YES generiert ist, kann in jeder Knoten-Anwendung einen offenen Vorgang haben.
- In Anwendungen ohne USER kann ein LTERM, das mit RESTART=YES generiert ist, in jeder Knoten-Anwendung einen offenen Vorgang haben.

Vorgangswiederanlauf bei UTM-F-Anwendungen

Der Vorgangswiederlauf wird auch in UTM-F-Anwendungen unterstützt, die Vorgangsdaten werden aber nur beim Abmelden eines Benutzers gesichert.

Daher ist nach einem abnormalen Ende einer Knoten-Anwendung kein Vorgangswiederanlauf mehr möglich, wenn der Benutzer

- zur Zeit des abnormalen Endes an der Knoten-Anwendung angemeldet war,
- oder einen an die abnormal beendete Knoten-Anwendung gebundenen Vorgang hat.

7.2.5 Ablaufumgebung

- Dateien
- Ablegen der Dateien

7.2.5.1 Dateien

Zur Ablaufumgebung einer UTM-Cluster-Anwendung gehören Cluster-globale und Knoten-lokale Dateien.

Den Ablageort der Dateien legen Sie bei der UTM-Generierung mit folgenden KDCDEF-Anweisungen fest:

- CLUSTER CLUSTER-FILEBASE = *cluster_filebase*
  cluster_filebase beschreibt den Ablageort der UTM-Cluster-Dateien.
- CLUSTER-NODE FILEBASE = *node_filebase*
  node_filebase beschreibt den Ablageort der Knoten-lokalen Dateien.

cluster_filebase müssen Sie beim Start der Knoten-Anwendungen über den Startparameter CLUSTER-FILEBASE = *cluster_filebase* für den Anwendungslauf angeben. Alle Knoten-Anwendungen müssen für diesen Startparameter den gleichen Wert angeben.

> ¡ Der Wert, den Sie bei der UTM-Generierung für *cluster_filebase* angegeben haben, muss nicht mit dem Wert übereinstimmen, den Sie über den Startparameter für *cluster_filebase* angegeben haben.
>
> Entscheidend ist, dass zum Zeitpunkt des Starts der ersten Knoten-Anwendung die UTM-Cluster-Dateien, wie z.B. die Cluster-Konfigurationsdatei, unter dem in den Startparametern angegebenen Basisnamen (*cluster_filebase)* zur Verfügung stehen.

UTM-Cluster-Dateien

Für den Ablauf einer UTM-Cluster-Anwendung wird eine Anzahl von Dateien benötigt, auf die von allen Knoten-Anwendungen gemeinsam zugegriffen werden kann. Diese UTM-Cluster-Dateien werden mit einem für die UTM-Cluster-Anwendung spezifischen Basisverzeichnis, der Cluster-Filebase, angelegt (*cluster_filebase*).

Die folgende Liste führt alle UTM-Cluster-Dateien auf. In dieser Liste wird der Dateiname ohne Basisverzeichnis angegeben, der komplette Name lautet jeweils:

cluster_filebase `/UTM-C.`xxxx auf Unix- und Linux-Systemen

cluster_filebase `\UTM-C.`xxxx auf Windows-Systemen

xxxx=CFG, USER, ..., LOCK

`UTM-C.CFG *)`	Cluster-Konfigurationsdatei. Sie enthält die Konfiguration des Clusters, den aktuellen Status aller Knoten des Clusters, weitere Informationen zu allen Knoten-Anwendungen einer UTM-Cluster-Anwendung sowie Angaben zu Cluster-globalen Daten.
UTM-C.USER *)	Cluster-User-Datei. Enthält benutzerspezifische Information zur Verwaltung der Benutzer einer UTM-Cluster-Anwendung. ¡ In einer UTM-Cluster-Anwendung ohne explizit generierte Benutzerkennungen wird die Cluster-User-Datei nicht benötigt und daher nicht erzeugt.

`UTM-C.CP` *nn* *) (*nn* = 01, ..., 10)	Cluster-Pagepool-Dateien, deren Anzahl bei der UTM-Generierung festgelegt wird. Enthalten Anwenderdaten, die in einer UTM-Cluster-Anwendung Cluster-global verwaltet werden (GSSB, ULS und die Vorgangsdaten von Benutzern).
`UTM-C.CPMD` *)	Verwaltungsdatei für den Cluster-Pagepool
`UTM-C.GSSB` *)	Cluster-GSSB-Datei. Dient zur Verwaltung von GSSB in einer UTM-Cluster-Anwendung
`UTM-C.ULS` *)	Cluster-ULS-Datei. Dient zur Verwaltung von ULS in einer UTM-Cluster-Anwendung.
`UTM-C.JRN1` `UTM-C.JRN2`	Administrations-Journal mit der Protokollierung globaler Administrationsaktionen („Gedächtnis “ der Administrationsfunktionen, siehe Abschnitt „Administrations-Journal“). Mit Hilfe dieser Dateien stellt openUTM sicher, dass globale administrative Änderungen Cluster-global konsistent wirksam werden.
`UTM-C.JKAA`	Journal-Datei mit Kopie der KDCS Application Area (KAA), aus der administrative Änderungen übernommen werden, die nicht mehr im Adminsstrations-Journal enthalten sind (siehe Abschnitt„Administrations-Journal“).
`UTM-C.LOCK`	Cluster-Lock-Datei. Dient zur Verwaltung von Warteschlangen in einer UTM-Cluster-Anwendung
UTM-C.SLCK	Lock-Datei zur Serialisierung der Startphase der Knoten-Anwendungen

Die mit *) gekennzeichneten UTM-Cluster-Dateien werden beim Generieren von KDCDEF angelegt (siehe Abschnitt „UTM-Generierung“).

Die Journaldateien (.JRN1, .JRN2, .JKAA) und die Lock-Dateien werden beim ersten Start der ersten Knoten-Anwendung von openUTM eingerichtet.

> **ACHTUNG!**
>
> Keine dieser Dateien dürfen Sie umbenennen oder umkopieren, weder während des Betriebs einer UTM-Cluster-Anwendung noch nachdem die UTM-Cluster-Anwendung beendet wurde.

Knoten-lokale Dateien

Zur Ablaufumgebung einer UTM-Cluster-Anwendung gehören neben den Cluster-globalen die Knoten-lokalen Dateien. Für die Knoten-lokalen Dateien wird jedem Knoten ein im Cluster eindeutiges Dateinamens-Präfix zugeordnet (*node_filebase*). Zu den Knoten-lokalen Dateien gehören für jede Knoten-Anwendung:

- die KDCFILE-Dateien (inkl. Pagepool und Wiederanlaufbereiche) als Kopien der initialen KDCFILE-Dateien:

  node_filebase /KDCA

  node_filebase \KDCA
  node_filebase /PxxA

  node_filebase \PxxA

  bei entsprechender UTM-Generierung

  node_filebase /RxxA
  node_filebase \RxxA
  bei entsprechender UTM-Generierung

  Die initialen KDCFILE-Dateien werden mit KDCDEF erzeugt (siehe Abschnitt „UTM-Generierung“). Sie müssen diese Dateien für jede Knoten-Anwendung kopieren.

  Die KDCFILEs der Knoten-Anwendungen müssen Sie so anlegen, dass alle KDCFILEs der Knoten-Anwendungen von allen Knoten-Anwendungen aus zugreifbar sind.
- System-Protokolldatei (SYSLOG-Datei)

  node_filebase /SYSLOG
  node_filebase \SYSLOG

  Die System-Protokolldatei SYSLOG kann eine Einzeldatei oder eine Dateigenerationsgruppe (FGG) sein.
- Benutzer-Protokolldatei

  node_filebase /USLA
  node_filebase \USLA

  Die Benutzer-Protokolldatei USLOG muss eine Dateigenerationsgruppe (FGG) sein.
- Ablaufprotokolle
  - Diagnose-Dateien
  - weitere anwendungsspezifische Dateien

SYSLOG-Datei und Benutzer-Protokolldatei sowie weitere anwendungsspezifische Dateien müssen Sie für jede Knoten-Anwendung einrichten.

> ¡ Auf Unix-, Linux- und Windows-Systemen können Sie unterschiedliche Versionen des Anwendungsprogramms halten. Bei allen laufenden Knoten-Anwendungen mit einer KDCFILE desselben Generierungslaufs muss aber jeweils die gleiche Version des Anwendungsprogramms geladen werden.

7.2.5.2 Ablegen der Dateien

Auf Unix- und Linux-Systemen

- Die Cluster-Filebase muss auf einem von allen Knoten zugänglichen Dateisystem liegen, d.h. sie liegt typischerweise auf einem per NFS4 zugreifbaren Speicher-Subsystem.
- Als NFS4 Mount Points müssen auf allen Knoten aus Konsistenzgründen gleichnamige Verzeichnisse verwendet werden.
- Es wird empfohlen, für die Cluster-Filebase und alle Filebase-Verzeichnisse der Knoten einen gemeinsamen Mount Point zu verwenden.

Auf Windows-Systemen

- Die Cluster-Filebase muss auf einem von allen Knoten zugänglichen Netzlaufwerk liegen.
- Auf Windows-Systemen wird mit Netzlaufwerken gearbeitet, auf die mit dem CIFS-Protokoll zugegriffen wird.

  Die Windows-Netzlaufwerke können über folgende zwei Formate angesprochen werden:

  - Über den Netzlaufwerksbuchstaben z. B. C:\
  - Über den UNC-Namen, d.h. im Format *ServerName**FreigabeName*\

Von allen Knoten muss auf diese Laufwerke mit dem identischen Format zugegriffen werden können. Diesen Namen müssen Sie bei der UTM-Generierung entsprechend angeben.

7.2.6 Einsatzvorbereitung

KDCFILE verteilen

Jede Knoten-Anwendung benötigt zum Ablauf eine Kopie der initialen KDCFILE aus dem gemeinsamen Generierungslauf mit einem ihr exklusiv zugeordneten Basisnamen. Dazu müssen Sie nach dem Generierungslauf die initiale KDCFILE (inkl. Pagepool, Wiederanlaufbereiche) für jede Knoten-Anwendung in die zugehörige knotenspezifische Filebase kopieren.

Startparameterdatei erstellen

Beim Starten einer Knoten-Anwendung müssen Sie den Startparameter CLUSTER-FILEBASE statt FILEBASE angeben, siehe auch Abschnitt „Startparameter für openUTM“. Unter dem bei CLUSTER-FILEBASE angegebenen Basisnamen müssen die Cluster-globalen Dateien vorhanden sein.

> ¡ Wenn Sie in der Startparameterdatei für *cluster_filebase* einen anderen Namen angeben wollen als Sie in den KDCDEF-Anweisungen gesetzt haben, müssen Sie die von KDCDEF erzeugten UTM-Cluster-Dateien vor dem Start der ersten Knoten-Anwendung entsprechend umbenennen.

7.2.7 Beispiel für Unix- und Linux-Systeme

In diesem Beispiel wird für die Cluster-Filebase und für die Filebase-Verzeichnisse aller Knoten-Anwendungen ein gemeinsamer Mount Point verwendet:

Das Volume `/vol/vol1` eines NFS4-Systems (z.B. NetApp Filer), der unter dem Namen `MyFiler` im Netzwerk angesprochen werden kann, wird auf allen Knoten im lokalen Verzeichnis `/myVol1` gemountet.

Dazu müssen auf allen Knoten folgende Aktionen ausgeführt werden:

login root
`mkdir /myVol1` (nur vor erstem `mount`-Befehl erforderlich)
`mount -t nfs4 MyFiler:/vol/vol1 /myVol1` (für Linux)
`mount MyFiler:/vol/vol1 /myVol1` (für Solaris)

> Beachten Sie, dass nach jedem Neustart eines Rechners der `mount`-Befehl wiederholt werden muss. Es wird empfohlen, das Mounten vom Systemverwalter automatisieren zu lassen.

Dateiverzeichnisse

In diesem Beispiel

- werden zwei Knoten generiert: `UTMHOST1`, `UTMHOST2`
- benutzen zwei Knoten-Anwendungen der UTM-C-Anwendung den gemeinsamen Mount Point `/myVol1`
- dient das Verzeichnis `/myVol1/UTMCAPPL` als Cluster-Filebase
- werden die Verzeichnisse für die Knoten-Anwendungen (Knoten-Filebase) unterhalb der Cluster-Filebase angeordnet und mit dem jeweiligen Hostnamen gekennzeichnet. Dieser Verzeichnisaufbau dient der besseren Übersichtlichkeit und ist weitgehend selbsterklärend.

`/myVol1/`	Mount Point
`/myVol1/UTMCAPPL/`	Cluster-Filebase
`/myVol1/UTMCAPPL/UTMHOST1/`	Filebase für UTMHOST1
`/myVol1/UTMCAPPL/UTMHOST2/`	Filebase für UTMHOST2

Beide Knoten-Anwendungen müssen Zugriffsberechtigung auf die Cluster-Filebase und auf die knotenspezifischen Verzeichnisse haben.

KDCDEF-Anweisungen

Für die Generierung der UTM-Cluster-Anwendung in diesem Beispiel sind folgende Generierungsanweisungen erforderlich:

```
OPTION GEN=(KDCFILE,ROOTSRC,CLUSTER)
CLUSTER CLUSTER-FILEBASE=/myVol1/UTMCAPPL,
        LISTENER-PORT=1234,BCAMAPPL=NAMECLT,
        CHECK-ALIVE-TIMER-SEC=60,USER-FILEBASE=/myVol1/UTMCAPPL
CLUSTER-NODE FILEBASE=/myVol1/UTMCAPPL/UTMHOST1,HOSTNAME=UTMHOST1
CLUSTER-NODE FILEBASE=/myVol1/UTMCAPPL/UTMHOST2,HOSTNAME=UTMHOST2
...
```

Ablage der Dateien

- Cluster-globale Dateien

  Beim Generierungslauf werden die Cluster-Konfigurationsdatei (`UTM-C.CFG`) und einige zentrale Cluster-Dateien im Verzeichnis `/myVol1/UTMCAPPL` angelegt. Die übrigen zentralen Cluster-Dateien werden beim Start der ersten Knoten-Anwendung ebenfalls im Verzeichnis `/myVol1/UTMCAPPL` angelegt.

`/myVol1/`	NFS4 Mount Point
`/myVol1/UTMCAPPL/`	Cluster-Filebase
`/myVol1/UTMCAPPL/UTM-C.CFG`	Cluster-Konfigurationsdatei
`/myVol1/UTMCAPPL/UTM-C.USER`	Cluster-User-Datei
`/myVol1/UTMCAPPL/UTM-C.CPMD`	Verwaltungsdatei des Cluster-Pagepools
`/myVol1/UTMCAPPL/UTM-C.CP01`	Cluster-Pagepool-Datei
`/myVol1/UTMCAPPL/UTM-C.GSSB`	Cluster-GSSB-Datei
`/myVol1/UTMCAPPL/UTM-C.ULS`	Cluster-ULS-Datei
Diese Dateien werden beim Generieren von KDCDEF angelegt.	

`/myVol1/UTMCAPPL/UTM-C.JRN1`
`/myVol1/UTMCAPPL/UTM-C.JRN2`
`/myVol1/UTMCAPPL/UTM-C.JKAA`
	Administrations-Journal
	wird beim ersten Start einer Knoten-Anwendung angelegt.

`/myVol1/UTMCAPPL/UTM-C.LOCK`	Cluster-Lock-Datei
`/myVol1/UTMCAPPL/UTM-C.SLCK`	Datei zur Serialisierung der Starts einzelner Knoten-Anwendungen
Diese Dateien werden beim ersten Start einer Knoten-Anwendung angelegt.	

  - Initiale KDCFILE:
    Die initiale KDCFILE, die bei diesem Generierungslauf erstellt wurde, müssen Sie in alle Filebase-Verzeichnisse der Knoten-Anwendungen kopieren.

- Dateien im Filebase-Verzeichnis am Knoten `UTMHOST1`

`/myVol1/UTMCAPPL/UTMHOST1/KDCA`	KDCFILE für UTMHOST1 müssen Sie vor dem ersten Start kopieren.
`/myVol1/UTMCAPPL/UTMHOST1/utmwork`	Programm für `utmwork`-Prozess von UTMHOST1 müssen Sie vor dem ersten Start zur Verfügung stellen.
...	restliche Dateien im Filebase-Verzeichnis auf UTMHOST1

- Dateien im Filebase-Verzeichnis am Knoten `UTMHOST2`

`/myVol1/UTMCAPPL/UTMHOST2/KDCA`	KDCFILE für UTMHOST2 müssen Sie vor dem ersten Start kopieren.
`/myVol1/UTMCAPPL/UTMHOST2/utmwork`	Programm für `utmwork`-Prozess von UTMHOST2 müssen Sie vor dem ersten Start zur Verfügung stellen.
`...`	restliche Dateien im Filebase-Verzeichnis auf UTMHOST2

Startparameterdateien

In beiden Startparameterdateien für die Knoten-Anwendungen muss der Name der Cluster-Filebase angegeben werden:

```
...
.UTM START CLUSTER-FILEBASE=/myVol1/UTMCAPPL
...
```

> Beachten Sie, dass die für stand-alone Anwendungen notwendige Anweisung `.UTM START FILEBASE=<filebase>` in einer Startparameterdatei einer UTM-Cluster-Anwendung nicht enthalten sein darf, siehe auch Abschnitt „Startparameter für openUTM“.

7.3 Konfiguration einer UTM-Cluster-Anwendung mit Datenbank

Da alle Knoten-Anwendungen die identische Konfiguration haben, arbeiten alle Knoten-Anwendungen mit dem gleichen Datenbank-System zusammen.

Verwendung von Oracle® Real Application Clusters (Oracle® RAC)

Beim Einsatz von Oracle® RAC wird folgende Konfiguration empfohlen:
Jeder Knoten-Anwendung ist jeweils ein primärer RAC-Knoten zugeordnet. Außerdem verwendet jede Knoten-Anwendung jeweils die anderen RAC-Knoten als Rückfallstufe für Failover.

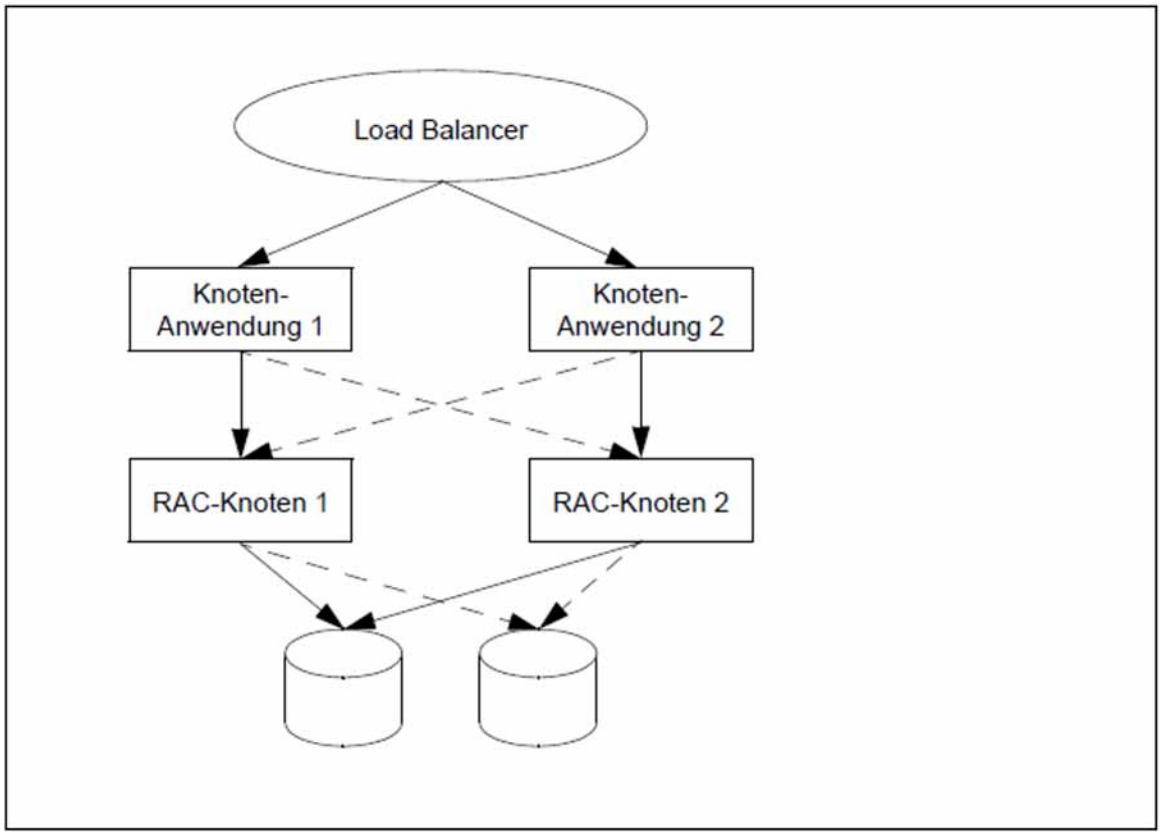


Bild 2: Konfiguration mit zwei Knoten-Anwendungen und zwei Oracle® RAC-Knoten

7.4 Starten einer UTM-Cluster-Anwendung

> ¡ Auf Unix- und Linux-Systemen: Setzen Sie vor dem Start der Anwendung die Umgebungsvariable LD_LIBRARY_PATH und ggf. LD_LIBRARY_PATH64 entsprechend den Verzeichnissen mit den verwendeten Shared Objects, wenn Sie keine einheitlichen Installationspfade verwenden konnten (siehe auch "Installation der UTM-Laufzeitkomponenten für Unix- und Linux-Systeme").

Eine UTM-Cluster-Anwendung starten Sie, indem Sie eine oder mehrere Knoten-Anwendungen starten. Sie starten jede Knoten-Anwendung einzeln wie eine stand-alone Anwendung (siehe Abschnitt „Starten einer UTM-Anwendung auf Unix- und Linux-Systemen“ und Abschnitt „Starten einer UTM-Anwendung auf Windows-Systemen“).

Startparameterdatei

Im Unterschied zu einer stand-alone UTM-Anwendung muss die Startparameterdatei statt der Anweisungen START FILEBASE=*filebase* die Anweisung START CLUSTER-FILEBASE=*cluster_filebase* enthalten.

Folgende Startparameter wirken Cluster-global:

- TESTMODE
- BTRACE
- OTRACE
- DUMP-MESSAGE
- Wert *interval* bei SYSPROT

Startparameter, die global auf alle Knoten wirken, werden von der ersten gestarteten Knoten-Anwendung über das Administrations-Journal an startende Folgeknoten verteilt. Sie gelten – auch während oder nach einer Änderungsgenerierung – so lange, bis die UTM-Cluster-Anwendung beendet wird oder der Wert administrativ geändert wird.

Falls die Knoten-Anwendungen keine spezifischen Startparameter erfordern, kann die Startparameterdatei für alle Knoten-Anwendungen gleich sein. Unter dem Basisnamen, den Sie bei CLUSTER-FILEBASE angegeben haben, müssen die von KDCDEF erzeugten UTM-Cluster-Dateien vorhanden sein. Diese Dateien müssen aus dem gleichen Generierungslauf stammen (siehe Abschnitt „Startparameterdatei erstellen“ (Einsatzvorbereitung)). Die Dateien der KDCFILE dürfen nicht älter sein als die UTM-Cluster-Dateien.

Beim Starten einer Knoten-Anwendung werden folgende Cluster-spezifischen Start-Aktionen ausgeführt:

- Es wird geprüft, ob die KDCFILE der Knoten-Anwendung und die Cluster-Konfigurationsdatei zusammenpassen.
- Beim ersten Start der ersten Knoten-Anwendung werden die Dateien des Administrations-Journals initialisiert und die Cluster-Lock-Datei sowie die Serialisierungs-Datei (UTM-C.SLCK) eingerichtet.
- Beim Start einer zweiten Knoten-Anwendung wird die Cluster-Überwachung gestartet, mit der sich die Knoten-Anwendungen überwachen.
- Beim Start einer weiteren Knoten-Anwendung wird die Cluster-Überwachung automatisch erweitert.
- Die Überwachungsbeziehungen werden dynamisch festgelegt (siehe Abschnitt „Anwendungsüberwachung der Knoten-Anwendungen“).

SYSLOG-Datei und Benutzer-Protokolldatei

Die System-Protokolldatei SYSLOG und die Benutzer-Protokolldatei müssen Sie für jede Knoten-Anwendung einrichten (siehe Abschnitte „System-Protokolldatei SYSLOG“ und „Benutzer-Protokolldatei“).

Die System-Protokolldatei SYSLOG muss entweder auf allen Knoten als einfache Datei oder auf allen Knoten als Dateigenerationsgruppe FGG (**F**ile **G**eneration **G**roup) eingerichtet sein (siehe Abschnitt „System-Protokolldatei SYSLOG“).

Alle laufenden Knoten-Anwendungen mit einer KDCFILE desselben Generierungslaufs müssen bezüglich der SYSLOG-Konfiguration gleich sein, ansonsten wird der Start eines Folgeknotens abgebrochen.

Verschlüsselungsfähigkeit

Es muss sichergestellt sein, dass auf allen Knoten die passende openSSL Bibliothek zur Verfügung gestellt wird.

7.5 Überwachung von Knoten-Anwendungen und Ausfallerkennung

Die Überwachung von Knoten-Anwendungen umfasst

- eine Anwendungsüberwachung
- sowie Maßnahmen bei Ausfallerkennung, wie beispielsweise das Starten eines Failure-Skripts.

7.5.1 Anwendungsüberwachung der Knoten-Anwendungen

Wenn für eine UTM-Cluster-Anwendung mehr als eine Knoten-Anwendung gestartet ist, dann wird jede Knoten-Anwendung durch eine andere Knoten-Anwendung überwacht.

Beim Start einer Knoten-Anwendung wird dynamisch bestimmt,

- welche andere Knoten-Anwendung von dieser Knoten-Anwendung überwacht werden soll,
- und welche andere Knoten-Anwendung diese Knoten-Anwendung überwachen soll.

Diese Überwachungsbeziehung wird in der Cluster-Konfigurationsdatei eingetragen. Beim Beenden der Knoten-Anwendung wird diese Beziehung wieder aufgelöst.

Überwachungsverfahren

Überwacht wird die Verfügbarkeit einer Knoten-Anwendung. Die Lebendüberwachung wird dabei mit Hilfe von Nachrichten durchgeführt, die über eine spezielle Verbindung ausgetauscht werden. Wenn es bei der Kommunikation zu Fehlern kommt, wird in einer zweiten Stufe geprüft, ob die KDCFILE des überwachten Knotens noch geöffnet ist.

Erst, wenn das Ergebnis aller dieser Prüfungen auf einen Ausfall hinweist, wird der Ausfall des überwachten Knotens angenommen.

Bei der Überwachung können Sie im Einzelnen (per UTM-Generierung) festlegen:

- Zeitintervall zwischen den Überwachungsnachrichten,
- Zeit, wie lange auf eine Antwort auf die Nachricht gewartet wird,
- Wiederholungsfaktor, d.h. wie oft bei Ausbleiben einer Antwort eine Nachricht wiederholt werden soll, bevor Stufe 2 der Überwachung zur Wirkung kommt.

openUTM-Handbuch „Anwendungen generieren“, CLUSTER-Anweisung
Die gegenseitige Überwachung der Knoten-Anwendungen konfigurieren Sie mit folgenden Operanden:

CHECK-ALIVE-TIMER-SEC=
COMMUNICATION-REPLY-TIMER-SEC=
COMMUNICATION-RETRY-NUMBER=

7.5.2 Aktionen der Knoten-Anwendungen bei Ausfallerkennung

Der Ausfall einer Knoten-Anwendung wird angenommen, wenn eine überwachte Anwendung innerhalb der konfigurierten Wartezeit und unter Berücksichtigung der konfigurierten Wiederholungsversuche nicht auf die Nachrichten antwortet, und wenn im Anschluss daran anhand der KDCFILE der überwachten Anwendung erkannt wird, dass diese Anwendung nicht mehr läuft, aber auch nicht normal beendet wurde.

Wird ein Ausfall oder eine abnormale Beendigung der überwachten Knoten-Anwendung erkannt, geht openUTM wie folgt vor:

- Die Knoten-Anwendung wird in der Cluster-Konfigurationsdatei als ausgefallen gekennzeichnet und aus den Überwachungsbeziehungen ausgetragen.
- Falls Sie in der UTM-Generierung ein sogenanntes Failure-Skript angegeben haben, startet die überwachende Knoten-Anwendung dieses Skript auf dem Rechner der überwachenden Knoten-Anwendung. Dem Failure-Skript werden folgende Daten der ausgefallenen Anwendung übergeben:
  - Anwendungsname
  - Basisname der Knoten-Anwendung
  - Rechnername
  - Virtual Host-Name oder Leerzeichen
  - Referenzname der Knoten-Anwendung
  - Fehlercode des UTM-Dumps (Term Application Reason)

  openUTM-Handbuch „Anwendungen generieren“, CLUSTER-Anweisung
  Um das Failure-Skript zu konfigurieren, geben Sie den Operanden FAILURE-CMD an. In diesem Operanden wird ein Kommando-String übergeben, der ein auszuführendes Kommando sowie Aufrufparameter enthält.

- Die überwachende Knoten-Anwendung startet einen Restart-Überwachungs-Timer, sofern Sie diesen konfiguriert haben:

  openUTM-Handbuch „Anwendungen generieren“, CLUSTER-Anweisung
  Um den Restart-Überwachungs-Timer zu konfigurieren, geben Sie den Operanden RESTART-TIMER-SEC an. Er legt die Zeit in Sekunden fest, die eine Knoten-Anwendung nach einem Ausfall maximal für einen Warm-Start benötigt.

- Falls Sie in der UTM-Generierung ein Emergency-Skript angegeben haben, startet die überwachende Knoten-Anwendung dieses Skript, wenn die ausgefallene Knoten-Anwendung nach Ablauf des Timers für die Restart-Überwachung nicht wieder verfügbar ist. Dem Emergency-Skript werden folgende Daten der ausgefallenen Anwendung übergeben:
  - Anwendungsname
  - Basisname der Knoten-Anwendung
  - Rechnername
  - Virtual Host-Name oder Leerzeichen
  - Referenzname der Knoten-Anwendung
  - Fehlercode des UTM-Dumps (Term Application Reason)

openUTM-Handbuch „Anwendungen generieren“, CLUSTER-Anweisung
Um das Emergency-Skript zu konfigurieren, geben Sie den Operanden EMERGENCY-CMD an. In diesem Operanden wird ein Kommando-String übergeben, der ein auszuführendes Kommando sowie Aufrufparameter enthält.

Beispiel-Skript bei Ausfallerkennung

Mit openUTM werden Beispiele für Failure- und Emergency-Skripte ausgeliefert. Die Beispiele geben beim Aufruf die übergebenen Parameter aus. Um die Beispiele in einer Produktionsumgebung einzusetzen, müssen Sie sie an die Anforderungen des betreffenden Clusters anpassen.

Unix- und Linux-Systeme

Folgende Beispiel-Skripte werden im Verzeichnis *utmpfad*`/shsc` ausgeliefert:

- `utm-c.emergency`
- utm-c.failure

Windows-Systeme

Folgende Beispiel-Skripte werden im Verzeichnis *utmpfad*`\shsc` ausgeliefert:

- `utm-c.emergency.cmd`
- `utm-c.failure.cmd`

7.5.3 Anwendungsdaten nach abnormaler Beendigung einer Knoten-Anwendung

In UTM-Cluster-Anwendungen gibt es Cluster-global gültige Anwendungsdaten und Knoten-lokale Anwendungsdaten:

- Cluster-global gültige Anwendungsdaten sind GSSB, ULS und Vorgangsdaten von nicht knoten-gebundenen Vorgängen. Diese Daten werden in den UTM-Cluster-Dateien gehalten.
- Die Knoten-lokalen Daten wie z.B. TLS und Vorgangsdaten von knotengebundenen Vorgängen (siehe "Vorgangswiederanlauf") werden in der KDCFILE der jeweiligen Knoten-Anwendung gesichert.

Wenn sich eine Knoten-Anwendung abnormal beendet, hat dies folgende Auswirkungen auf die Anwendungsdaten:

- Sperren auf die Cluster-globalen Speicherbereiche ULS und GSSB, die zum Zeitpunkt des Abbruchs der Knoten-Anwendung gehalten wurden, bleiben bestehen.
- Benutzer, die zu diesem Zeitpunkt an der Knoten-Anwendung exklusiv angemeldet waren, bleiben angemeldet.
- Auf Vorgangsdaten von Benutzern, die zum Zeitpunkt des Absturzes an der Knoten-Anwendung angemeldet waren, kann bis zum Warmstart nicht zugegriffen werden.
- Die von der abnormal beendeten Knoten-Anwendung reservierten Seiten des Cluster-Pagepools bleiben belegt.
- Weder ein Knoten- oder Cluster-Update noch ein Online-Import sind möglich.

Daher sollte für eine abnormal beendete Knoten-Anwendung zeitnah ein Warmstart durchgeführt werden.

7.5.4 Maßnahmen nach abnormaler Beendigung einer Knoten-Anwendung

Dieser Abschnitt beschreibt, was ein Benutzer nach einer abnormalen Beendigung einer Knoten-Anwendung tun muss und welche Maßnahmen der Administrator der UTM-Cluster-Anwendung in diesem Fall ergreifen kann.

7.5.4.1 Maßnahmen für Benutzer

Benutzer, die zum Zeitpunkt der abnormalen Beendigung an die Knoten-Anwendung angemeldet waren oder die einen offenen, an diese Knoten-Anwendung gebundenen Vorgang besitzen, können sich an eine andere Knoten-Anwendung anmelden. Dabei geht ein für einen solchen Benutzer offener Vorgang verloren. Ein offener Vorgang kann nur dann fortgesetzt werden, wenn das Neu-Anmelden **nach** dem Warmstart der ausgefallenen Knoten-Anwendung erfolgt.

Dabei kann sich ein solcher Benutzer jedoch erst an eine andere Knoten-Anwendung anmelden, nachdem die abnormale Beendigung der Knoten-Anwendung erkannt wurde:

- Die abnormale Beendigung der Knoten-Anwendung wurde bereits erkannt:

  Ein Benutzer mit RESTART=NO kann sich an einer anderen laufenden Knoten-Anwendung anmelden. Ein Benutzer mit RESTART=YES kann sich an eine andere laufende Knoten-Anwendung anmelden, wenn die Anwendung mit CLUSTER ABORT-BOUND-SERVICES=YES generiert ist, und der Benutzer keinen knotengebundenen Vorgang mit einer Transaktion im Zustand PTC hat.
- Die abnormale Beendigung der Knoten-Anwendung wurde noch nicht erkannt:

  Der Anmeldeversuch wird so lange abgelehnt bis der überwachende Knoten den Ausfall erkannt hat. Sobald der Ausfall erkannt wurde, wird wie im ersten Fall fortgefahren.

7.5.4.2 Maßnahmen für den Administrator

Abhängig davon ob die Knoten-Anwendung auf demselben Knoten wieder gestartet werden kann oder nicht, können folgende Maßnahmen notwendig sein, damit die Daten nicht „verloren“ sind:

- Falls die Knoten-Anwendung nach dem Ausfall auf demselben Knoten wieder gestartet werden kann, kann man ohne Probleme mit den bisherigen Daten weiterarbeiten. Über ein Failure-Skript kann z.B. ein automatischer Neustart der Knoten-Anwendung angestoßen werden.
- Falls ein Neustart auf demselben Knoten nicht möglich ist, weil z.B. der Rechner ausgefallen ist, gibt es folgende Alternativen:
    a. Verlagerung der Knoten-Anwendung auf einen Ersatzrechner mit identischem Hostnamen / IP-Adresse. Ein Neustart der Knoten-Anwendung auf diesem neuen Rechner ist ohne weitere Maßnahmen möglich.
    b. Verlagerung der Knoten-Anwendung auf einen Ersatzrechner mit identischem virtuellen Hostnamen / IP-Adresse. Bevor die Knoten-Anwendung auf diesem neuen Rechner gestartet werden kann, muss der Hostname des ausgefallenen Knotens in der Cluster-Konfigurationsdatei per Administration in den Hostnamen des Ersatzrechners geändert werden. Danach ist ein Neustart der Knoten-Anwendung auf diesem neuen Rechner möglich.
    c. Durchführen einer Knoten-Recovery, siehe Abschnitt „Knoten-Recovery“.

7.5.4.3 Knoten-Recovery

Ist ein Warmstart für eine abnormal beendete Knoten-Anwendung auf dem eigenen Knoten-Rechner nicht möglich und wurde auch kein virtueller Host definiert, dann kann für diese Knoten-Anwendung auf einem anderen Knoten des UTM-Clusters eine Knoten-Recovery durchgeführt werden, um die laufende UTM-Cluster-Anwendung nicht zu beeinträchtigen.

Voraussetzungen für die Nutzung der Knoten-Recovery

Die Knoten-Recovery benötigt Cluster-weit zugreifbare SYSLOG-Dateien mit knotenspezifischen Namen.

Sie können die Startparameterdatei mit dem gewünschtem Knoten-Namen für den Startparameter NODE-TO-RECOVER dynamisch erzeugen.

Alternativ dazu können Sie für jeden Knoten des Clusters eine vorgefertigte Cluster-weit zugreifbare Startparameterdatei für die Knoten-Recovery bereitstellen.

Wenn Sie UTM oder weitere Laufzeitkomponenten auf den einzelnen Cluster-Knoten entgegen den Empfehlungen unter verschiedenen Pfaden installiert haben, und dieser Code dynamisch aus Shared Objects geladen wird, beachten Sie Folgendes:

1. Für den Aufruf der Knoten-Recovery muss die Anwendung für diesen Einsatzfall passend gebunden sein.
2. Zusätzlich müssen Sie die Umgebungsvariablen $LD_LIBRARY_PATH sowie ggf. $LD_LIBRARY_PATH64 auf die lokal zugreifbaren Pfade setzen, d.h. wie beim Start der lokalen Knoten-Anwendung.

Detaillierte Information dazu entnehmen Sie den Abschnitten „Installation der UTM-Laufzeitkomponenten für Unix- und Linux-Systeme“ und „Installation weiterer Laufzeitkomponenten für Unix- und Linux-Systeme“.

Knoten-Recovery starten

Die Knoten-Recovery wird über die nachfolgend aufgelisteten Startparameter gesteuert.

NODE-TO-RECOVER

> wählt einen Knoten der UTM-Cluster-Anwendung aus, für den die Knoten-Recovery durchgeführt werden soll.

RESET-PTC

> legt fest, ob Transaktionen im Zustand PTC bei der Knoten-Recovery zurückgesetzt werden sollen.

Eine detaillierte Beschreibung dieser Startparameter entnehmen Sie dem Abschnitt „Startparameter für openUTM“.

utmmain für Knoten-Revocery aufrufen

> Starten Sie das Programm *utmmain* als Hintergrundprozess (siehe "Starten einer UTM-Anwendung auf Unix- und Linux- Systemen" für Unix- und Linux-Systeme bzw. "Starten einer UTM-Anwendung auf Windows-Systemen" für Windows-Systeme), um die Knoten-Recovery zu starten.

  Geben Sie dabei als erstes Argument den *filebase*-Namen der Knoten-Anwendung an, für die die Knoten-Recovery durchgeführt werden soll, und starten Sie die Knoten-Recovery in diesem *filebase*-Verzeichnis.

  > **ACHTUNG!**
  >
  > Die Startprozedur zum Starten der Knoten-Anwendung, die die Knoten-Recovery durchführen soll, darf keine Kommandos enthalten, die Auswirkungen auf eine parallel auf diesem Knoten-Rechner laufende Knoten-Anwendung haben. Dazu zählt z.B. ein Aufruf des Dienstprogramms *kdcrem* vor dem Start von *utmmain*.

Meldungen

Beim Start der Knoten-Recovery wird die Meldung K192 auf *stdout* und *stderr* ausgegeben. Darin sind die Werte der Startparameter NODE-TO-RECOVER und RESET-PTC zusammen mit dem aktuellen Rechnernamen protokolliert.

Für jede gefundene Transaktion im Zustand PTC wird eine Meldung K193 ausgegeben, unabhängig vom Wert des Parameters RESET-PTC.

Für jede zurückgesetzte Transaktion wird eine Meldung K160 ausgegeben.

Am Ende der Knoten-Recovery wird eine Meldung K194 ausgegeben, die jeweils die Anzahl der von diesem Knoten noch gesperrten GSSB und ULS anzeigt.

7.6 Online-Import von Anwendungsdaten

Nachdem eine Knoten-Anwendung normal beendet wurde, kann eine andere, laufende Knoten-Anwendung Nachrichten an (OSI-)LPAPs, LTERMs, Asynchron-TACs oder TAC-Queues und offene Asynchron-Vorgänge aus der beendeten Knoten-Anwendung importieren. Voraussetzung ist, dass ihre KDCFILE aus demselben Generierungslauf stammt. Importierte Daten werden in der beendeten Knoten-Anwendung gelöscht.

Ein Online-Import ist nur in UTM-S-Anwendungen (UTM-Secure) möglich und muss administrativ angestoßen werden, z.B. über WinAdmin oder WebAdmin.

Importierte Nachrichten werden wie neu erzeugte Nachrichten behandelt, d.h. sie werden an das Ende der Queue gehängt und nicht anhand ihres Erzeugungszeitpunkts in eine bestehende Nachrichten-Queue einsortiert.

Folgende Daten werden nicht importiert:

- Asynchron-Nachrichten an einen TAC, dessen Queue Level (QLEV) erreicht ist. Dies gilt auch dann, wenn der TAC mit QMODE = WRAP-AROUND generiert ist. Damit wird sichergestellt, dass durch den Import keine Asynchron-Nachrichten in der importierenden Anwendung gelöscht werden.

7.7 Administration einer UTM-Cluster-Anwendung

Sie können die Knoten-Anwendungen einer UTM-Cluster-Anwendung gemeinsam administrieren:

- WinAdmin/WebAdmin

  Mit **WinAdmin** und **WebAdmin** stehen Ihnen Administrationsfunktionen zur Verfügung, die Sie auf alle Knoten-Anwendungen in einer UTM-Cluster-Anwendung global anwenden können. Außerdem bieten WinAdmin /WebAdmin z.B. auch zusammenfassende Statistikanzeigen, die alle laufenden Knoten-Anwendungen einbeziehen.
  Mit WinAdmin/WebAdmin ist es aber auch möglich, gezielt einzelne Knoten-Anwendungen zu administrieren.

  Aus diesem Grund wird empfohlen, UTM-Cluster-Anwendungen mit WinAdmin oder WebAdmin zu administrieren.

  Detaillierte Informationen zur Administration von UTM-Cluster-Anwendungen mit WinAdmin oder WebAdmin entnehmen Sie der jeweiligen Online-Hilfe von WinAdmin/WebAdmin und dem Dokument „WinAdmin Beschreibung“ bzw. „WebAdmin Beschreibung“.

- Mit eigenen Administrationsprogrammen oder Administrationskommando

  Neben WinAdmin/WebAdmin besteht auch die Möglichkeit, eine UTM-Cluster-Anwendung per programmierter Administration oder per Administrationskommando zu administrieren. Je nach Art der Änderung wirkt sich der Administrationsauftrag entweder global auf alle Knoten-Anwendungen der UTM-Cluster-Anwendung oder nur auf eine einzelne Knoten-Anwendung aus.

  Detaillierte Informationen zur Programmschnittstelle und den Administrationskommandos entnehmen Sie dem openUTM-Handbuch „Anwendungen administrieren“.

Beachten Sie bei der Administration von UTM-Cluster-Anwendungen:

- Dynamisch erzeugbare Objekte müssen grundsätzlich per Administration gelöscht werden. Diese Objekte können nicht durch eine Neugenerierung allein gelöscht werden.
- Dynamisch erzeugbare Objekte können in einer UTM-Cluster-Anwendung nicht sofort (immediate) gelöscht werden, sondern nur verzögert (delayed).
- Um Speicherplatz von verzögert gelöschten Objekten in der KDCFILE freizugeben, müssen Sie die KDCFILE neu generieren.
- In einer UTM-Cluster-Anwendung können Sie Reserve-Knoten mit vorläufigen Eigenschaften definieren, die Sie später auf einfache Weise durch Modifizieren z.B. mit Hilfe von WinAdmin oder WebAdmin in „echte“ Knoten umwandeln können.
- Sie können sich Verteilte Transaktionen im Zustand PTC anzeigen lassen und den lokalen Teil einer solchen Transaktion zurücksetzen. Hierdurch wird ggf. auch die Transaktion in einer lokal angeschlossenen Datenbank zurückgesetzt.

7.7.1 Cluster-globale und Knoten-lokale Aktionen

Bei der Administration einer UTM-Cluster-Anwendung müssen Sie zwischen global wirkenden und lokal wirkenden Aktionen unterscheiden:

Cluster-globale Aktionen

Cluster-globale Aktionen werden für jede Knoten-Anwendung wirksam. Dies ist unabhängig davon, ob eine Knoten-Anwendung gerade aktiv ist oder nicht. Alle Knoten-Anwendungen vollziehen diese Änderungen anhand des Administrations-Journals nach (siehe Abschnitt „Administrations-Journal“).

Globale administrative Änderungen sind z.B.:

- Ändern des Passworts für eine Benutzerkennung
- Austausch des Anwendungsprogramms bzw. von Anwendungsteilen im laufenden Betrieb
- Erzeugen von Objekten
- Löschen von Objekten aus der Konfiguration

Knoten-lokale Aktionen

Knoten-lokale Aktionen wirken nur auf die Knoten-Anwendung, in der diese Aktionen ausgeführt werden.

Knoten-lokale administrative Änderungen sind z.B.:

- Beenden einer einzelnen Knoten-Anwendung
- Administrativer Aufbau einer Verbindung

Welche Aktionen sich Cluster-global oder Knoten-lokal auswirken, finden Sie in der Beschreibung der Operationscodes bzw. der Datenstrukturen im openUTM-Handbuch „Anwendungen administrieren“.

7.7.2 Administrations-Journal

Das Administrations-Journal enthält die Protokollierung zurückliegender globaler Administrationsaktionen, also das „Gedächtnis“ der Administrationsaktionen. openUTM richtet das Administrations-Journal beim ersten Start der ersten Knoten-Anwendung unter dem Filebase-Namen der zugehörigen UTM-Cluster-Anwendung ein (siehe auch Abschnitt „UTM-Cluster-Dateien“).

Das Administrations-Journal liegt, wie alle Cluster-globalen Dateien, auf einem gemeinsam zugreifbaren Speichermedium (siehe Abschnitt „Ablaufumgebung“). Konkurrierende Dateizugriffe werden vom UTM-Systemcode über NFS4 Locks realisiert.

Anhand des Administrations-Journals vollziehen alle Knoten-Anwendungen die global wirkenden administrativen Änderungen nach:

- Laufende Anwendungen ziehen diese Aktionen zeitnah nach. Sie tun dies spätestens, bevor sie selbst globale Administrationsaktionen veranlassen. Abhängig von der Auslastung eines Knotens geschieht dies in der Regel innerhalb weniger Sekunden. Sie werden dazu von der Knoten-Anwendung benachrichtigt, die direkt administriert wurde.

  Die Benachrichtigung kann durch ein Netzproblem auch einmal verloren gehen. Daher wird das Administrations-Journal abhängig vom Operanden
  CHECK-ALIVE-TIMER-SEC der CLUSTER-Anweisung in regelmäßigen Abständen von den laufenden Knoten-Anwendungen geprüft.
- Knoten-Anwendungen, die später neu gestartet werden, arbeiten die Änderungen während der Start-Phase ein.

7.7.3 Anzahl der Knoten verringern

Sie können die Anzahl der Knoten im Cluster verringern, ohne die Generierung der UTM-Cluster-Anwendung zu ändern.

Dazu gehen Sie wie folgt vor:

1. Beenden Sie die Knoten-Anwendungen der Knoten, die Sie für längere Zeit aus dem Cluster nehmen möchten.
2. Führen Sie auf einer noch laufenden Knoten-Anwendung einen Online-Import für die beendeten Knoten-Anwendungen durch, siehe auch Abschnitt „Online-Import von Anwendungsdaten“.

7.8 Shutdown einer UTM-Cluster-Anwendung

Sie haben verschiedene Möglichkeiten, eine UTM-Cluster-Anwendung zu beenden:

- Shutdown einer Knoten-Anwendung, z.B. mit dem Kommando KDCSHUT GRACE.
- Shutdown aller laufenden Knoten-Anwendungen der UTM-Cluster-Anwendung, z.B. mit KDCSHUT GRACE, SCOPE=GLOBAL.

  openUTM-Handbuch „Anwendungen administrieren“, Administrationskommando KDCSHUT

- Mit WinAdmin/WebAdmin:
  Beenden einer einzelnen Knoten-Anwendung oder Beenden einer UTM-Cluster-Anwendung mit allen laufenden Knoten-Anwendungen.

  WinAdmin Online-Hilfe bzw. **WebAdmin Online-Hilfe,** Anwendung beenden

- Mit einem selbst erstellten Administrationsprogramm:
  Beenden einer einzelnen Knoten-Anwendung oder Beenden einer UTM-Cluster-Anwendung mit allen laufenden Knoten-Anwendungen.

Wenn nur eine Knoten-Anwendung läuft, ist der Shutdown dieser letzten Knoten-Anwendung in der Wirkung gleichwertig mit dem Shutdown der kompletten UTM-Cluster-Anwendung.

7.9 Einsatz von openUTM-Korrekturstufen in der UTM-Cluster-Anwendung

Sie können openUTM-Korrekturstufen grundsätzlich bei laufendem Betrieb einsetzen, ohne die UTM-Cluster-Anwendung zu beenden. Ein Teil der Knoten-Anwendungen kann weiter laufen, während für die übrigen Knoten-Anwendungen die Korrekturstufe eingespielt wird.

Hierzu muss eine Knoten-Anwendung nach der anderen heruntergefahren und anschließend mit der neuen Korrekturstufe wieder gestartet werden.

Vorgehensweise auf Unix- und Linux-Systemen

Auf Unix- und Linux-Systemen werden UTM-Korrekturstufen immer unter einem für die jeweilige Stufe spezifischen Dateiverzeichnis installiert, d.h. die Dateien von zwei Korrekturstufen sind getrennt abgelegt und können sich nicht gegenseitig überschreiben.

Deshalb müssen Sie vor dem Start mit einer neuen Korrekturstufe die Werte der Umgebungsvariablen $UTMPATH anpassen. Das gilt auch für weitere Umgebungsvariablen, die von $UTMPATH abhängen, z.B. $PATH oder $LD_LIBRARY_PATH.

Gehen Sie folgendermaßen vor:

1. Neue Korrekturstufe installieren.
2. Knoten-Anwendung beenden.
3. Umgebungsvariable $UTMPATH auf die neue Installation zeigen lassen, d.h. gegebenenfalls das Start-Skript anpassen.
4. Anwendungsprogramm `utmwork` neu erstellen, falls dies für den Einsatz der Korrekturstufe nötig ist.
5. Knoten-Anwendung neu starten.
6. Wiederholen Sie die Schritte 1 bis 5 für alle anderen Knoten-Anwendungen der UTM-Cluster-Anwendung entsprechend.
7. Alte Korrekturstufe, falls nicht mehr benötigt, deinstallieren.

Vorgehensweise auf Windows-Systemen

Auf Windows-Systemen können Sie eine Korrekturstufe unter dem identischen Pfad wie eine Vorgängerstufe installieren. In diesem Fall werden die Dateien der vorhandenen Alt-Version überschrieben. Die Installation ist dann nur bei heruntergefahrener UTM-Anwendung möglich, weil ein Teil der Dateien sonst wegen Dateisperren des Systems nicht überschrieben werden kann.

> **i** Beachten Sie außerdem, dass diese Alt-Installation nicht zusätzlich von anderen UTM-Anwendungen, z. B. zusätzlichen Testanwendungen weiter verwendet werden kann.

Installieren Sie UTM-Korrekturstufen unter einem für die jeweilige Stufe spezifischen Dateiverzeichnis, sind die Dateien von zwei Korrekturstufen getrennt abgelegt und können sich nicht gegenseitig überschreiben.

In diesem Fall müssen Sie vor dem Start mit einer neuen Korrekturstufe die Werte der Umgebungsvariablen % UTMPATH% anpassen. Das gilt auch für weitere Umgebungsvariablen, die von %UTMPATH% abhängen, z.B. % PATH%.

Gehen Sie folgendermaßen vor:

1. Knoten-Anwendung mit dem alten Korrekturstand beenden.
2. Neue Korrekturstufe installieren.
3. Ggf. Rechner neu starten.
   Dieser Schritt ist nur notwendig, wenn sich der Installationspfad geändert hat, weil die Installation die Variable %UTMPATH% in der Systemumgebung anpasst.
4. Anwendungsprogramm `utmwork.exe` neu erstellen, falls dies für den Einsatz der Korrekturstufe nötig ist.
5. Knoten-Anwendung neu starten.
6. Wiederholen Sie die Schritte 1 bis 5 für alle anderen Knoten-Anwendungen der UTM-Cluster-Anwendung entsprechend.
7. Alte Korrekturstufe deinstallieren, falls sie nicht mehr benötigt wird.

Hinweise für den Ablauf als Dienst:

Wenn Sie den Installationspfad geändert haben, gehen Sie folgendermaßen vor:

1. Deinstallieren Sie vor dem Systemstart die alte Version des Dienstes.
2. Starten Sie das System neu.
3. Installieren Sie die neue Version des Dienstes wieder im System.

7.10 Diagnose in einer UTM-Cluster-Anwendung

Jede Knoten-Anwendung schreibt ihren eigenen Satz von Protokoll- und Diagnosedateien. Daher werden für die Diagnose immer mindestens die Protokolldateien der Knoten-Anwendung benötigt, in der ein konkreter Fehler aufgetreten ist.

Meldungen zur Knoten-Überwachung

Die überwachende Knoten-Anwendung gibt bei Beginn der Überwachung die Meldung K169 aus.

Detaillierte Informationen zu den Meldungen entnehmen Sie dem openUTM-Handbuch „Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen“**.**

Diagnoseunterlagen

Für die Diagnose von Cluster-Problemen sind zusätzlich zu den üblichen Unterlagen die folgenden Dateien nötig:

- alle UTM-Cluster-Dateien
- bei Problemen mit der Cluster-globalen Administration das Administrations-Journal
- bei Problemen, die durch das Zusammenspiel der Knoten-Anwendungen verursacht wurden, die Protokolldateien von allen Knoten-Anwendungen
- die Startprozedur und die bei der UTM-Generierung als EMERGENCY-CMD und FAILURE-CMD angegebenen Prozeduren

8 Arbeiten mit einer UTM-Anwendung

Dieses Kapitel beschreibt, wie ein Benutzer über die unterschiedlichen Clients mit einer UTM-Anwendung kommunizieren kann. Für alle Clients verläuft die Kommunikation immer nach folgendem Prinzip:

1. Anmelden an die UTM-Anwendung
   Ein Benutzer kann sich nur über Clients anmelden, für die LTERM-Partner, LTERM-Pools oder OSI-LPAP-Partner in der UTM-Anwendung generiert sind, siehe openUTM-Handbuch „Anwendungen generieren“. Eine Anmeldung z.B. über Remote-Login-Mechanismen ist nicht möglich.
2. Services (=Vorgänge) der UTM-Anwendung aufrufen
   Für der Zugriffsschutz bietet openUTM ein eigenes Berechtigungskonzept, siehe "Berechtigungskonzept von openUTM".
3. Ggf. UTM-Benutzerkommandos eingeben.
4. Abmelden von der UTM-Anwendung

Im Detail unterscheiden sich diese Schritte je nach Art des Clients. Die folgenden Abschnitte beschreiben, welche Möglichkeiten es für die verschiedenen Clients gibt.

Für den Zugang werden UTM-Benutzerkennungen verwendet, sofern die Anwendung mit Benutzerkennungen generiert ist. Das Anmelden an eine UTM-Anwendung ohne Benutzerkennungen wird im Kapitel "Anmeldeverfahren ohne Benutzerkennungen" behandelt.

8.1 Anmeldeverfahren mit Benutzerkennungen

Ist eine Anwendung mit Benutzerkennungen generiert, dann führt openUTM für den Benutzer abhängig von der Art des Client ein Standard-Anmeldeverfahren durch. Für manche Arten von Clients besteht daneben die Möglichkeit, anstelle des Standard-Verfahrens selbst erstellte Anmeldeverfahren zu verwenden, siehe Abschnitt „Anmeldeverfahren mit Anmelde-Vorgängen“.

Ein Benutzer kann sich über folgende Client-Zugänge anmelden:

- Terminals (siehe unten)
- UPIC-Clients und TS-Anwendungen ("Anmeldeverfahren für UPIC-Clients und TS-Anwendungen")
- OSI TP-Partner ("Anmeldeverfahren für OSI TP-Partner")
- HTTP-Clients ("Anmeldeverfahren für HTTP-Clients")
- über das Internet mit Hilfe von Web Services (WS4UTM) ("Anmeldeverfahren im Internet über Web Services (WS4UTM)")
- über das Internet mit Hilfe von WebTransactions ("Anmeldeverfahren im Internet über WebTransactions")

Unter bestimmten Voraussetzungen können sich auch mehrere Benutzer unter einer Benutzerkennung anmelden, siehe Abschnitt „Mehrfach-Anmeldungen unter einer Benutzerkennung“.

8.1.1 Standard-Anmeldeverfahren für Terminals

Um über ein Terminal mit einer UTM-Anwendung zu arbeiten, startet der Benutzer einen Dialog-Terminalprozess, siehe "Starten des Dialog-Terminalprozesses durch den Benutzer". Dabei meldet sich der Benutzer gleichzeitig an die UTM-Anwendung an.

Erst dann kann der Benutzer Vorgänge starten und im Dialog bearbeiten, siehe Abschnitt „UTM-Services aufrufen“.

Mit der Umgebungsvariablen LANG kann der Benutzer das Verhalten der Anwendung beeinflussen:

Mit LANG gibt er an, in welcher Sprache die an ihn gerichteten UTM-Meldungen ausgegeben werden. Es werden standardmäßig die Einstellungen LANG=De... für die deutschen Meldungen und LANG=En... für die englischen Meldungen unterstützt. Die Meldungen werden aus NLS-Meldungskatalogen aufgebaut.

8.1.1.1 Starten des Dialog-Terminalprozesses durch den Benutzer

Standardmäßig muss der Benutzer den Dialog-Terminalprozess selbst starten, nachdem er sich beim System angemeldet hat.

- Auf Unix- und Linux-Systemen gibt er dazu folgendes Kommando ein:

  utmpfad `/ex/utmdtp [-S[`*username*`]] [-A`*applicationname*`] [-P`*ptermname*`] [-D]`

  Es gibt auch die Möglichkeit, den Dialog-Terminalprozess durch das Unix- oder Linux-System starten zu lassen, siehe "Starten des Dialog-Terminalprozesses durch das Unix- oder Linux-System".
- Auf Windows-Systemen gehen Sie folgendermaßen vor:
  - Setzen Sie ggf. zuerst den korrekten *utmpfad*:

    `SET UTMPATH=`*utmpfad*
  - Starten Sie ein Eingabeaufforderungs-Fenster und geben Sie folgendes Kommando ein:

    utmpfad `\ex\utmdtp [-S[`*username*`]] [-A`*applicationname*`] [-P`*ptermname*`]`

  Die Eingabeaufforderung darf erst geschlossen werden, wenn der Benutzer sich von der Anwendung abgemeldet hat.

Erläuterung

Die Angaben in eckigen Klammern stellen Schalter dar, die man angeben kann, aber nicht angeben muss. Die Schalter haben folgende Bedeutung:

-S[*username*]

Mit diesem Schalter steuert der Benutzer die Berechtigungsprüfung (Zugangskontrolle) im Dialog, die openUTM nach dem erfolgreichen Verbindungsaufbau zur UTM-Anwendung durchführt.

Dialog-Terminalprozess **mit** Schalter -S:
Startet der Benutzer den Dialog-Terminalprozess mit Schalter -S, so muss er eine UTM-Benutzerkennung *username* zur Berechtigungsprüfung an openUTM übergeben. Mit -S*username* kann der Benutzer die UTM-Benutzerkennung direkt beim Start des Dialog-Terminalprozesses angeben. Mit -S ohne *username* erfragt openUTM die UTM-Benutzerkennung nach dem Verbindungsaufbau im Dialog.

Gibt der Benutzer eine UTM-Benutzerkennung an, für die ein Passwort generiert ist, dann fragt openUTM die entsprechenden Daten im Dialog ab, siehe Beschreibung ab "Standard-Anmeldedialog".

Dialog-Terminalprozess **ohne** Schalter -S:
Startet der Benutzer den Dialog-Terminalprozess ohne Schalter -S, so übergibt der Dialog-Terminalprozess die Benutzerkennung des Systems zur Berechtigungsprüfung. Das zugehörige Passwort wird nicht an openUTM übergeben. In der UTM-Anwendung darf für die Benutzerkennung ein Passwort vergeben werden, zu dessen Eingabe - wie im Fall der expliziten Angabe der Benutzerkennung - der Benutzer aufgefordert wird.

Verläuft eine Prüfung mit der Benutzerkennung des Systems negativ, dann folgt ein expliziter Berechtigungsdialog wie bei der Verwendung des Schalters -S.

Wie der Berechtigungsdialog abläuft, ist in Abschnitt „Standard-Anmeldedialog“ beschrieben.

> ¡ Schalter -S und *username* bilden einen String und dürfen durch kein Zeichen getrennt werden.

-A*applicationname*

Mit diesem Schalter gibt der Benutzer die Anwendung an, mit der er verbunden werden will. *applicationname* ist der mit MAX APPLINAME generierte Name der Anwendung. Wird -A*applicationname* beim Start des Dialog-Terminalprozesses **nicht** angegeben, dann wird der Benutzer zur Eingabe des Anwendungsnamens aufgefordert.

> ¡ Schalter -A und *applicationname* bilden einen String und dürfen durch kein Zeichen getrennt werden.

-P*ptermname*

ptermname ist der Name des Terminals, über das der Benutzer die Verbindung zu openUTM aufbaut. Dieser Name muss in einer PTERM-Anweisung generiert sein oder es muss ein LTERM-Pool für lokale Terminals definiert sein (TPOOL PTYPE=TTY,LTERM=*ltermprefix*,NUMBER=*number*). Wenn ein solcher LTERM-Pool generiert ist, kann der Schalter -P immer weggelassen werden. Andernfalls gilt:

- Unix- und Linux-Systeme
  Falls auf Unix- und Linux-Systemen kein solcher LTERM-Pool generiert ist, dann kann der Schalter -P bis auf den unten beschriebenen Ausnahmefall weggelassen werden, denn standardmäßig verwendet openUTM auf Unix- und Linux-Systemen als *ptermname* den letzten Teil der Ausgabe des tty-Kommandos. Das ist der Term hinter dem letzten Schrägstrich, er entspricht der Ausgabe von `basename `tty``.
  Der Schalter -P ist nur dann notwendig, wenn die Standardzuordnung von *ptermname* durch openUTM nicht eindeutig ist, z.B. wenn mehrere Pseudoterminals existieren, die sich in dem letzten Term des tty (hinter dem letzten Schrägstrich) nicht unterscheiden.

  Beispiel
  Es existieren gleichzeitig die ttys `/dev/pts/12` und `/dev/inet/12`. Diese ttys sind generiert mit PTERM pts/12 und PTERM inet/12. In diesem Fall muss der Benutzer den Dialog-Terminalprozess mit Schalter -P starten, z.B. wie folgt:
  utmpfad `/ex/utmdtp` ... `-Ppts/12`.
- Windows-Systeme
  Falls auf Windows-Systemen kein solcher LTERM-Pool generiert ist, dann muss der Benutzer immer -P *ptermname* angeben, denn openUTM verwendet auf Windows-Systemen für *ptermname* standardmäßig den (nicht generierten) Wert `tty` *nnnnn* `,` *nnnnn* `=PID`.
  Beispiel:
  Wenn das lokale Terminal mit PTERM console,PTYPE=TTY,... generiert ist, dann muss er den Dialog-Terminalprozess starten mit
  utmpfad `\ex\utmdtp` ... `-Pconsole`.

> ¡ Schalter -P und *ptermname* bilden einen String und dürfen durch kein Zeichen getrennt werden.

-D Dieser Schalter gilt nur für Unix- und Linux-Systeme.

Der Benutzer bestimmt auf Unix- und Linux-Systemen mit diesem Schalter die Reaktion des Dialog-Terminalprozesses auf die Taste DEL.

Schalter -D angegeben: Die Taste DEL wird vom Dialog-Terminalprozess ignoriert.

Schalter -D **nicht** angegeben: Drücken der Taste DEL führt zum Abmelden von der Anwendung und zur Beendigung des Dialog-Terminalprozesses.

Beispiel:
Der Benutzer möchte sich unter der UTM-Benutzerkennung `user1` an die Anwendung `sample` anmelden; die Taste DEL soll ignoriert werden. Dazu muss er folgendes Kommando eingeben:

utmpfad/ex/utmdtp -Suser1 -Asample -D

8.1.1.2 Starten des Dialog-Terminalprozesses durch das Unix- oder Linux-System

Ein Unix- oder Linux-System kann den Dialog-Terminalprozess auch von sich aus starten, nachdem der Benutzer sich erfolgreich beim System angemeldet hat. Dazu muss der Benutzer das Kommando zum Starten des Dialog-Terminalprozesses z.B. wie folgt in seine eigene `.profile` eintragen:

- `.../utmdtp` *schalter* bewirkt, dass nach Beendigung von `utmdtp` die Shell noch aktiv bleibt.
- `exec .../utmdtp` *schalter* bewirkt, dass nach Beendigung von `utmdtp` die Shell ebenfalls beendet wird.

Eine zweite Möglichkeit besteht darin, dass der Systemverwalter in der Datei `/etc/passwd` für den Benutzer als Programmnamen den Dialog-Terminalprozess einträgt oder (bei Verwendung von Schaltern) eine Shell-Prozedur startet, in welcher der Aufruf des Dialog-Terminalprozesses steht.

Der Dialog-Terminalprozess ist sinnvollerweise erst dann zu starten, wenn die Workprozesse den erfolgreichen Kalt- oder Warmstart der Anwendung gemeldet haben. Sonst beendet sich der Dialog-Terminalprozess wieder mit der Meldung:

`U111 UTM-Anwendung` *applicationname* `ist nicht gestartet.`

Tritt ein Fehler auf beim Ablauf des Dialog-Terminalprozesses, beendet sich dieser mit der folgenden Meldung. Die Fehlernummern *nnnn* sind im openUTM-Handbuch „Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen“ beschrieben:

`U120 utmdtp-Prozeß beendet sich wegen Fehlernummer` *nnnn*`.`

8.1.1.3 Standard-Anmeldedialog

Der Standard-Anmeldedialog wird immer dann durchgeführt, wenn die beiden folgenden Bedingungen erfüllt sind:

1. Für das Terminal ist kein automatisches KDCSIGN (= automatische Berechtigungsprüfung) generiert (siehe "Automatisches KDCSIGN").
2. Für den Standard-Anwendungsnamen (generiert in MAX APPLINAME), unter dem sich der Benutzer angemeldet hat, ist kein Anmelde-Vorgang generiert (siehe "Anmeldeverfahren mit Anmelde-Vorgängen").

Beim Standard-Anmeldedialog führt openUTM eine Berechtigungsprüfung durch (Zugangskontrolle). Der Anmeldedialog ist nicht modifizierbar.

Die Berechtigungsprüfung kann unterschiedlich streng festgelegt werden. Eine Übersicht aller Möglichkeiten enthält die Abbildung in Abschnitt „Szenarien für die UTM-Berechtigungsprüfung“.

Die Berechtigungsprüfung führt openUTM im Dialog mit dem Benutzer durch, wenn beim Start des entsprechenden Dialog-Terminalprozesses der Schalter `-S` angegeben wird. In diesem Fall fragt openUTM die Benutzerkennung ab und fordert - falls generiert- die Angabe eines Passworts an.

Wird der Schalter `-S` nicht angegeben, führt openUTM die Berechtigungsprüfung mit der Benutzerkennung durch, unter der sich der Benutzer beim System angemeldet hat. In diesem Fall kann in der UTM-Anwendung für die Benutzerkennung ebenfalls ein Passwort generiert werden, zu dessen Eingabe der Benutzer bei der Berechtigungsprüfung aufgefordert wird.

> ¡ Zu beachten ist jedoch, dass ein Benutzer nicht unter einer Benutzerkennung gleichzeitig mit mehreren Dialog-Terminalprozessen arbeiten kann.

Eingabe des Passworts

Wenn für die Benutzerkennung ein **Passwort** generiert ist (KDCDEF-Anweisung USER...,PASS=*password*[, DARK]), dann wird das Passwort immer in ein Feld ohne Dunkelsteuerung eingegeben.

Bei jeder Anmeldung an die UTM-Anwendung hat der Benutzer die Möglichkeit, ein neues Passwort einzugeben, welches das bisherige ersetzen soll, vorausgesetzt, die minimale Gültigkeitsdauer erlaubt die Passwortänderung zu diesem Zeitpunkt. Das neue Passwort muss dann einmal in ein Feld ohne Dunkelsteuerung eingegeben werden. openUTM prüft das angegebene alte Passwort und ggf. das neue Passwort. Ist das alte Passwort nicht richtig, dann wird der Benutzer mit einer UTM-Meldung informiert und erneut zur Eingabe einer Benutzerkennung aufgefordert.

Gültigkeitsdauer des Passwortes

Bei der UTM-Generierung der Benutzerkennung kann eine maximale und eine minimale Gültigkeitsdauer für das Passwort vereinbart werden:

USER ...,PROTECT-PW=(...,*maxtime*,*mintime*)

Die minimale Gültigkeitsdauer bedeutet, dass der Benutzer nach einer Passwort-Änderung die nächste Änderung erst nach Ablauf der festgelegten Zeit vornehmen kann. Die maximale Gültigkeitsdauer bedeutet, dass der Benutzer das Passwort jeweils innerhalb der angegebenen Zeitspanne ändern muss.

Wird das Passwort innerhalb der nächsten 14 Tage nach der Anmeldung ungültig, warnt openUTM den Benutzer mit einer K-Meldung, sofern die minimale Gültigkeitsdauer des Passworts eine Änderung zu diesem Zeitpunkt erlaubt. Ein Passwort kann der Benutzer wie unter „Eingabe des Passworts“ beschrieben ändern.
Um zu verhindern, dass Benutzer, die längere Zeit nicht mit der Anwendung arbeiten, die Passwort-Änderung versäumen und sich dann an den Administrator wenden müssen, kann die UTM-Anwendung so konfiguriert werden, dass auch nach Ablauf des Passworts eine Anmeldung erlaubt wird, siehe nächste Seite, Abschnitt „Grace-Sign-On“.

openUTM überprüft bei der Passwortänderung, ob

- das neue Passwort sich von dem alten Passwort unterscheidet, wenn eine maximale Passwortgültigkeitsdauer generiert ist.
  Wenn eine Passwort-Historie generiert ist (SIGNON ...,PW-HISTORY=*n*), dann wird auch gegen die letzten *n* Passworte geprüft.
- das neue Passwort der Komplexitätsstufe entspricht, die für die Benutzerkennung generiert wurde (USER ..., PROTECT-PW=).
- die Länge des Passwortes größer oder gleich der generierten Mindestlänge ist (USER ...,PROTECT-PW=).

Erfüllt das neue Passwort alle diese Punkte, nimmt openUTM die Passwortänderung vor. Die Gültigkeitsdauer des neuen Passwortes entspricht wieder dem generierten Wert.
Erfüllt das neue Passwort einen dieser Punkte nicht, so wird der Benutzer mit der folgenden Meldung aufgefordert, die KDCSIGN-Eingabe mit dem alten Passwort zu wiederholen:

```
K097 Angaben zum neuen Passwort sind nicht verwendbar - Bitte anmelden
```

Ist die Gültigkeitsdauer des Passwortes bei der Anmeldung bereits abgelaufen und ist kein Grace-Sign-On generiert, so wird die Anmeldung mit der folgenden Meldung zurückgewiesen:

```
K120 Die Gueltigkeit des Passworts ist abgelaufen - Bitte anmelden
```

Eine Anmeldung an die UTM-Anwendung unter dieser Benutzerkennung ist erst dann wieder möglich, wenn der UTM-Administrator der Kennung ein neues Passwort zugewiesen hat.

Grace-Sign-On

Ist die Gültigkeitsdauer des Passwortes bei der Anmeldung bereits abgelaufen und ist die Anwendung mit Grace-Sign-On generiert (SIGNON ...,GRACE=YES), so wird der Benutzer mit einer K-Meldung darauf hingewiesen, dass sein Passwort nicht mehr gültig ist. Gleichzeitig wird er aufgefordert, sein bisheriges Passwort und ein neues Passwort einzugeben.

Szenarien für die UTM-Berechtigungsprüfung

Das folgende Diagramm zeigt, welche Varianten der UTM-Berechtigungsprüfung möglich sind - abhängig von der KDCDEF-Generierung. Bei fehlerhaften Eingaben gibt openUTM eine spezifische Meldung aus und fordert den Benutzer zu einer neuen Eingabe auf. Werden von einem bestimmten Terminal aus oder unter einer bestimmten Benutzerkennung nacheinander mehrere erfolglose Anmeldeversuche unternommen, dann erzeugt openUTM die Meldung K094 mit dem Standardziel SYSLOG (System-Protokolldatei). Die maximal zulässige Anzahl erfolgloser Anmeldeversuche bis zum Auslösen der Meldung K094 kann mit SIGNON ... SILENT-ALARM= bei der KDCDEF-Generierung festgelegt werden. Ein MSGTAC-Teilprogramm kann auf diese Meldung reagieren.

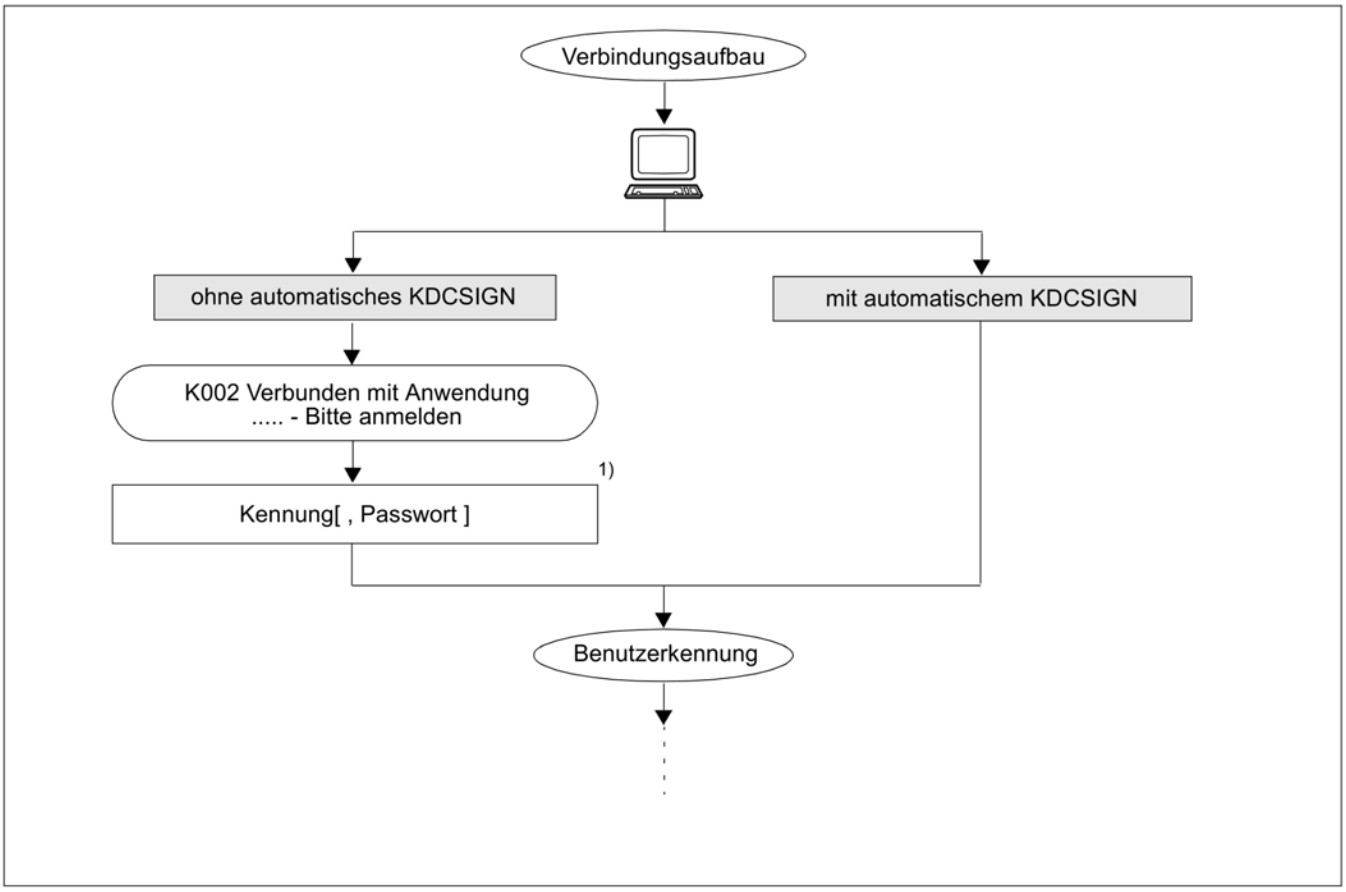


Bild 3: Szenarien der Berechtigungsprüfung bei Anwendungen mit Benutzerkennungen (Teil 1)

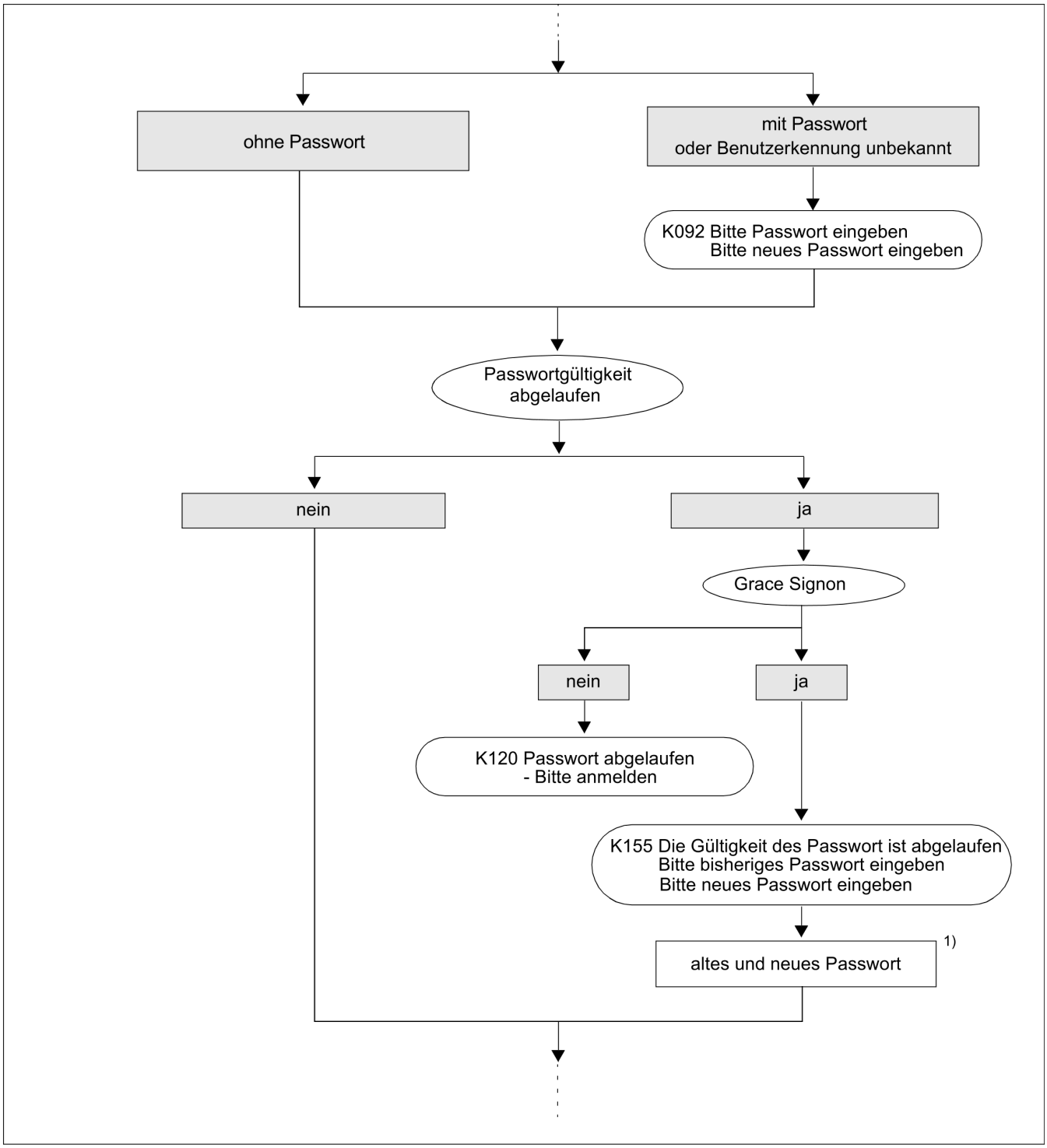


Bild 4: Szenarien der Berechtigungsprüfung bei Anwendungen mit Benutzerkennungen (Teil 2)

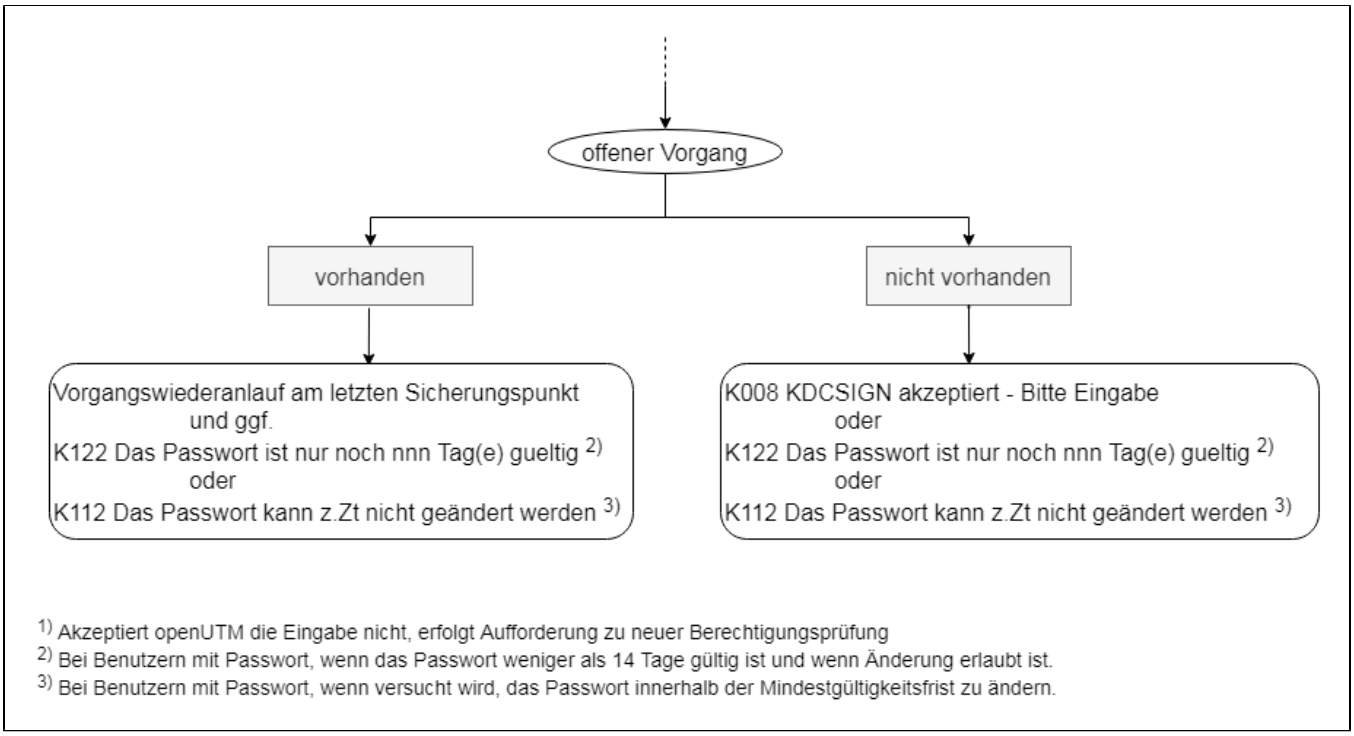


Bild 5: Szenarien der Berechtigungsprüfung bei Anwendungen mit Benutzerkennungen (Teil 3)

8.1.1.4 Automatisches KDCSIGN

Wenn für ein Terminal die KDCDEF-Anweisung LTERM...,USER=*username* angegeben wurde, dann verhält sich openUTM nach dem Verbindungsaufbau so, als ob der Benutzer seine Benutzerkennung schon eingegeben hätte. Wenn für diese Benutzerkennung die Eingabe eines Passworts vorgeschrieben ist, dann fordert openUTM diese Eingabe(n) vom Benutzer an.

Nach Eingabe von KDCOFF BUT kann man an diesem Terminal auch unter einer anderen Kennung arbeiten (siehe Abschnitt „Abmelden von der UTM-Anwendung“).

8.1.2 Anmeldeverfahren für UPIC-Clients und TS-Anwendungen

UPIC-Clients und TS-Anwendungen sind Clients mit Partnertyp UPIC-L, UPIC-R, APPLI oder SOCKET, und im Falle von Partnertyp SOCKET über das UTM Socket Protokoll (USP) kommunizieren.

Bei UPIC-Clients wird die Verbindung durch den Client und bei TS-Anwendungen durch den Client oder durch openUTM aufgebaut, wobei der Verbindungsaufbau durch openUTM nur möglich ist, wenn die TS-Anwendung explizit mit einer PTERM-Anweisung generiert ist.

Falls der Client die Verbindung aufbaut, muss dieser den Namen bzw. im Falle von Partnertyp SOCKET die Portnummer der UTM-Anwendung sowie Rechnernamen und/oder Rechneradresse kennen.

Nach einem erfolgreichen Verbindungsaufbau wird ein UPIC-Client oder eine TS-Anwendung in zwei Schritten angemeldet:

1. Implizites Anmelden über eine **Verbindungs-Benutzerkennung**

   Eine Verbindungs-Benutzerkennung ist dem LTERM-Partner einer TS-Anwendung oder eines UPIC-Clients fest zugeordnet und wird bei der UTM-Generierung explizit oder implizit erzeugt:

   - Explizit durch die Angabe bei USER= in der LTERM-Anweisung.
     Für eine so definierte Verbindungs-Benutzerkennung können mit der KDCDEF-Anweisung USER weitere Eigenschaften festgelegt werden.
   - Implizit durch openUTM, wenn in der LTERM-Anweisung kein USER angegeben wurde oder wenn es sich um einen LTERM-Pool handelt (TPOOL-Anweisung). Als Name der Verbindungs-Benutzerkennung wird dann der LTERM-Name genommen; bei einem LTERM-Pool setzt sich der LTERM-Name aus dem generierten Präfix und einer laufenden Nummer zusammen, z.B. UPIC0025. Für LTERM-Pools können der Verbindungs-Benutzerkennung mit TPOOL ...USER-KSET= spezielle Keycodes zugeordnet werden. Damit lassen sich die Zugriffsmöglichkeiten der Verbindungs-Benutzerkennung einschränken.

   Folgt keine Anmeldung unter einer echten Benutzerkennung, wird die vorläufige Anmeldung der Verbindungs-Benutzerkennung zu einer endgültigen. Dies wird mit einer Meldung protokolliert. Bei UPIC-Clients wird diese Meldung auch ausgegeben, wenn sich dieser anschließend unter einer echten Benutzerkennung anmeldet.
2. Explizites Anmelden über eine **echte Benutzerkennung** (optional)

   Das Verhalten von UPIC-Clients und TS-Anwendungen ist dabei unterschiedlich:

   - Bei UPIC-Clients müssen die Benutzerkennung und die Berechtigungsdaten in den jeweiligen UPIC-Schnittstellenaufrufen gesetzt werden, UPIC übergibt diese Werte anschießend an openUTM. openUTM führt dann die Anmeldung für die übergebene Benutzerkennung durch. Diese ersetzt die Verbindungs-Benutzerkennung für die Dauer der Conversation.

     Am Ende der Conversation wird der Benutzer wieder abgemeldet, falls die Benutzerkennung mit SIGNON OMIT-UPIC-SIGNOFF=NO generiert ist. Ist die Benutzerkennung mit SIGNON OMIT-UPIC-SIGNOFF=YES generiert, dann bleibt der Benutzer nach Ende der Conversation angemeldet und wird erst abgemeldet,

     - wenn vor dem Start einer neuen UPIC Conversation über dieselbe Verbindung im UPIC-Protokoll eine andere Benutzerkennung übergeben wird,
     - oder wenn die Verbindung abgebaut wird.

     Übergibt der UPIC-Client in den UPIC-Schnittstellenaufrufen keine Berechtigungsdaten, dann ist das Anmelden über eine echte Benutzerkennung nur mit einem entsprechenden Anmelde-Vorgang möglich, siehe "Anmeldeverfahren mit Anmelde-Vorgängen".

- Über eine Transportsystemverbindung kann sich ein Benutzer unter einer echten Benutzerkennung nur dann anmelden, wenn für die Anwendung ein entsprechender Anmelde-Vorgang generiert ist, siehe "Anmeldeverfahren mit Anmelde-Vorgängen". Eine Anmeldung mit einer echten Benutzerkennung über den Standard-Anmeldedialog ist nicht möglich.

  Wurde eine TS-Anwendung über eine echte Benutzerkennung angemeldet, dann ersetzt diese Benutzerkennung die Verbindungs-Benutzerkennung für die gesamte Dauer der Verbindung

Sowohl für UPIC-Clients als auch für TS-Anwendungen bleibt die Verbindungs-Benutzerkennung mindestens so lange angemeldet wie die echte Benutzerkennung. Tritt ein Verbindungsverlust auf, kann das dazu führen, dass ein erneuter Verbindungsaufbau abgelehnt wird, wenn unter der echten Benutzerkennung noch ein Programm abläuft und deshalb auch die Verbindungs-Benutzerkennung als angemeldet gilt. In diesem Fall muss der Benutzer mit einer Neuanmeldung warten, bis das Programm beendet ist.

8.1.3 Anmeldeverfahren für OSI TP-Partner

Damit sich ein OSI TP-Partner an die UTM-Anwendung anmelden kann, muss der Partner die Adresse des OSI TP-Zugriffspunktes der UTM-Anwendung kennen. Diese Daten werden im OSI TP-Partner konfiguriert.

Bei OSI TP-Partnern kann die Initiative zum Verbindungsaufbau sowohl vom Partner als auch von openUTM kommen. Dabei können über eine logische Verbindung mehrere parallele Verbindungen aufgebaut werden, die Associations genannt werden. Jeder Association ist ein Association-Name zugeordnet.

Nach erfolgreichem Verbindungsaufbau wird der Client zunächst unter seinem Association-Namen angemeldet. Dieser wird zusammengesetzt aus dem in OSI-LPAP ...,ASSOCIATION-NAMES= angegebenen Namen sowie einer laufenden Nummer, z.B. ASSOC03.

Wenn ein Application Context, der die abstrakte Syntax UTMSEC enthält, für die OSI TP-Kommunikation zwischen beiden Partnern generiert ist (bei openUTM mit dem Parameter APPLICATION-CONTEXT der OSI-LPAP-Anweisung), kann der Kommunikationspartner in den jeweiligen Protokollfeldern eine echte Benutzerkennung und Berechtigungsdaten übergeben. openUTM führt dann die Anmeldung für die übergebene Benutzerkennung durch. Diese Anmeldung gilt für die Dauer des OSI TP-Dialogs. Am Ende des OSI TP-Dialogs wird der Benutzer wieder abgemeldet.

Wird keine echte Benutzerkennung übergeben, dann bleibt der Client unter seinem Association-Namen angemeldet.

8.1.4 Anmeldeverfahren für HTTP-Clients

HTTP-Clients sind Clients, die mit PTYPE=SOCKET generiert wurden, und über das HTTP-Protokoll kommunizieren.

Damit sich ein HTTP-Client an die UTM-Anwendung anmelden kann, muss er die Adresse des Transportsystem-Endpunkts der UTM-Anwendung kennen. Bei HTTP-Clients geht die Initiative zum Verbindungsaufbau immer vom Client aus, dabei sendet ein HTTP-Client einen HTTP-Request an den Transportsystem-Endpunkt.

Nach einem erfolgreichen Verbindungsaufbau wird ein HTTP-Client in zwei Schritten angemeldet:

1. Implizites Anmelden über eine **Verbindungs-Benutzerkennung**

   Eine Verbindungs-Benutzerkennung ist dem LTERM-Partner eines HTTP-Clients fest zugeordnet und wird bei der UTM-Generierung explizit oder implizit erzeugt:

   - Explizit durch die Angabe bei USER= in der LTERM-Anweisung.
     Für eine so definierte Verbindungs-Benutzerkennung können mit der KDCDEF-Anweisung USER weitere Eigenschaften festgelegt werden.
   - Implizit durch openUTM, wenn in der LTERM-Anweisung kein USER angegeben wurde oder wenn es sich um einen LTERM-Pool handelt (TPOOL-Anweisung). Als Name der Verbindungs-Benutzerkennung wird dann der LTERM-Name genommen; bei einem LTERM-Pool setzt sich der LTERM-Name aus dem generierten Präfix und einer laufenden Nummer zusammen, z.B. HTTP0025. Für LTERM-Pools können der Verbindungs-Benutzerkennung mit TPOOL ...USER-KSET= spezielle Keycodes zugeordnet werden. Damit lassen sich die Zugriffsmöglichkeiten der Verbindungs-Benutzerkennung einschränken.

   Folgt keine Anmeldung unter einer echten Benutzerkennung, wird die vorläufige Anmeldung der Verbindungs-Benutzerkennung zu einer endgültigen. Dies wird mit einer Meldung protokolliert.

2. Explizites Anmelden über eine **echte Benutzerkennung** (optional)

   Ein HTTP-Client kann in einem HTTP-Request im Header-Feld Authorization Authentifizierungsdaten an die UTM-Anwendung senden. Ist dieser Header angegeben, dann muss der Wert mit “basic “ beginnen und <userid>:<password> muss base64 codiert sein. Andernfalls wird der Request mit Status-Code "400 invalid http header format“ zurückgewiesen.
   openUTM prüft die übergebenen Authentifizierungsdaten. Schlägt die Prüfung fehl, dann lehnt openUTM die Anmeldung mit "401 user not authorized" ab. Ansonsten wird der Benutzer angemeldet und kann die Services der Anwendung aufrufen, siehe "Vorgänge vom HTTP-Client aus starten".
   Die UTM-Anwendung kann auch verlangen, dass der HTTP-Client Authentifizierungsdaten sendet. Dies kann entweder für alle HTTP-Requests, die über einen Transportsystem-Endpunkt empfangen werden, konfiguriert werden, indem der Parameter USER-AUTH im KDCDEF-Statement BCAMAPPL auf den Wert *BASIC gesetzt oder für die Path-Angabe eines HTTP-Requests, indem der Parameter USER-AUTH im KDCDEF-Statement HTTP-DESCRIPTOR auf den Wert *BASIC gesetzt wird. In diesem Fall fordert openUTM die Authentifizierungsdaten vom Client an, wenn der Client keine Authentifizierungsdaten mitgesendet hat.

8.1.5 Anmeldeverfahren im Internet über Web Services (WS4UTM)

Ein Service einer UTM-Anwendung kann über WS4UTM von Web Service-Clients aus aufgerufen werden. Damit kann ein Benutzer über das Internet auf bestimmte Services einer UTM-Anwendung zugreifen.

Die Anmeldung über WS4UTM kann durch den Web Service-Client gestaltet werden:

1. Der Benutzer gibt in seinem Web Service-Client einen Web Service-Namen und eine Methode an. Durch die Konfiguration ist der Web Service fest mit einer UTM-Anwendung verknüpft. Die Verbindung zur UTM-Anwendung wird über UPIC aufgebaut.
   Eventuell führt der Web Service-Client zuvor einen eigenen Zwischendialog, z.B. für einen Berechtigungsnachweis.
2. Der Benutzer muss eventuell wie beim Terminal die UTM-Benutzerkennung und ggf. das Passwort angeben. Ob der Benutzer einen derartigen Berechtigungsdialog führen muss und wie dieser aussieht, hängt jedoch von der Gestaltung des Web Service-Clients ab. Es ist z.B. möglich, die UTM-Benutzerkennung/-Passwort im Web Service-Client zu „verstecken“ oder sie in der Konfiguration des Web Services vorzugeben, so dass der Berechtigungsdialog intern abläuft.
3. Die Auftragsdaten (TAC und Benutzerdaten) werden zusammen mit den Berechtigungsdaten über http/Soap an einen Web Service-Server und dann über die UPIC-Verbindung an die UTM-Anwendung gesendet. Nach Rückgabe der Antwort an den Web Service-Client wird die UPIC-Verbindung wieder abgebaut.

Als Web Service-Server wird der Axis-Server von Apache verwendet.

Die Kommunikation erfolgt mit Soap-Nachrichten und http-Protokoll über Apache Tomcat und Axis. Für den Anschluss an die UTM-Anwendung nutzt WS4UTM die UPIC-Schnittstelle von openUTM.

Nähere Informationen finden Sie im Handbuch „WebServices for openUTM“.

8.1.6 Anmeldeverfahren im Internet über WebTransactions

Eine UTM-Anwendung kann über WebTransactions an das Internet angeschlossen werden. Damit kann ein Benutzer über einen Browser auf die Services einer UTM-Anwendung zugreifen.

Die Anmeldung über WebTransactions kann durch die WebTransactions-Anwendung gestaltet werden:

1. Der Benutzer gibt in seinem Browser die URL der WebTransactions-Anwendung an. Danach wird die Verbindung zur UTM-Anwendung aufgebaut. Eventuell führt die WebTransactions-Anwendung zuvor einen eigenen Zwischendialog, z.B. für einen Berechtigungsnachweis für den Zugang zur WebTransactions-Anwendung.
2. Der Benutzer muss eventuell wie beim Terminal die UTM-Benutzerkennung und ggf. das Passwort angeben. Ob der Benutzer einen derartigen Berechtigungsdialog führen muss und wie dieser aussieht, hängt jedoch von der Gestaltung der WebTransactions-Anwendung ab. Es ist z.B. möglich, die UTM-Benutzerkennung/-Passwort in der WebTransactions-Anwendung zu „verstecken“, so dass der Berechtigungsdialog intern abläuft und der Benutzer nach der Eingabe der URL sofort angemeldet ist.

Anschließend kann der Benutzer die Services der Anwendung aufrufen, siehe "UTM-Services aufrufen".

WebTransactions nutzt für den Anschluss an die UTM-Anwendung die UPIC-Schnittstelle von UTM. Nähere Informationen finden Sie im WebTransactions-Handbuch „Anschluss an UTM-Anwendungen über UPIC“.

8.1.7 Mehrfach-Anmeldungen unter einer Benutzerkennung

Ist die Benutzerkennung bei der KDCDEF-Generierung mit RESTART=NO und die UTM-Anwendung mit dem Standardwert MULTI-SIGNON=YES generiert worden, dann kann ein Benutzer über verschiedene Verbindungen mehrfach bei der UTM-Anwendung angemeldet sein, allerdings nur einmal über eine Verbindung zu einem Terminal. Mehrfach-Anmeldungen sind nur für echte Benutzerkennungen möglich, **nicht** jedoch für Verbindungs-Benutzerkennungen. Näheres zu Verbindungs-Benutzerkennungen finden Sie auf "Anmeldeverfahren für UPIC-Clients und TS-Anwendungen".

Meldet sich ein Benutzer unter einer mit RESTART=YES generierten Benutzerkennung über einen HTTP-Client oder einen OSI TP-Partner an, für dessen Conversation die Functional Unit „Commit“ ausgewählt ist, ist unter dieser Benutzerkennung ebenfalls eine weitere Anmeldung möglich, weil openUTM in diesem Falle keinen Vorgangswiederanlauf durchführt und die Benutzerkennung dann so behandelt, als sei kein Wiederanlauf generiert. Das gleiche gilt, wenn sich der Benutzer über einen OSI TP-Partner anmeldet und einen asynchronen Auftrag ausführt.

Ansonsten kann sich ein Benutzer unter einer mit RESTART=YES generierten Benutzerkennung zu einem Zeitpunkt nur einmal anmelden, da die für den Vorgangswiederanlauf benötigten Ressourcen der Benutzerkennung zugeordnet werden.

Verhindern von Mehrfach-Anmeldungen für Benutzerkennungen mit RESTART=NO

Über den Parameter MULTI-SIGNON der SIGNON-Anweisung können Sie bei der UTM-Generierung festlegen, dass unabhängig von der Wiederanlaufeigenschaft ein Benutzer zu jedem Zeitpunkt nur einmal bei openUTM angemeldet sein darf.
Diese Festlegung gilt jedoch nicht für Anmeldungen über einen OSI TP-Partner zur Ausführung von Asynchron-Aufträgen.

8.1.8 Anmeldeverfahren mit Anmelde-Vorgängen

Anmelde-Vorgänge, auch als Event-Services SIGNON bezeichnet, sind selbst programmierte Vorgänge, mit deren Hilfe eigene Anmeldeverfahren definiert werden können. Anmelde-Vorgänge können von Terminals, UPIC-Clients und TS-Anwendungen genutzt werden, d.h. von Clients, die über PTERM- oder TPOOL-Anweisung generiert sind (nicht aber von HTTP-Clients).

Aufruf von Anmelde-Vorgängen

Ein Anmelde-Vorgang ist an den Anwendungsnamen gebunden. Meldet sich ein Client unter einem bestimmten Anwendungsnamen an, dann wird der zu diesem Anwendungsnamen gehörige Anmelde-Vorgang gestartet und ersetzt das in den vorigen Abschnitten geschilderte Standard-Anmeldeverfahren. Falls mit BCAMAPPL-Anweisungen mehrere Anwendungsnamen generiert sind, dann können mehrere unterschiedliche Anmelde-Vorgänge in einer Anwendung existieren. Damit lassen sich Client-spezifische Anmelde-Vorgänge erstellen, z.B. einer für Terminals, einer für UPIC-Clients und einer für TS-Anwendungen. Weitere Details finden Sie in den Abschnitten „Anmelde-Vorgang fürTerminals“ bis „Anmelde-Vorgang für UPIC-Clients“.

Ist für einen Anwendungsnamen kein Anmelde-Vorgang generiert, dann durchläuft der Client das Standard-Anmeldeverfahren.

Generieren von Anmelde-Vorgängen

Anmelde-Vorgänge werden wie folgt generiert, siehe auch openUTM-Handbuch „Anwendungen generieren“:

- Mit TAC KDCSGNTC wird der Anmelde-Vorgang für den Standard-Anwendungsnamen (definiert in MAX APPLINAME) generiert.
- Mit BCAMAPPL *appliname2...,*SIGNON=*signon-tac* wird der Anmelde-Vorgang für den Anwendungsnamen *appliname2* generiert. *signon-tac* muss in einer TAC-Anweisung definiert sein.
- Sollen Anmelde-Vorgänge auch von UPIC-Clients genutzt werden können, dann muss zusätzlich SIGNON ..., UPIC=YES generiert werden.

Für jeden dieser TACs ist auch eine PROGRAM-Anweisung nötig. Dort wird der Name des Teilprogramms angegeben, das im Anmelde-Vorgang als erstes durchlaufen wird.

Programmieren von Anmelde-Vorgängen

Für die Programmierung eines Anmelde-Vorgangs gibt es die speziellen KDCS-Aufrufe SIGN ST, SIGN ON und PEND PS. Wie ein Anmelde-Vorgang programmiert werden kann und welche Regeln dabei zu beachten sind, ist ausführlich im entsprechenden Abschnitt im openUTM-Handbuch „Anwendungen programmieren mit KDCS“ beschrieben.

8.1.8.1 Anmelde-Vorgang für Terminals

Ein Anmelde-Vorgang für Terminals setzt sich im allgemeinen aus zwei Teilen zusammen:

Anmelde-Vorgang	
Teil 1	Teil 2
Der Vorgang fordert den Benutzer auf, sich zu legitimieren, liest mit MGET die Berechtigungsdaten und übergibt diese zur Prüfung an openUTM. Der Vorgang ist noch **keiner** Benutzerkennung zugeordnet.	openUTM hat die Berechtigungsdaten akzeptiert und hat den Anmelde-Vorgang der ermittelten Benutzerkennung zugeordnet.

Zwischen dem ersten und zweiten Teil des Anmelde-Vorgangs fügt openUTM ggf. einen Zwischendialog ein, wenn die Gültigkeitsdauer des Passwortes bereits abgelaufen und die Anwendung mit Grace-Sign-On generiert ist. Der Benutzer wird mit einer K-Meldung darauf hingewiesen, dass sein Passwort nicht mehr gültig ist. Gleichzeitig wird er aufgefordert, sein bisheriges Passwort und ein neues Passwort einzugeben.

Spezialfälle des Anmelde-Vorgangs für Terminals

Bei der UTM-Generierung von LTERM-Partnern mit automatischem KDCSIGN muss der Anmelde-Vorgang entsprechend angepasst werden.

LTERM-Partner mit automatischem KDCSIGN

Der Anmelde-Vorgang erhält beim Aufruf SIGN ST die Information, dass die Benutzerkennung bereits bekannt ist. Abhängig von der UTM-Generierung kann jetzt ein Zwischendialog zum Ändern eines Passworts mit abgelaufener Gültigkeitsdauer durchgeführt werden.

8.1.8.2 Anmelde-Vorgang für TS-Anwendungen

Der Anmelde-Vorgang wird unter der Verbindungs-Benutzerkennung gestartet.

Im Anmelde-Vorgang können über den Aufruf SIGN ON die Berechtigungsdaten einer echten Benutzerkennung übergeben werden. Wenn openUTM die Daten akzeptiert, wird der Benutzer beim ordnungsgemäßen Abschluss des Anmelde-Vorgangs unter der angegebenen Benutzerkennung angemeldet. Die Anmeldung wird abgelehnt, wenn die Berechtigungsdaten der TS-Anwendung nicht korrekt sind oder wenn unter der Verbindungs-Benutzerkennung ein Vorgang offen ist.

Ist die Anmeldung unter einer echten Benutzerkennung einmal fehlgeschlagen, dann muss in demselben Anmelde-Vorgang eine erfolgreiche Anmeldung unter einer echten Benutzerkennung folgen, ansonsten wird bei Beendigung des Anmelde-Vorgangs die Verbindung abgebaut. D.h. die Verbindungs-Benutzerkennung ist keine Rückfallstufe für einen erfolglosen Anmeldeversuch.

Wird im Anmelde-Vorgang keine Benutzerkennung übergeben, dann wird der Benutzer beim ordnungsgemäßen Abschluss des Anmelde-Vorgangs endgültig unter der Verbindungs-Benutzerkennung angemeldet.

8.1.8.3 Anmelde-Vorgang für UPIC-Clients

Beim Anmelden über einen Anmelde-Vorgang sind zwei Fälle zu unterscheiden:

- Der UPIC-Client übergibt im UPIC-Protokoll Berechtigungsdaten an openUTM. Wenn openUTM die Daten akzeptiert, dann wird der Anmelde-Vorgang unter der übergebenen echten Benutzerkennung gestartet und der Client wird beim ordnungsgemäßen Abschluss des Anmelde-Vorgangs unter dieser Benutzerkennung angemeldet.
- Falls UPIC-Client im UPIC-Protokoll keine Berechtigungsdaten übergibt, dann wird der Anmelde-Vorgang unter der Verbindungs-Benutzerkennung gestartet. Im Anmelde-Vorgang können die Berechtigungsdaten einer echten Benutzerkennung übergeben werden. Wenn openUTM diese Daten akzeptiert, wird der Benutzer beim ordnungsgemäßen Abschluss des Anmelde-Vorgangs unter dieser Benutzerkennung angemeldet. Werden keine Berechtigungsdaten übergeben, läuft die Conversation unter der Verbindungs-Benutzerkennung.

Ist die Anmeldung unter einer echten Benutzerkennung fehlgeschlagen, dann muss eine erfolgreiche Anmeldung unter einer echten Benutzerkennung folgen, ansonsten wird bei Beendigung des Anmelde-Vorgangs die Conversation beendet. D.h. die Verbindungs-Benutzerkennung ist keine Rückfallstufe für einen erfolglosen Anmeldeversuch.

Damit Client-Programme unabhängig davon eingesetzt werden können, ob die UTM-Anwendung einen Anmelde-Vorgang verwendet oder nicht, können Nachrichten vom Client, die im Anmeldevorgang nicht gelesen wurden, nach Beendigung eines Teilprogramms des Anmelde-Vorgangs mit PEND PA, PEND PR, PEND PS oder PEND FC ohne vorhergehenden MPUT im Folgeteilprogramm gelesen werden.

8.1.8.4 Anwendungsmöglichkeiten für Anmelde-Vorgänge

Anmelde-Vorgänge bieten dem Anwender eine Reihe praktischer Nutzungsmöglichkeiten, die im Folgenden skizziert werden:

- TS-Anwendungen können sich über einen Anmelde-Vorgang mit einer echten Benutzerkennung an eine UTM-Anwendung anmelden. Dadurch werden sie in das Zugangs- und Zugriffskonzept von openUTM eingebunden.
- Ein vom Benutzer eingegebener realer Name kann in eine Benutzerkennung umgesetzt werden, die per UTM-Generierung definiert ist (USER *username*).
- Im Falle eines DB/DC-übergreifenden Berechtigungskonzeptes kann man im 2. Teil des Anmelde-Vorgangs mit einem Datenbankaufruf das aktuelle Berechtigungsprofil für diesen USER aus der Datenbank holen und evtl. in einem Benutzer-spezifischen Langzeitspeicher (ULS) speichern.
- Der Anmelde-Vorgang kann den Benutzer im 2. Teil zur Änderung seines Passwortes auffordern, z.B. weil die Zeitspanne überwacht wird, in welcher der Benutzer dasselbe Passwort verwenden darf.
- Es kann eine Statistik über alle versuchten und erfolgreichen Anmeldungen erstellt werden.
- Auch im Falle eines nachfolgenden Vorgangswiederanlaufs kann der Anmelde-Vorgang dem Benutzer nützliche Informationen geben. Hierzu zählen Bulletin, Anzeige der Tastaturbelegung oder Anzeige des Vorgangswiederanlaufs. Dies erfordert einen zusätzlichen Dialogschritt.
- Wenn openUTM nach einem Aufruf SIGN OB (= KDCOFF BUT per Programm) den Anmelde-Vorgang startet, kann es sinnvoll sein, dass mit MGET die letzte Eingabe vom Terminal gelesen wird, wenn dort bereits neue Berechtigungsdaten eingetragen wurden.

8.1.8.5 Eigenschaften von Anmelde-Vorgängen

Ausgabe der letzten Dialog-Nachricht durch den Anmelde-Vorgang

Liegt kein Vorgangswiederanlauf vor und wird der Anmelde-Vorgang mit MPUT PM und PEND FI beendet, dann wird die letzte Dialog-Nachricht der letzten Sitzung des Benutzers ausgegeben, sofern dieser mit RESTART=YES generiert ist.. Der Benutzer kann dann nach Abschluss des Anmelde-Vorgangs mit dem gleichen Bildschirm weiterarbeiten, mit dem die letzte Sitzung beendet wurde - unabhängig davon, ob dies innerhalb oder außerhalb eines Vorgangs geschah.

Meldungen

Verwendet eine UTM-Anwendung einen Anmelde-Vorgang, werden die folgenden Meldungen nicht erzeugt (und demzufolge auch nicht an SYSLOG und MSGTAC ausgegeben):

K001, K002, K004 bis K008.

Die Meldung K033 (erfolgreiche Anmeldung) wird auch bei Verwendung eines Anmelde-Vorgangs erzeugt.

Fehlversuche beim Anmelde-Vorgang

Beim Anmelde-Vorgang können Fehlversuche des Benutzers beim Anmelden abgefangen werden: Akzeptiert openUTM die eingegebenen Berechtigungsdaten des Benutzers **nicht**, dann kann der Anmelde-Vorgang den Benutzer auffordern, die Eingabe zu wiederholen. Die maximale Anzahl der Eingabeversuche ist programmierbar. Wird sie überschritten, dann sollte sich der Anmelde-Vorgang beenden. Bei TS-Anwendungen und bei Terminals baut dann UTM die Verbindung ab, bei UPIC wird nur die Conversation beendet.

openUTM zählt außerdem alle Fehlversuche von einem Client oder unter einer Benutzerkennung mit, die in ununterbrochener Folge auftreten, auch über eine Folge von Anmelde-Vorgängen hinweg. Nach einer bei der UTM-Generierung festzulegenden Anzahl von fehlerhaften Anmeldeversuchen (siehe KDCDEF-Anweisung SIGNON, Operand SILENT-ALARM im openUTM-Handbuch „Anwendungen generieren“) meldet openUTM dieses Ereignis nach SYSLOG (stiller Alarm, Meldung K094). Anmeldeversuche unberechtigter Personen können aufgedeckt und mit einer MSGTAC-Routine abgewehrt werden.

Fehlverhalten des Anmelde-Vorgangs

openUTM kontrolliert, ob die Regeln für den Anmelde-Vorgang eingehalten werden. Das bietet auch einen Schutz gegen eventuelle Manipulationen der Teilprogramme des Anmelde-Vorgangs. Bei Fehlern dieser Art bricht openUTM den Anmelde-Vorgang mit PEND ER ab. Danach wird bei TS-Anwendungen und Terminals die Verbindung abgebaut; bei UPIC wird nur die Conversation beendet.

8.1.9 Verhalten bei gesperrten Clients/LTERM-Partnern

Verhalten bei gesperrten Clients

Clients können per UTM-Generierung (PTERM...,STATUS=OFF) oder per Administration gesperrt werden. Das Sperren eines Client hat folgende Wirkungen:

- Ein Verbindungswunsch wird abgelehnt.
- Eine bestehende Verbindung bleibt erhalten; die Sperre wird erst dann wirksam, wenn von diesem Client ein neuer Verbindungswunsch kommt.

Verhalten bei gesperrten LTERM-Partnern

LTERM-Partner können per UTM-Generierung (LTERM...,STATUS=OFF) oder per Administration gesperrt werden.

Bei UPIC-Clients und TS-Anwendungen wirkt das Sperren des LTERM-Partners wie das Sperren des Clients.

Bei Terminals hat das Sperren eines LTERM-Partners folgende Wirkungen:

- Der Verbindungswunsch wird ausgeführt, aber nach dem Verbindungsaufbau wird die Meldung ausgegeben:

  ```
  K027 LTERM-Partner &LTERM gesperrt - Administrator verstaendigen oder KDCOFF eingeben.
  ```

- Eine bestehende Verbindung bleibt erhalten, die nächste Eingabe vom Terminal wird mit der Meldung K027 quittiert.

8.2 Anmeldeverfahren ohne Benutzerkennungen

Bei UTM-Anwendungen, für die keine Benutzerkennungen generiert sind, nimmt openUTM keine Berechtigungsprüfung vor. Die Clients werden unter ihrem LTERM-Namen bzw. Association-Namen angemeldet. UPIC-Clients, HTTP-Clients und OpenCPIC-Clients dürfen in diesem Fall keine echte Benutzerkennung übergeben.

Verwendet die UTM-Anwendung Anmelde-Vorgänge ("Anmeldeverfahren mit Anmelde-Vorgängen"), dann kann damit eine Anwendungs-eigene Berechtigungsprüfung durchgeführt werden, zum Beispiel anhand einer Datenbank mit Berechtigungsdaten.

Wird kein Anmelde-Vorgang verwendet, dann kann der Benutzer nach erfolgreichem Verbindungsaufbau zu der UTM-Anwendung sofort mit dieser Anwendung arbeiten. Bei Terminals und TS-Anwendungen (Clients mit Partnertyp APPLI oder SOCKET, die im Falle von Partnertyp SOCKET über das UTM Socket Protokoll (USP) kommunizieren) erhält der Benutzer von openUTM eine Ausgabe, die davon abhängt, ob für diesen LTERM-Partner noch ein offener Vorgang bekannt ist:

- Wenn für den LTERM-Partner in der Anwendung kein offener Vorgang bekannt ist, gibt openUTM folgende Meldung aus:

  ```
  K001 Verbunden mit Anwendung beispiel - Bitte Eingabe
  ```

  Der Benutzer kann dann Vorgänge starten und UTM-Benutzer-Kommandos eingeben.
- Wenn für diesen LTERM-Partner in der Anwendung noch ein offener Vorgang bekannt ist, so bekommt der Benutzer die Ausgabe vom letzten Sicherungspunkt des unterbrochenen Vorgangs auf den Bildschirm, und er kann den Vorgang fortsetzen. Sehen Sie dazu auch openUTM-Handbuch „Anwendungen programmieren mit KDCS“. Dafür ist unter anderem Voraussetzung, dass für diesen LTERM-Partner RESTART=YES generiert wurde. Das heißt aber auch, dass der Benutzer eventuell den Vorgang eines anderen Benutzers fortsetzen kann.

Beachten Sie, dass openUTM in einer Anwendung ohne Benutzerkennungen einen Vorgang an den LTERM-Partner koppelt. Ein unterbrochener Vorgang kann deshalb auch nur vom selben Client fortgesetzt werden. Es sei denn, die Zuordnung LTERM-Partner und physikalischer Client (festgelegt in der PTERM-Anweisung) wird per Administration, z.B. mit dem Administrationskommando KDCSWTCH, geändert.

Das Verhalten bei gesperrten Clients ist das gleiche wie mit Benutzerkennungen, siehe "Verhalten bei gesperrten Clients/LTERM-Partnern".

> **ACHTUNG!**
>
> In einer UTM-Anwendung ohne Benutzerkennungen hat jeder Benutzer die Administrationsberechtigung.

8.3 UTM-Services aufrufen

Ist die UTM-Berechtigungsprüfung erfolgreich verlaufen, so ist der Benutzer berechtigt, mit der UTM-Anwendung zu arbeiten, d.h er kann neue Vorgänge (=Services) starten (siehe unten) oder offene Vorgänge fortsetzen.

Die Abschnitte „Vorgänge vom Terminal aus starten“ bis „Vorgänge von TS-Anwendungen aus starten“ zeigen, wie bei den einzelnen Client-Typen neue Vorgänge gestartet werden. Der Abschnitt „Vorgangswiederanlauf“ beschreibt die Situation, wenn für diese Benutzerkennung in der Anwendung noch ein offener Vorgang bekannt ist.

8.3.1 Vorgänge vom Terminal aus starten

Nach erfolgreicher Anmeldung kann der Benutzer einen Vorgang starten, indem er einen Transaktionscode (TAC) eingibt oder eine entsprechend generierte Funktionstaste drückt.

Starten eines Vorgangs durch Eingabe eines Transaktionscodes

Wenn kein Anmelde-Vorgang durchlaufen wird, dann gibt openUTM nach dem Anmelden die folgende Meldung aus:

```
K008 Anmeldung akzeptiert - Bitte Eingabe
```

Der Benutzer kann einen Vorgang starten, indem er einen TAC und eventuell eine Nachricht eingibt. Die ersten acht Zeichen seiner Eingabe werden von openUTM als TAC interpretiert. Ist der TAC kürzer als 8 Zeichen, dann muss er durch ein Leerzeichen von der Nachricht getrennt sein.

Wenn ein Anmelde-Vorgang durchlaufen wird, dann bestimmt der Anmelde-Vorgang den nächsten Schritt. Der Benutzer erhält dann eine Ausgabe oder es wird direkt ein Vorgang gestartet.

Tastaturbelegung bei Terminals auf Unix- und Linux-Systemen

Es gilt folgende Tastaturbelegung:

Taste	**Wirkung**
Korrektur	Wie in der Shell
return	Die Daten werden an die Anwendung gesendet.
END DEL	Der Dialog wird beendet und der Benutzer von der Anwendung abgemeldet. Die Taste DEL wirkt **nicht**, wenn sich der Benutzer mit dem Schalter -D angemeldet hat.
andere Systemtasten	Es werden sowohl die Escape-Sequenzen der Taste als auch die Daten an die Anwendung gesendet, daher auf **keinen** Fall verwenden.

Tastaturbelegung auf Windows-Systemen

Wenn man auf Windows-Systemen an der Console über die Eingabeaufforderung mit der Anwendung arbeitet, gilt folgende Tastaturbelegung:

Taste	**Wirkung**
Korrektur	Wie in der Eingabeaufforderung
return	Die Daten werden an die Anwendung gesendet.
END CTRL+C	Der Dialog wird beendet und der Benutzer von der Anwendung abgemeldet.
DEL	Die Taste wird ignoriert
Cursor-Tasten	Wie in der Eingabeaufforderung, d.h. Historie der letzten Eingaben
andere Systemtasten	Es werden sowohl die Escape-Sequenzen der Taste als auch die Daten an die Anwendung gesendet, daher auf **keinen** Fall verwenden.

Eingabe ungültiger Transaktionscodes

Wenn der Benutzer einen falschen TAC eingibt, dann erhält er die Meldung

```
K009 Der Transactionscode <tac> ist ungültig - Bitte Eingabe
```

Falls in der Anwendung ein Dialog-Vorgang BADTACS generiert ist, dann wird stattdessen der Vorgang BADTACS gestartet. Nachdem der Dialog-Vorgang BADTACS beendet
wurde, bleibt der Benutzer angemeldet und kann wie oben beschrieben einen Vorgang starten.

8.3.2 Vorgänge vom UPIC-Client und OSI TP-Partner aus starten

Nach dem Verbindungsaufbau können die OSI TP-Partner oder UPIC-Clients Vorgänge starten. Dazu wird der TAC durch den Client gesetzt, z.B. über die Funktion *Set_TP_Name* der CPI-C-Schnittstelle oder durch einen entsprechenden Eintrag in der Side Information Datei. Dieser TAC wird an openUTM übergeben, eventuell zusammen mit Berechtigungsdaten. Nach erfolgreicher Berechtigungsprüfung gilt:

- Bei OSI TP-Partnern und bei UPIC-Clients ohne Anmelde-Vorgang wird der zum TAC gehörige Vorgang sofort gestartet.
- Bei UPIC-Clients mit Anmelde-Vorgang wird der zum TAC gehörige Vorgang erst nach dem Ende des Anmelde-Vorgangs gestartet.

Am Ende des Vorgangs wird der Benutzer wieder abgemeldet, wenn er sich für diesen Vorgang unter einer echten Benutzerkennung angemeldet hat. Dies gilt nicht für UPIC-Clients in einer UTM-Anwendung, die mit SIGNON OMIT-UPIC-SIGNOFF=YES generiert wurde, siehe "Anmeldeverfahren für UPIC-Clients und TS-Anwendungen".

8.3.3 Vorgänge vom HTTP-Client aus starten

Nach dem Verbindungsaufbau können HTTP-Clients Vorgänge starten. Details zur Angabe der Berechtigungsdaten und des TACs sind im openUTM-Handbuch "Konzepte und Funktionen" sowie im openUTM-Handbuch "Anwendungen programmieren mit KDCS" beschrieben.

8.3.4 Vorgänge von TS-Anwendungen aus starten

TS-Anwendungen (Clients mit Partnertyp APPLI oder SOCKET, die im Falle von Partnertyp SOCKET über das UTM Socket Protokoll (USP) kommunizieren) verhalten sich ähnlich wie Terminals:

- Ohne Anmelde-Vorgang erhält die TS-Anwendung die Meldung K001, falls dieser Meldung das Meldungsziel PARTNER zugewiesen wurde, siehe Beschreibung des Tools KDCMMOD im openUTM-Handbuch „Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen“.

  Anschließend kann die TS-Anwendung einen Vorgang starten, indem sie einen TAC und eventuell eine Nachricht an die UTM-Anwendung übergibt. Dazu werden die ersten 8 Zeichen der Nachricht als TAC interpretiert. Ist der TAC kürzer als 8 Zeichen, muss er durch Leerzeichen von der Nachricht getrennt sein.
- Wird ein Anmelde-Vorgang durchlaufen, dann bestimmt dieser den nächsten Schritt. Der Anmelde-Vorgang kann entweder direkt einen Vorgang starten oder eine Ausgabe-Nachricht an die TS-Anwendung schicken. Wird eine Ausgabe-Nachricht an die TS-Anwendung geschickt, dann muss die nächste Nachricht in den ersten 8 Zeichen einen TAC enthalten, d.h. es gilt das gleiche wie ohne Anmelde-Vorgang, siehe oben.

Nach dem Beenden des Vorgangs kann der nächste Vorgang gestartet werden.

8.3.5 Vorgangswiederanlauf

Ein Vorgangswiederanlauf wird in der Regel durchgeführt, wenn:

- sich ein Client unter einer Benutzerkennung anmeldet, die mit RESTART=YES generiert ist,
- und für diese Benutzerkennung in der Anwendung noch ein offener Vorgang bekannt ist.

Wurde am letzten Sicherungspunkt eine Nachricht an dem Client gesendet, so sendet openUTM diese Nachricht erneut an den Client; der Benutzer kann den Vorgang anschließend fortsetzen. Ansonsten wird der offene Vorgang sofort fortgesetzt.

Abhängig von der Art des Client und vom Anmeldeverfahren gilt für den Vorgangswiederanlauf Folgendes:

- Standard-Anmeldeverfahren für Terminals und TS-Anwendungen:
  openUTM führt den Vorgangswiederanlauf automatisch durch.
- Standard-Anmeldeverfahren für UPIC-Clients und OSI TP-Partner:
  Der Client muss eine spezielle Conversation starten, die mittels des UTM-Benutzerkommandos KDCDISP den Wiederanlauf anfordert (siehe z.B. Handbuch „openUTM-Client für das Trägersystem UPIC“). Über OSI TP-Partner ist kein Fortsetzen des Vorgangs möglich, wenn die Functional Unit „Commit“ ausgewählt wurde.
- Existiert ein offener Vorgang für den Benutzer und es wird kein Vorgangswiederanlauf angefordert, so wird der offene Vorgang abormal beendet.
- Anmelden über Anmelde-Vorgang:
  Der Anmelde-Vorgang muss den Wiederanlauf initiieren oder den offenen Vorgang abnormal beenden.

> ¡ In einer Anwendung mit Benutzerkennungen ist ein Vorgang an die Benutzerkennung gebunden. Deshalb kann der Benutzer einen unterbrochenen Vorgang auch an einem anderen Client fortsetzen, sofern der LTERM-Partner des Client die Berechtigung dazu hat und der Typ des Clients gleich bleibt.

8.4 Berechtigungskonzept von openUTM

openUTM bietet zusätzlich zum Zugangsschutz durch Benutzerkennungen ein ausgefeiltes Zugangs- und Zugriffskonzept. Damit lässt sich steuern, welcher Benutzer über welche LTERM-Partner auf welche Services der UTM-Anwendung zugreifen darf.

Dabei kann zwischen einer benutzerorientierten Variante (**Lock-/Keycode**-Konzept) und einer rollenorientierten Variante (**Access-List-**Konzept) gewählt werden. Diese Varianten werden mit Hilfe von Lockcodes, Access-Lists, Keysets und Keycodes generiert:

- Ein Service wird entweder mit Lockcodes (Lock-/Keycode-Konzept) oder mit einer Access-List (Access-List-Konzept) geschützt (TAC-Anweisung, LOCK= bzw. ACCESS-LIST=).
- Eine Benutzerkennung erhält ein Keyset mit einem oder mehreren Keycodes (USER-Anweisung, KSET=). Die Keycodes definieren die Berechtigungen.
- Ein LTERM-Partner erhält ein Keyset mit einem oder mehreren Keycodes sowie Lockcodes, wenn das Lock-/Keycode-Konzept verwendet wird (LTERM- bzw. TPOOL-Anweisung, Operanden KSET= und LOCK=).
- Keysets werden eigens in KSET-Anweisungen definiert

Die folgende Tabelle zeigt für die beiden Konzept-Varianten, unter welchen Voraussetzungen ein Benutzer sich anmelden kann und wann er einen Vorgang starten oder fortsetzen darf (nach einem Vorgangswiederanlauf).

Tätigkeit	**Voraussetzungen**	
	Lock-/Keycode-Konzept	**Access-List-Konzept**
Anmelden über bestimmten LTERM-Partner	Ein Keycode der Benutzerkennung stimmt mit dem Lockcode des LTERM-Partners überein.	Anmelden immer möglich.
Starten eines Vorgangs	Benutzerkennung und LTERM-Partner besitzen einen Keycode, der mit dem Lockcode des TACs übereinstimmt.	Benutzerkennung und LTERM-Partner besitzen jeweils einen Keycode, der in der Access-List des TACs enthalten ist. Die Keycodes von Benutzerkennung und LTERM müssen nicht identisch sein.
Vorgang fortsetzen (nach Vorgangswiederanlauf)	Ein Keycode des LTERM-Partners, über den der Benutzer den Vorgang fortsetzt, muss mit dem Lockcode des Folge-TACs übereinstimmen.	Ein Keycode des LTERM-Partners, über den der Benutzer den Vorgang fortsetzt, muss in der Access-List des Folge-TACs enthalten sein.

Meldungen bei fehlenden Berechtigungen

Wenn Berechtigungen fehlen, kann der Terminalbenutzer folgende Meldungen erhalten (beim Anmelde-Vorgang kommt ein entsprechender Returncode):

`K005 Die Benutzerkennung` *`user`* `ist gesperrt - Bitte Anmelden`

wenn der Keycode des Benutzers nicht mit dem Keycode des LTERM-Partners übereinstimmt (Anmelde-Vorgang: Returncode U02).

`K009 Der Transactionscode <tac> ist ungültig - Bitte Eingabe`

wenn der Benutzer oder das LTERM nicht die Berechtigung haben, den Vorgang zu starten. Wenn ein BADTAC-Vorgang generiert ist, dann wird stattdessen der BADTAC-Vorgang gestartet.

```
K123 LTERM hat nicht die Berechtigung den Vorgang fortzusetzen - Bitte Anmelden
```

wenn der LTERM-Partner, über den sich der Benutzer beim Vorgangswiederanlauf angemeldet hat, nicht die Berechtigung hat, den Folge-TAC zu starten (Anmelde-Vorgang: Returncode U16). Diese Meldung kann insbesondere dann auftreten, wenn ein Benutzer den Vorgang von einem anderen Terminal und damit einem anderen LTERM fortsetzt.

Weitere Informationen finden Sie im openUTM-Handbuch „Konzepte und Funktionen“ sowie im openUTM-Handbuch „Anwendungen generieren“.

8.5 Abmelden von der UTM-Anwendung

Die folgenden Abschnitte beschreiben die verschiedenen Möglichkeiten, wie sich ein Client bei der UTM-Anwendung abmelden kann oder ein Client durch UTM abgemeldet wird. Dabei unterscheiden sich Terminals von allen anderen Clients, denn nur von Terminals aus können sich Benutzer explizit von der Anwendung abmelden.

Abmelden bei Zeitüberschreitung - Zeitüberwachung

Bei der UTM-Generierung können maximale Wartezeiten festgelegt werden durch:

- die Operanden TERMWAIT= (PEND KP-Timer) und PGWTTIME= (PGWT-Timer) in der KDCDEF-Steueranweisung MAX
- den Operanden IDLETIME= (Transaktionsende-Timer) der PTERM- oder TPOOL-Anweisung bzw. der OSI-LPAP-Anweisung bei OSI TP-Partnern.

Wenn eine mit diesen Timern eingestellte Wartezeit abgelaufen ist, dann wird bei Terminals folgende Meldung ausgegeben:

```
K021 Eine Eingabe ist nicht in der vorgegebenen Zeit erfolgt
```

Anschließend meldet openUTM die Benutzerkennung ab und baut die Verbindung zum Client ab. Der Client kann sich danach wieder bei der Anwendung anmelden und einen eventuell offenen Vorgang fortsetzen, siehe Abschnitt „Vorgangswiederanlauf“.

Abmelden mit KDCOFF-Kommando

Ein Terminal-Benutzer kann sich durch Eingabe des UTM-Kommandos KDCOFF bzw. KDCOFF BUT von der UTM-Anwendung abmelden. Sehen Sie hierzu auch das UTM-Benutzerkommando KDCOFF auf "KDCOFF - Abmelden von einer UTM-Anwendung".

KDCOFF aus einem Programm

openUTM bietet die Funktionsaufrufe SIGN OF und SIGN OB, mit denen in einem Dialog-Teilprogramm die Wirkung des Benutzerkommandos KDCOFF bzw. KDCOFF BUT ausgelöst werden kann. SIGN OF/OB ist für Terminals, UPIC-Clients und TS-Anwendungen möglich. In Teilprogrammen, die für einen OSI TP-Partner laufen, sind diese Aufrufe nicht erlaubt.

SIGN OF und SIGN OB wirken wie folgt:

SIGN-Aufruf	Kommando	Auswirkung
SIGN OF	KDCOFF	openUTM baut Verbindung zum Client ab.
SIGN OB	KDCOFF BUT	Für Terminals bleibt die Verbindung bleibt erhalten, der Benutzer wird abgemeldet. Bei UPIC-Clients und TS-Anwendungen wird die Verbindung abgebaut (wie SIGN OF).

Der Aufruf wirkt bei Terminals und UPIC-Clients/TS-Anwendungen unterschiedlich:

- Bei Terminals gibt openUTM zunächst die MPUT-Nachricht und die Meldung K095 an das Terminal aus. Erst durch die nächste (beliebige) Eingabe vom Terminal wird der Benutzer abgemeldet und (bei SIGN OF) die Verbindung abgebaut.

- Bei UPIC-Clients und TS-Anwendungen wird die MPUT-Nachricht gesendet und anschließend sofort der Verbindungsabbau initiiert.

Im Folgenden werden einige Anwendungsmöglichkeiten für den Funktionsaufruf SIGN OF/OB skizziert:

- Anwendungen mit besonderen Sicherheitsanforderungen. Nach dem Anmelden darf ein Benutzer nur einen einzigen Vorgang bearbeiten.
- Der Steuerteil des Bildschirms bietet als mögliche nächste Aktionen auch „Abmelden“ oder „Neu-Anmeldung“ an. Das Folgeteilprogramm erzeugt dann abhängig von der Eingabe einen Aufruf SIGN OF bzw. SIGN OB. Nach der Dialogausgabe dieses Teilprogramms und der darauf folgenden Eingabe wird entweder die Verbindung zum Terminal abgebaut oder aber der Anmelde-Vorgang gestartet.
- Verbesserter Datenschutz: Der Bildschirm kann durch den letzten MPUT überschrieben werden, es bleiben keine Vorgangs-spezifischen Daten am Bildschirm stehen.

8.6 UTM-Benutzerkommandos für Terminals

Dieser Abschnitt enthält eine Beschreibung aller UTM-Benutzerkommandos, die dem Terminal-Benutzer zur Verfügung stehen:

- KDCOUT, um asynchrone Nachrichten anzufordern
- KDCDISP, um die letzte Dialog-Nachricht nochmals anzufordern
- KDCLAST, um die letzte Ausgabe zu wiederholen
- KDCOFF, um sich abzumelden

8.6.1 KDCOUT - Asynchrone Nachricht ausgeben

Mit dem KDCOUT-Kommando kann der Benutzer die Ausgabe asynchroner Nachrichten anfordern.

openUTM kündigt asynchrone Nachrichten mit der folgenden Meldung an:

```
K012 nnn Nachricht(en) vorhanden
```

Die Meldung erscheint zusammen mit einer Dialogausgabe an dieses Terminal in der Systemzeile. Mit *nnn* wird die Anzahl der asynchronen Nachrichten angegeben. Der Benutzer kann diese Nachrichten mit dem Kommando KDCOUT abholen. Sind bei Eingabe von KDCOUT keine Nachrichten für das Terminal vorhanden, so meldet openUTM:

```
K020 keine Nachricht vorhanden
```

Wird eine asynchrone Nachricht mit KDCOUT abgeholt, dann wird sie durch die nächste Eingabe gelöscht, außer bei Eingabe von KDCLAST (siehe "KDCLAST - Letzte Ausgabe wiederholen").

Die KDCDEF-Anweisung LTERM ..., RESTART= NO hat zur Folge, dass beim Verbindungsaufbau oder -abbau zu diesem LTERM-Partner anstehende asynchrone Nachrichten gelöscht werden.

Die Funktionsvarianten von openUTM haben folgende Auswirkungen auf die Behandlung von asynchronen Nachrichten:

- Bei UTM-S-Anwendungen bleiben asynchrone Nachrichten auch über Unterbrechungen des Anwendungslaufes hinweg gesichert, bis sie mit KDCOUT abgeholt werden.
- Bei UTM-F-Anwendungen werden asynchrone Nachrichten nur während des Anwendungslaufes gespeichert. Sie gehen mit der Beendigung des Anwendungslaufes verloren.

8.6.2 KDCDISP - Letzte Dialog-Nachricht ausgeben

In einer laufenden UTM-Anwendung kann sich ein Benutzer mit dem KDCDISP-Kommando die letzte Dialog-Nachricht noch einmal ausgeben lassen.

Gibt der Benutzer das KDCDISP-Kommando nach Abschluss des Anmelde-Vorgangs oder nach der Rückkehr von einem eingeschobenen Vorgang ein, dann zeigt openUTM nochmals den letzten Bildschirm der letzten Session, bzw. den letzten Bildschirm des unterbrochenen Vorgangs.

Das KDCDISP-Kommando ist in folgenden Situationen nützlich:

- Aufgrund von Fehlbedienung am Terminal ist der Bildschirminhalt nach einer Dialogausgabe teilweise oder vollständig zerstört.
- Der Benutzer hat während der Bearbeitung eines Vorgangs asynchrone Nachrichten am Bildschirm erhalten. Die asynchronen Nachrichten wurden entweder durch openUTM direkt gesendet oder der Benutzer hat sie mit KDCOUT selbst angefordert. Wenn der Benutzer danach den offenen Vorgang fortsetzen möchte, lässt er sich dazu mit einem KDCDISP-Kommando die letzte Dialogausgabe nochmals ausgeben.
- Nach Beendigung und erneutem Start der UTM-Anwendung kann der Benutzer (zur Orientierung) mit dem KDCDISP-Kommando die letzte Dialogausgabe des vor Beendigung der Anwendung abgeschlossenen Vorgangs wiederholen. Das gilt aber nur, wenn es sich um eine UTM-S-Anwendung handelt und wenn der Vorgangswiederanlauf nicht explizit ausgeschaltet wurde durch die KDCDEF-Anweisung USER ..., RESTART=NO (bzw. LTERM ...,RESTART=NO, wenn die Anwendung ohne Benutzerkennungen generiert wurde).

8.6.3 KDCLAST - Letzte Ausgabe wiederholen

Das Kommando KDCLAST ermöglicht die Wiederholung der letzten Ausgabe-Nachricht am Terminal unabhängig davon, ob es eine Dialog-Nachricht oder eine asynchrone Nachricht war.

Wurde als letztes eine asynchrone Nachricht ausgegeben, so wird diese Ausgabe mit KDCLAST wiederholt. Die asynchrone Nachricht wird dabei aber noch nicht freigegeben.

Wird das Kommando KDCLAST nach Abschluss des Anmelde-Vorgangs eingegeben, dann zeigt openUTM nochmals den letzten Bildschirm des Anmelde-Vorgangs. Wird es nach der Rückkehr von einem eingeschobenen Vorgang eingegeben, wird der letzte Bildschirm des eingeschobenen Vorgangs ausgegeben.

8.6.4 KDCOFF - Abmelden von einer UTM-Anwendung

Der Benutzer kann sich durch Eingabe des UTM-Kommandos KDCOFF von der UTM-Anwendung abmelden. Dadurch wird die Verbindung zwischen Dialog-Terminalprozess und UTM-Anwendung abgebaut und der Dialog-Terminalprozess beendet.

Wurde der Dialog-Terminalprozess nach erfolgreichem Anmelden automatisch durch das Unix- oder Linux-System gestartet (siehe "Starten des Dialog-Terminalprozesses durch das Unix- oder Linux-System"), so wird auch der Dialog mit dem System beendet.

Meldet er sich während der Bearbeitung eines Vorgangs am Transaktionsende ab, wird die Bearbeitung unterbrochen. Sie kann nach späterem Anmelden an die UTM-Anwendung wieder fortgesetzt werden, sofern der Benutzer mit RESTART=YES generiert ist.
.

KDCOFF BUT

Durch Eingabe von KDCOFF BUT kann der Benutzer sich so abmelden, dass die Verbindung zwischen Dialog-Terminalprozess und UTM-Anwendung bestehen bleibt. Er wird gleich zum erneuten Anmelden aufgefordert, bzw. der Anmelde-Vorgang wird gestartet.

Meldungen

Nach Eingabe von KDCOFF [BUT] reagiert openUTM mit einer der Meldungen:

`K019 KDCOFF von Anwendung beispiel akzeptiert`

Der Benutzer hat KDCOFF eingegeben, oder er hat in einer Anwendung ohne Benutzerkennungen KDCOFF BUT eingegeben. Das Terminal ist nicht mehr mit der UTM-Anwendung verbunden.

`K018 KDCOFF von Anwendung beispiel akzeptiert - Bitte Anmelden`

In einer Anwendung mit Benutzerkennungen und ohne Anmelde-Vorgang hat der Benutzer KDCOFF BUT eingegeben. openUTM fordert ihn zum erneuten Anmelden mit einer Benutzerkennung auf. Dies gilt auch dann, wenn der Benutzer sich ohne Schalter -S angemeldet hat.

`K003 Das Kommando KDCOFF ist in dieser Situation nicht erlaubt`

Die Eingabe erfolgte nach einem PEND KP-Aufruf oder blockierenden Aufruf (z.B. PGWT) des Teilprogramms.

9 Programmaustausch im Betrieb

Für den Austausch von Programmen im laufenden Betrieb bietet openUTM zwei unterschiedliche Verfahren an:

- Austausch des gesamten Anwendungsprogramms mit Hilfe des Tools KDCPROG und des Administrationskommandos KDCAPPL bzw. eines Administrationsprogramms, das KDCADMI mit Operationscode KC_CHANGE_APPLICATION aufruft.
- Austausch von Shared Objects, d.h. von Teilen des Anwendungsprogramms mit Hilfe des Administrationskommandos KDCPROG bzw. eines Administrationsprogramms, das KDCADMI mit Operationscode KC_MODIFY_OBJECT und Objekttyp KC_LOAD_MODULE aufruft.

Die Verfahren können gemischt werden, d.h. eine Anwendung, die Shared Objects enthält, kann auch als Ganzes ausgetauscht werden. Beide Verfahren werden im Folgenden beschrieben.

9.1 Anwendung austauschen

openUTM bietet Ihnen die Möglichkeit, das Anwendungsprogramm während des Anwendungslaufs auszutauschen. D.h. Sie können z.B. Teilprogramme Ihres Anwendungsprogramms verändern, eine neue Version des Anwendungsprogramms erstellen und diese Version des Anwendungsprogramms in Betrieb nehmen, ohne dass Sie den Anwendungslauf beenden müssen.

Die auszutauschende Anwendung kann sowohl nur aus statisch in *utmwork* eingebundenen Teilprogrammen bestehen als auch Teilprogramme enthalten, die als Shared Objects gebunden sind.

Bei den Änderungen des Anwendungsprogramms ist Folgendes zu beachten:

- In Anwendungen **ohne** Shared Objects dürfen die Änderungen keine Auswirkung auf die KDCDEF-Generierung und die KDCFILE haben. D.h. die Funktion „Anwendungsaustausch" können Sie **nicht** anwenden, wenn Sie das Anwendungsprogramm um neue Teilprogramme erweitern wollen. Werden neue Teilprogramme hinzugefügt, dann sind diese nicht in den Tabellen der KDCFILE enthalten.
- In Anwendungen **mit** Shared Objects können Sie neue Teilprogramme ins Anwendungsprogramm aufnehmen. Diese Teilprogramme müssen jedoch in Shared Objects gebunden werden, die in der Konfiguration der Anwendung enthalten sind. Die Teilprogramme und die zugehörigen Transaktionscodes müssen per Administration in die Tabellen der KDCFILE eingetragen werden.

Fehlen nach dem Austausch Teilprogramme, die in dem vorherigen Anwendungsprogramm vorhanden waren, so können Aufträge für TACs bei der Anwendung eintreffen, für die kein Teilprogramm mehr vorhanden ist. Diese Aufträge beendet openUTM mit PEND ER. Die Transaktionscodes können vom UTM-Administrator aus der Konfiguration gelöscht werden.

Die verschiedenen Versionen Ihres Anwendungsprogramms und den Austausch verwaltet das Tool KDCPROG (siehe Abschnitt „Das Tool KDCPROG“). KDCPROG verwaltet die Versionen des Anwendungsprogramms in einem Dateigenerationsverzeichnis (FGG=**F**ile **G**eneration **G**roup).

Wird ein als Shared Object vorliegendes Teilprogramm ausgetauscht, so wird zunächst das alte Shared Object durch das neue ersetzt. Das führt dazu, dass die Verweise vom statischen Teil des Anwendungsprogramms auf das Shared Object dann unbefriedigt sind. Erst wenn der UTM-Administrator mit dem Administrationskommando KDCAPPL PROG=NEW | OLD den Programmaustausch für das Anwendungsprogramm initiiert, werden alle unbefriedigten Verweise auf das Shared Object befriedigt und das neue Shared Object im Anwendungsprogramm wirksam.

> ¡ Mit KDCAPPL PROG=SAME können Sie das Anwendungprogramm neu laden. D.h. bei Verwendung von Dateigenerationen wird wieder das Programm der gleichen Dateigeneration geladen.

9.1.1 Voraussetzungen für den Anwendungsaustausch

Folgende Arbeitsschritte müssen Sie für den Anwendungsaustausch durchführen:

1. **Dateigenerationsverzeichnis PROG einrichten**

   Damit Sie ein UTM-Anwendungsprogramm im laufenden Betrieb austauschen können, sollten Sie die verschiedenen Versionen des Anwendungsprogramms (auch das aktuell geladene) mit Hilfe des Tools KDCPROG verwalten. Dazu müssen Sie mit KDCPROG im Basisverzeichnis *filebase* der Anwendung das Dateigenerationsverzeichnis PROG anlegen (Funktion KDCPROG CREATE). Falls Sie kein Dateigenerationsverzeichnis erstellt haben, lädt KDCAPPL PROG= (bzw. der entsprechende Aufruf an der Programmschnittstelle zur Administration) das Anwendungsprogramm *utmwork* aus dem Basisverzeichnis *filebase* neu.

   In dem Dateigenerationsverzeichnis (im Folgenden FGG genannt; **F**ile **G**eneration **G**roup) verwaltet KDCPROG die verschiedenen Versionen Ihres Anwendungsprogramms. Sehen Sie dazu Abschnitt „Dateigenerationsverzeichnis PROG“. Dazu müssen Sie jedes *utmwork*-Programm als eine Generation in der FGG ablegen.

   Die FGG müssen Sie nur einmal einrichten. Sie bleibt erhalten. In ihr können Sie mit Hilfe von KDCPROG mehrere Versionen Ihres Anwendungsprogramms verwalten.

   Die FGG können Sie einrichten, bevor oder nachdem Sie die erste Version des Anwendungsprogramms erstellt haben. Das Anwendungsprogramm, d.h. *utmwork*, erzeugen Sie wie im Kapitel „Anwendungsprogramm erzeugen“ beschrieben.

   > **ACHTUNG!**
   >
   > Für jede Anwendung kann nur eine FGG für den Anwendungsaustausch existieren. Eine bereits vorhandene FGG und die darin enthaltenen Versionen des Anwendungsprogramms werden gelöscht, wenn mit KDCPROG CREATE eine neue FGG eingerichtet wird.

2. **Anwendungsprogramm in die FGG transferieren**

   Sie transferieren Ihr Anwendungsprogramm mit KDCPROG TRANSFER in die FGG PROG (sehen Sie dazu auch Abschnitt „TRANSFER - utmwork in die FGG übertragen“). Sie müssen dem Anwendungsprogramm dabei die Version - d.h. die absolute Generationsnummer - 0001 zuordnen.

   Zusätzlich muss mit KDCPROG SWITCH die Basis der FGG auf die Generationsnummer 0001 umgeschaltet werden.

   Danach können Sie die Anwendung starten, wie im Kapitel „UTM-Anwendung starten“ beschrieben. openUTM lädt *utmwork* aus der FGG.

3. **Weitere Versionen des Anwendungsprogramms erstellen und in die FGG transferieren**

   Unabhängig davon, ob Ihre Anwendung gestartet ist oder nicht, können Sie weitere Versionen Ihres Anwendungsprogramms erstellen. *utmwork* transferieren Sie mit KDCPROG TRANSFER in die FGG. Dabei ordnen Sie jeder Version des Anwendungsprogramms eine Generationsnummer zu. Die Generationsnummern in der FGG müssen fortlaufend ansteigend sein.

> **ACHTUNG!**
>
> Existiert in der FGG bereits eine Generation des Anwendungsprogramms mit der Generationsnummer, die Sie beim Transfer einer neuen Generation angeben, so wird dieses ohne Warnung überschrieben.

Wenn diese Voraussetzungen erfüllt sind, können Sie den Anwendungsaustausch jederzeit und beliebig oft anfordern. Dazu stehen Ihnen folgende Möglichkeiten zur Verfügung:

das Administrationskommando KDCAPPL PROG=... und an der Programmschnittstelle zur Administration der KDCADMI-Operationscode KC_CHANGE_APPLICATION
Beide Möglichkeiten sind im openUTM-Handbuch „Anwendungen administrieren“ beschrieben.

Wenn im Folgenden auf Aktionen hingewiesen wird, die mit KDCAPPL PROG= durchgeführt werden können, dann gilt dies auch für Administrationsprogramme, die KDCADMI mit Operationscode KC_CHANGE_APPLICATION absetzen.

Beim Start lädt openUTM die Version des Anwendungsprogramms, dessen Generationsnummer Basisnummer der FGG ist. Sehen Sie dazu Abschnitt „Dateigenerationsverzeichnis PROG“.

Bei einem Programmaustausch während des Anwendungslaufs setzt openUTM die Basis auf die aktuell geladene Version. So ist gewährleistet, dass der nächste Start mit der zuletzt geladenen Version des Anwendungsprogramms erfolgt.

Hinweis zum Programm-Austausch in einer UTM-Cluster-Anwendung

In einer UTM-Cluster-Anwendung besitzt jede Knoten-Anwendung ihr eigenes Dateigenerationsverzeichnis PROG, das Sie im Schritt 1. einrichten müssen.

Damit Sie die Schritte 2. und 3. nicht für jede Knoten-Anwendung explizit durchführen müssen, wird empfohlen, die PROG-Verzeichnisse so einzurichten, dass die Verzeichnisse aufeinander gelinkt sind (z.B. mit `ln -s <filebase1>/PROG <filebase2>/PROG`).
Damit ist sicher gestellt, dass sämtliche Knoten-Anwendungen immer auf die identischen Versionen des Anwendungsprogramms zugreifen.

9.1.2 Dateigenerationsverzeichnis PROG

openUTM verwaltet die verschiedenen Versionen Ihres Anwendungsprogramms, die Sie für den Austausch verwenden, in dem Dateigenerationsverzeichnis (FGG) PROG. Die FGG richten Sie wie folgt ein (siehe dazu auch Abschnitt „Das Tool KDCPROG“):

Auf Unix- und Linux-Systemen mit dem Kommando:
utmpfad `/ex/kdcprog CREATE` *operanden*

Auf Windows-Systemen in einem Eingabeaufforderung-Fenster mit dem Kommando:*utmpfad* `\ex\kdcprog CREATE` *operanden*

In der folgenden Tabelle sind den zentralen FGG-Begriffen jeweils die openUTM-spezifischen Definitionen bezüglich Versionsverwaltung und Austausch des Anwendungsprogramms gegenübergestellt.

FGG-Begriffe	**Versionsverwaltung für den Anwendungsaustausch durch openUTM**
Generation des FGG	Eine Version Ihres Anwendungsprogramms besteht aus *utmwork* . Die Datei *utmwork* bildet eine *Generation* der FGG.
absolute Generationsnummer	Beim Transfer einer Generation in die FGG ordnen Sie der Generation eine laufende Nummer zu. Dies ist die absolute Generationsnummer der Generation. openUTM spricht die einzelnen Generationen über ihre Generationsnummern an.
Name der Generation	Der Name einer Generation ist: • auf Unix- und Linux-Systemen: PROG/ *absolute_generationsnummer* • auf Windows-Systemen: PROG\ *absolute_generationsnummer* Dabei ist: 0001 <= *absolute_generationsnummer* <= 9999. Dem ersten Anwendungsprogramm, das Sie in die FGG transferieren, müssen Sie die absolute Generationsnummer 0001 zuordnen. Der Name der Generation ist damit PROG /0001 bzw. PROG\0001.
aufsteigende Generationsnummer	Die Generationsnummern der folgenden Generationen müssen fortlaufend aufsteigend sein (z.B. 0002, 0003, 0004, usw.).
Basis der FGG	Eine Generation der FGG ist die Basis der FGG. UTM lädt beim Start der Anwendung immer die Basis.
Basisnummer	Die absolute Generationsnummer der Basis ist die Basisnummer der FGG. Nach dem Einrichten der FGG ist 0001 die Basisnummer und die erste in die FGG transferierte Generation die Basis der FGG. Während des Anwendungslaufs schaltet openUTM die Basis immer auf die aktuell geladene Generation um, d.h. bei einem Programmaustausch schaltet openUTM die Basis auf die beim Austausch geladene Generation um.
relative Generationsnummer	Jeder Generation einer FGG wird relativ zur Basis eine relative Generationsnummer und ein relativer FGG-Name zugeordnet.

relativer FGG-Name	Der relative FGG-Name einer Generation ist PROG(*relative_generationsnr.*) wobei: *relative_generationsnr.* = *absolute_generationsnr.* - *Basisnummer.* Die relative Generationsnummer wird immer mit Vorzeichen (+ / -) angegeben.
maximale Anzahl der Generationen	Beim Einrichten der FGG legen Sie die maximale Anzahl der Generationen fest, die gleichzeitig in der FGG existieren dürfen.
erste Generation	Ist die maximale Anzahl der Generationen erreicht, dann geht openUTM beim nächsten Transfer wie folgt vor: • openUTM erzeugt eine neue Generation mit einer neuen Generationsnummer • openUTM löscht die Generation der FGG mit der niedrigsten Generationsnummer, d. h. die erste Generation.
letzte Generation	Die Generation der FGG mit der höchsten Generationsnummmer ist die letzte Generation der FGG.

Beispiele

1. Beim Start der Anwendung ist die Generation mit der absoluten Generationsnummer 0001 Basis der FGG. Während des Betriebs der Anwendung wird das Anwendungsprogramm mit KDCAPPL PROG=NEW ausgetauscht, die Generation mit der Generationsnummer 0002 wird geladen. Sie ist dann automatisch Basis der FGG. Beim nächsten Start der Anwendung lädt openUTM dann diese Generation (0002). Zwischen zwei Anwendungsläufen können Sie die Basis mit Hilfe der Funktion SWITCH des Tools KDCPROG umschalten:
   - Die Basis hat die relative Generationsnummmer +0000.
   - Die Generation, auf die beim Anwendungsaustausch mit KDCAPPL PROG=NEW geschaltet wird, hat die relative Generationsnummmer +0001.
   - Die Generation, auf die mit KDCAPPL PROG=OLD geschaltet wird, hat die relative Generationsnummmer -0001.
2. Die Basisnummer der FGG ist 6. Dann hat die Generation mit der absoluten Generationsnummer 0001 den relativen FGG-Namen PROG(-5), die Generation mit der absoluten Generationsnummer 0008 den relativen FGG-Namen PROG(+2).
3. Die maximale Anzahl der Generationen in der FGG ist 3. In der FGG existieren im Verzeichnis PROG die Generationen 0001, 0002 und 0003:
   - 0001 ist dabei die erste Generation und 0003 die letzte Generation.
   - Beim Transfer einer neuen Generation mit der Generationsnummer 0004 in die FGG löscht openUTM die erste Generation 0001. Die FGG enthält damit die drei Generationen 0002, 0003 und 0004.
   - Die erste Generation ist jetzt die mit der absoluten Generationsnummer 0002. Sie wird beim Transfer einer weiteren Generation in die FGG von openUTM gelöscht.

9.1.3 Ablauf des Anwendungsaustausches

Den Austausch des Anwendungsprogramms veranlasst der Administrator der UTM-Anwendung z.B. mit dem Kommando KDCAPPL PROG=... . Der Programmaustausch läuft dann entkoppelt ab.

Geben Sie KDCAPPL PROG=NEW an, dann wird das Anwendungsprogramm der Generation PROG(+1) geladen. Bei KDCAPPL PROG=OLD wird das Anwendungsprogramm der Generation PROG(-1) geladen. Mit KDCAPPL PROG=SAME wird das Anwendungsprogramm der aktuellen Dateigeneration neu geladen.

Aus diesem Grund ist es sinnvoll, die letzte Generation der FGG als Basis zu definieren. Diese Generation wird dann beim Start der Anwendung geladen. Eine neue Generation des Anwendungsprogramms transferieren Sie dann mit der relativen Generationsnummer PROG(+1) in die FGG (Standardeinstellung bei KDCPROG TRANSFER). Beim Programmaustausch mit KDCAPPL PROG=NEW wird dann die neue Generation des Anwendungsprogramms geladen. openUTM schaltet automatisch die Basis auf die aktuell geladene Generation, d. h. die letzte Generation, um. Wollen Sie dann auf das zuvor geladene Anwendungsprogramm zurückschalten, geben Sie KDCAPPL PROG=OLD an.

Der Austausch wird für jeden Workprozess der Anwendung nacheinander ausgeführt. Für jeden einzelnen Workprozess wird dazu nach Ausführung des aktuellen Auftrags das laufende Anwendungsprogramm beendet und das neue Anwendungsprogramm geladen. Erst wenn der Austausch für diesen Workprozess beendet ist, wird der Austausch für den nächsten Workprozess durchgeführt. Dadurch wird vermieden, dass der Anwendungslauf durch den Austausch nennenswert gestört wird. Der Benutzer bemerkt auf diese Weise von dem Anwendungsaustausch nichts. Er kann ungehindert weiterarbeiten.

Während des Anwendungsaustausches kann es durch den oben beschriebenen Ablauf vorkommen, dass Aufträge zur gleichen Zeit von einzelnen Workprozessen noch mit dem alten Anwendungsprogramm, von anderen Workprozessen bereits mit dem neuen Anwendungsprogramm bearbeitet werden. Das können Sie verhindern, wenn Sie vor dem Austausch die maximal zulässige Anzahl der Workprozesse auf 1 herabsetzen (z.B. mit dem Administrationskommando KDCAPPL TASKS=1).

Der UTM-Administrator wird nach Beendigung des Anwendungsaustausches durch eine Meldung informiert, ob der Austausch erfolgreich durchgeführt oder mit Fehler abgebrochen wurde. Erst wenn ein Austausch für alle Workprozesse beendet ist, kann der Administrator den nächsten Anwendungsaustausch starten.

9.1.4 Das Tool KDCPROG

Das Tool KDCPROG wird wie folgt aufgerufen:

Auf Unix- und Linux-Systemen rufen Sie KDCPROG von der Shell-Ebene mit dem Kommando auf:
utmpfad`/ex/kdcprog` *funktion operanden*

Auf Windows-Systemen starten Sie die Eingabeaufforderung und geben folgendes Kommando ein:
utmpfad`\ex\kdcprog` *funktion operanden*

KDCPROG bietet die folgenden Funktionen:

Funktion	**Bedeutung**
CREATE	Dateigenerationsverzeichnis PROG für den Anwendungsaustausch einrichten
INFO	Informationen über den aktuellen Zustand der FGG ausgeben
TRANSFER	Neue Version des Anwendungsprogramms in eine Generation der FGG übertragen
SWITCH	Basisnummer der FGG umschalten

Die Beschreibung der Operanden finden Sie auf den folgenden Seiten.

9.1.4.1 CREATE - Dateigenerationsverzeichnis (FGG) einrichten

KDCPROG CREATE erzeugt eine FGG für den Anwendungsaustausch. Es wird ein Dateiverzeichnis PROG im Basisverzeichnis *filebase* der Anwendung angelegt. Ein bereits bestehendes Verzeichnis PROG wird von KDCPROG vorher vollständig gelöscht.

`KDCPROG CREATE` *filebase number_entries*

filebase

Name des Dateiverzeichnisses, das bei der KDCDEF-Generierung mit MAX...,KDCFILE=*filebase* festgelegt wurde.

number_entries

Maximale Anzahl Generationen, die gleichzeitig in der FGG PROG existieren können. Sobald *number_entries* in der FGG vorhanden sind, wird beim Transfer einer neuen Generation in die FGG die erste Generation der FGG gelöscht.

Minimalwert: 2
Maximalwert: 9999

9.1.4.2 INFO - Aktuellen Zustand der FGG abfragen

KDCPROG INFO zeigt den aktuellen Zustand der FGG an, wobei die folgenden Daten ausgegeben werden:

- aktuelle Anzahl der Einträge
- Basisnummer der FGG
- Dateigeneration mit der niedrigsten Generationsnummer (erste Generation)
- Dateigeneration mit der höchsten Generationsnummer (letzte Generation)
- Liste der in der FGG enthaltenen Dateigenerationen mit absoluten und relativen Namen
- Liste der vorhandenen Generationen im Verzeichnis PROG. Die Ausgabe entspricht auf Unix- und Linux-Systemen der Ausgabe des Kommandos `ls -l` für das Dateiverzeichnis PROG.
  Auf Windows-Systemen wird ein allgemeinerer Hinweis auf das Kommando `dir` ausgegeben.

Beispiele für die Ausgabe von KDCPROG INFO finden Sie in Abschnitt „Beispiel für Anwendungsaustausch“.

`KDCPROG INFO` *filebase*

filebase

> Name des Dateiverzeichnisses, das bei der KDCDEF-Generierung mit MAX...,KDCFILE=*filebase* festgelegt wurde.

9.1.4.3 TRANSFER - utmwork in die FGG übertragen

KDCPROG TRANSFER überträgt *utmwork* aus *filebase* in die FGG.

`KDCPROG TRANSFER` *filebase generationnumber*

filebase

Name des Dateiverzeichnisses, das bei der KDCDEF-Generierung mit MAX...,KDCFILE=*filebase* festgelegt wurde.

generationnumber

Nummer der Generation, in welche *utmwork* übertragen werden sollen.

Die Angabe von *generationnumber* ist beim Transfer der ersten Version des Anwendungsprogramms Pflicht. Für *generationnumber* müssen Sie 0001 (absolut) oder +0 (relativ) angeben. Bei folgenden Transfers ist die Angabe optional. Geben Sie *generationnumber* nicht an, nimmt openUTM den Wert +1 an.

Wird für *generationnumber* eine Generation angegeben, die bereits in der FGG vorhanden ist, so wird diese Generation überschrieben.

Die Angabe von *generationnumber* ist auf zwei Arten möglich:

1. Angabe einer absoluten Generationsnummer.

   Maximal- und Minimalwert von *generationnumber* hängen von der Anzahl der FGG-Einträge ab.

   Die erste Generation, die Sie in die FGG transferieren, muss immer die absolute Generationsnummer 0001 haben. Neue Generationsnummern, die Sie danach vergeben, müssen fortlaufend ansteigend sein(0002, 0003...). Sie können auch Generationsnummern von Generationen angeben, die bereits in der FGG vorhanden sind. Diese werden dann überschrieben.

   KDCPROG gibt folgende absolute Grenzwerte vor:

   Minimalwert: 1
   Maximalwert: 9999

   Hinweis

   Die Angabe (aktuelle Basisnummer - 1) für *generationnumber* gibt die Dateigeneration an, auf die beim Anwendungsaustausch mit KDCAPPL PROG=OLD geschaltet wird.

   Die Angabe (aktuelle Basisnummer + 1) für *generationnumber* gibt die Dateigeneration an, auf die beim Anwendungsaustausch mit KDCAPPL PROG=NEW geschaltet wird.

2. Angabe einer relativen Generationsnummer mit führendem Vorzeichen (+ oder -). Maximal- und Minimalwert von *generationnumber* hängen vom aktuellen Basiswert und der Anzahl der FGG-Einträge ab. Ist beispielsweise die letzte Generation (höchste absolute Generationsnummer) Basis, dann dürfen Sie keine relative Generationsnummer angeben, die größer als +1 ist.
   KDCPROG gibt folgende absolute Grenzwerte vor:
   Minimalwert: - 99
   Maximalwert: + 99
   Hinweis
   Die Angabe -1 für *generationnumber* gibt die Dateigeneration an, auf die beim Anwendungsaustausch mit KDCAPPL PROG=OLD geschaltet wird.
   Die Angabe +1 für *generationnumber* gibt die Dateigeneration an, auf die beim Anwendungsaustausch mit KDCAPPL PROG=NEW geschaltet wird.
   Standardwert: +1 (relative Generationsnummer)
   Mit dem Standardwert wird die neue Version des Anwendungsprogramms in den FGG-Eintrag transferiert, auf die mit KDCAPPL PROG=NEW geschaltet wird.

9.1.4.4 SWITCH - Basis der FGG umschalten

Mit KDCPROG SWITCH können Sie außerhalb des Anwendungslaufs die Basis der FGG umschalten. Beim nächsten Start der Anwendung wird dann die neue Basis der FGG geladen. D.h. mit KDCPROG SWITCH können Sie zwischen zwei Anwendungsläufen die Funktionen von KDCAPPL PROG=NEW bzw. PROG=OLD ausführen.

Der Aufruf von KDCPROG SWITCH während des Anwendungslaufs wird abgelehnt.

`KDCPROG SWITCH` *filebase basenumber*

filebase

Name des Dateiverzeichnisses, das bei der KDCDEF-Generierung mit MAX...,KDCFILE=*filebase* festgelegt wurde.

basenumber

Angabe der neuen Basisgeneration. Für *basenumber* dürfen Sie nur eine Generationsnummer angeben, zu der schon eine Generation in der FGG existiert.

Die Angabe kann auf zwei Arten erfolgen:

1. Angabe einer absoluten Generationsnummer

   basenumber bezeichnet die neue Basisgeneration direkt.

   Minimalwert: 0
   Maximalwert: 9999

2. Angabe einer relativen Generationsnummer mit negativem VorzeichenDie Basisgeneration wird relativ zur letzten Generation (= Generation mit der höchsten Generationsnummer) angegeben. Der Wert von *basenumber* muss in diesem Fall immer mit führendem Minuszeichen (-) angegeben werden.

   Generationsnummer der Basis = Nummer der letzten Generation - *basenumber*

   Minimalwert: -99
   Maximalwert: -1

Beispiel

Die Generation mit der Geneationsnummer 0010 ist die letzte Generation in der FGG.`KDCPROG SWITCH` *filebase* `-1` bewirkt, dass die Dateigeneration mit der Generationsnummer 0009 neue Basis der FGG ist.
`KDCPROG SWITCH` *filebase* `0` bewirkt, dass die letzte Generation (0010) neue Basis der FGG ist.

9.1.5 Beispiel für Anwendungsaustausch

In den folgenden Abschnitten wird ein Beispiel-Anwendungsaustausch mit dem Tool KDCPROG durchgeführt.

1. Arbeitsschritt

Es wird eine FGG zum Anwendungsaustausch erzeugt, die maximal drei Generationen des Anwendungsprogramms enthalten darf. Danach wird die erste Generation des Anwendungsprogramms in die FGG transferiert. Diese Generation ist dann die Basis der FGG. Für *filebase* wird das aktuelle Dateiverzeichnis („.“) angegeben.

Auf Unix- und Linux-Systemen:	
Eingabe:	`utmpfad/ex/kdcprog CREATE . 3`
Ausgabe:	`U181 Programm kdcprog V07.0A00 auf <System + Bit-Modus> gestartet (pid: 1234, ...)` `U376 kdcprog: FGG-Dateien fuer ./PROG erzeugt.`
Eingabe:	`utmpfad/ex/kdcprog TRANSFER . +1` `utmpfad/ex/kdcprog SWITCH . 1`

Ausgabe:	`U181 Programm kdcprog V07.0A00 auf <System + Bit-Modus> gestartet (pid: 1234, ...)`
	`U383 kdcprog: TRANSFER : /bin/cp ./utmwork ./PROG/0001`
	`U389 kdcprog: TRANSFER erfolgreich`
	`U388 kdcprog: neue Basis der Programm FGG ./PROG ist 1`

Auf Windows-Systemen:	
Eingabe:	`utmpfad\ex\kdcprog CREATE . 3`
Ausgabe:	`U181 Programm kdcprog V07.0A00 auf <System + Bit-Modus> gestartet (pid: 1234, ...)` U376 kdcprog: FGG-Dateien fuer ./PROG erzeugt (`pid:`1234,...)
Eingabe:	`utmpfad\ex\kdcprog TRANSFER .`

Ausgabe:	U181 Programm kdcprog V07.0A00 auf <System + Bit-Modus> gestartet (`pid:`1234,...)
	`U391 kdcprog: TRANSFER fuer KDCAPPL PROG=NEW angestossen`
	`U383 kdcprog: TRANSFER : UTMCMD COPY ./utmwork.exe ./PROG/0001`
	`1 Datei(en) kopiert.`
	`U389 kdcprog: TRANSFER erfolgreich`

Eingabe:	`utmpfad\ex\kdcprog SWITCH . 1`
Ausgabe:	`U181 Programm kdcprog V07.0A00 auf <System + Bit-Modus> gestartet (pid: 1234,...)` `U388 kdcprog: neue Basis der Programm FGG ./PROG ist 1`

2. Arbeitsschritt

Während des Anwendungslaufs wird eine neue Version des Anwendungsprogramms erstellt. Diese Version soll als nächste Generation (Generationsnummer 0002) in die FGG transferiert werden. Die relative Generationsnummer dieser Generation ist dann (+1). Dies ist die Standardeinstellung beim TRANSFER.

Auf Unix- und Linux-Systemen:	
Eingabe:	`utmpfad/ex/kdcprog TRANSFER .`

Ausgabe:	`U181 Programm kdcprog V07.0A00 auf <System + Bit-Modus> gestartet (pid: 1234, ...)`
	`U383 kdcprog: TRANSFER : /bin/cp ./utmwork ./PROG/0002`
	`U389 kdcprog: TRANSFER erfolgreich`

Auf Windows-Systemen:	
Eingabe:	`utmpfad\ex\kdcprog TRANSFER .`

Ausgabe:	`U181 Programm kdcprog V07.0A00 auf <System + Bit-Modus> gestartet (pid: 1234, ...)`
	`U391 kdcprog: TRANSFER fuer KDCAPPL PROG=NEW angestossen`
	`U383 kdcprog: TRANSFER : UTMCMD COPY ./utmwork.exe ./PROG/0002`
	`1 Datei(en) kopiert.`
	`U389 kdcprog: TRANSFER erfolgreich`

Die transferierte Generation wird bei KDCAPPL PROG=NEW verwendet.

3. Arbeitsschritt

Es wird Information über die FGG angefordert.

Auf Unix- und Linux-Systemen:	
Eingabe:	`utmpfad/ex/kdcprog INFO .`

Ausgabe:	`U181 Programm kdcprog V07.0A00 auf <System + Bit-Modus> gestartet (pid: 1234, ...)`
	`U378 INFO fuer FGG ./PROG`
	`FGG Maximal Anzahl Versionen 3`
	`FGG Basis 0001`
	`FGG erste Generation 0001`
	`FGG letzte Generation 0002`
	`Datei PROG/0001 ist PROG(+0000) <=`
	`Datei PROG/0002 ist PROG(+0001)`
	`Die folgenden Programmdateien sind verfuegbar:`
	`-rwx------ 1 beispiel other 2845876 Apr 22 15:35 ./PROG/0001`
	`-rwx------ 1 beispiel other 2845876 Apr 22 15:37 ./PROG/0002`

Auf Windows-Systemen:	
Eingabe:	`utmpfad\ex\kdcprog INFO .`

Ausgabe:	U181 Programm kdcprog V07.0A00 auf <System + Bit-Modus> gestartet (`pid:`1234, ...)
	`U378 INFO fuer FGG ./PROG`
	`FGG Maximal Anzahl Versionen 3`
	`FGG Basis 0001`
	`FGG erste Generation 0001`
	`FGG letzte Generation 0002`
	`Datei PROG/0001 ist PROG(+0000) <=`
	`Datei PROG/0002 ist PROG(+0001)`
	`Die folgenden Programmdateien sind verfuegbar:`
	`kdcprog: type "DIR .\PROG\*.EXE" to get full information`

Bei der Ausgabe ist zu beachten, dass `PROG/000`*x* den Namen der jeweiligen Generation angibt. `PROG(+000`*x*`)` ist der relative FGG-Name. Der Pfeil „<=“ zeigt auf die Basis der FGG; das ist die aktuell geladene Generation des Anwendungsprogramms.

4. Arbeitsschritt

Der UTM-Administrator führt einen Anwendungsaustausch durch. Es soll die Generation 0002 (alias PROG(+1)) geladen werden. Dazu meldet sich der Administrator bei der UTM-Anwendung an und gibt z.B. ein:

```
KDCAPPL PROG=NEW
```

Nach dem Austausch wird noch einmal Information über die FGG angefordert.

Auf Unix- und Linux-Systemen:	
Eingabe:	`utmpfad/ex/kdcprog INFO .`

Ausgabe:	`U181 Programm kdcprog V07.0A00 auf <System + Bit-Modus> gestartet (pid: 1234, ...)`
	`U378 INFO fuer FGG ./PROG`
	`FGG Maximal Anzahl Versionen 3`
	`FGG Basis 0002`
	`FGG erste Generation 0001`
	`FGG letzte Generation 0002`
	`Datei PROG/0001 ist PROG(-0001)`
	`Datei PROG/0002 ist PROG(+0000) <=`
	`Die folgenden Programmdateien sind verfuegbar:`
	`-rwx------ 1 beispiel other 2845876 Apr 22 15:35 ./PROG/0001`
	`-rwx------ 1 beispiel other 2845876 Apr 22 15:37 ./PROG/0002`

Auf Windows-Systemen:	
Eingabe:	`utmpfad\ex\kdcprog INFO .`

Ausgabe:	U181 Programm kdcprog V07.0A00 auf <System + Bit-Modus> gestartet (pid: 1234, ...) U378 INFO fuer FGG ./PROG FGG Maximal Anzahl Versionen 3 FGG Basis 0002 FGG erste Generation 0001 FGG letzte Generation 0002 Datei PROG/0001 ist PROG(-0001) Datei PROG/0002 ist PROG(+0000) <= Die folgenden Programmdateien sind verfuegbar: kdcprog: type "DIR .\PROG*.EXE" to get full information

Die Ausgabe zeigt, dass openUTM die Basis geändert hat. Basis ist nun die Generation mit der Generationsnummer 0002, die beim Anwendungsaustausch geladen wurde.
Die Generation 0001 wird verwendet, falls KDCAPPL PROG=OLD eingegeben wird. Für KDCAPPL PROG=NEW ist kein Programm vorhanden.

5. Arbeitsschritt

Es wird eine weitere Version des Anwendungsprogramms in die FGG transferiert. Damit steht für einen erneuten Anwendungsaustausch mit KDCAPPL PROG=NEW eine neue Version des Anwendungsprogramms zur Verfügung.

Nach dem Transfer wird erneut Information über die FGG angefordert.

Auf Unix- und Linux-Systemen:	
Eingabe:	utmpfad/ex/kdcprog TRANSFER .
Ausgabe:	U181 Programm kdcprog V07.0A00 auf <System + Bit-Modus> gestartet (pid: 1234, ...) U383 kdcprog: TRANSFER : /bin/cp ./utmwork ./PROG/0003 U389 kdcprog: TRANSFER erfolgreich
Eingabe	utmpfad/ex/kdcprog INFO .
Ausgabe	U181 Programm kdcprog V07.0A00 auf <System + Bit-Modus> gestartet (pid: 1234, ...) U378 INFO fuer FGG ./PROG FGG Maximal Anzahl Versionen 3 FGG Basis 0002 FGG erste Generation 0001 FGG letzte Generation 0003 Datei PROG/0001 ist PROG(-0001) Datei PROG/0002 ist PROG(+0000) <= Datei PROG/0003 ist PROG(+0001) Die folgenden Programmdateien sind verfuegbar: -rwx------ 1 beispiel other 2845876 Apr 22 15:35 ./PROG/0001 -rwx------ 1 beispiel other 2845876 Apr 22 15:37 ./PROG/0002 -rwx------ 1 beispiel other 2845876 Apr 22 15:43 ./PROG/0003

Auf Windows-Systemen:	
Eingabe:	`utmpfad\ex\kdcprog TRANSFER .`

Ausgabe:	`U181 Programm kdcprog V07.0A00 auf <System + Bit-Modus> gestartet (pid: 1234, ...)`
	
	`(pid: 261,...)`
	`U391 kdcprog: TRANSFER fuer KDCAPPL PROG=NEW angestossen`
	`U383 kdcprog: TRANSFER : UTMCMD COPY ./utmwork.exe ./PROG/0003`
	`1 Datei(en) kopiert.`
	`U389 kdcprog: TRANSFER erfolgreich`

Eingabe	`utmpfad\ex\kdcprog INFO .`

Ausgabe	`U181 Programm kdcprog V07.0A00 auf <System + Bit-Modus> gestartet (pid: 1234, ...)`
	`U378 INFO fuer FGG ./PROG`
	`FGG Maximal Anzahl Versionen 3`
	`FGG Basis 0002`
	`FGG erste Generation 0001`
	`FGG letzte Generation 0003`
	`Datei PROG/0001 ist PROG(-0001)`
	`Datei PROG/0002 ist PROG(+0000) <=`
	`Datei PROG/0003 ist PROG(+0001)`
	`Die folgenden Programmdateien sind verfuegbar:`
	`kdcprog: type "DIR .\PROG\*.EXE" to get full information`

Jetzt steht je ein Programm für einen Anwendungsaustausch mit KDCAPPL PROG=OLD und mit KDCAPPL PROG=NEW zur Verfügung.

6. Arbeitsschritt

Es wird eine weitere Version des Anwendungsprogramms in die FGG transferiert und KDCPROG INFO aufgerufen.

Auf Unix- und Linux-Systemen:	
Eingabe:	`utmpfad/ex/kdcprog TRANSFER .`

Ausgabe:	`U181 Programm kdcprog V07.0A00 auf <System + Bit-Modus> gestartet (pid: 1234, ...)`
	`U383 kdcprog: TRANSFER : /bin/cp ./utmwork ./PROG/0004`
	`U389 kdcprog: TRANSFER erfolgreich`

Eingabe	`utmpfad/ex/kdcprog INFO .`

Ausgabe	U181 Programm kdcprog V07.0A00 auf <System + Bit-Modus> gestartet (pid: 1234, ...) U378 INFO fuer FGG ./PROG FGG Maximal Anzahl Versionen 3 FGG Basis 0002 FGG erste Generation 0002 FGG letzte Generation 0004 Datei PROG/0002 ist PROG(+0000) <= Datei PROG/0003 ist PROG(+0001) Datei PROG/0004 ist PROG(+0002) Die folgenden Programmdateien sind verfuegbar: -rwx------ 1 beispiel other 2845876 Apr 22 15:37 ./PROG/0002 -rwx------ 1 beispiel other 2845876 Apr 22 15:43 ./PROG/0003 -rwx------ 1 beispiel other 2845876 Apr 22 15:59 ./PROG/0004

Auf Windows-Systemen:	
Eingabe:	utmpfad\ex\kdcprog TRANSFER .
Ausgabe:	U181 Programm kdcprog V07.0A00 auf <System + Bit-Modus> gestartet (pid:1234,...) U391 kdcprog: TRANSFER fuer KDCAPPL PROG=NEW angestossen U383 kdcprog: TRANSFER : UTMCMD COPY ./utmwork.exe ./PROG/0004 1 Datei(en) kopiert. U389 kdcprog: TRANSFER erfolgreich
Eingabe	utmpfad\ex\kdcprog INFO .
Ausgabe	U181 Programm kdcprog V07.0A00 auf <System + Bit-Modus> gestartet (pid: 1234, ...) U378 INFO fuer FGG ./PROG FGG Maximal Anzahl Versionen 3 FGG Basis 0002 FGG erste Generation 0002 FGG letzte Generation 0004 Datei PROG/0002 ist PROG(+0000) <= Datei PROG/0003 ist PROG(+0001) Datei PROG/0004 ist PROG(+0002) Die folgenden Programmdateien sind verfuegbar: kdcprog: type "DIR .\PROG*.EXE" to get full information

Die Generation 0001 wurde gelöscht, da maximal drei Generationen in der FGG enthalten sein dürfen.

9.2 Shared Objects austauschen

Mit der Funktion „Shared Objects austauschen“ können Sie während des laufenden Betriebs einzelne Teile des Anwendungsprogramms austauschen. Diese Anwendungsteile müssen als Shared Objects erzeugt und dynamisch zur Anwendung dazugebunden werden. Dabei müssen Sie bestimmte Schritte beim Übersetzen, Binden und Generieren vornehmen.

Auf Windows-Systemen werden Shared Objects mit Hilfe von DLLs realisiert. Einzelheiten dazu finden Sie im Abschnitt „Anwendungsprogramme als DLLs erstellen (Windows-Systeme)“.

Shared Objects können Sie mit dem Administrationskommando KDCPROG austauschen oder mit einem eigenen Administrationsprogramm, das KDCADMI mit Operationscode KC_MODIFY_OBJECT und Objekttyp KC_LOAD_MODULE aufruft.
Beide Möglichkeiten sind im openUTM-Handbuch „Anwendungen administrieren“ beschrieben.

Zusätzlich können Sie Shared Objects mit den Administrationstools WinAdmin/WebAdmin austauschen.

Wenn im Folgenden auf Aktionen hingewiesen wird, die mit dem Kommando KDCPROG durchgeführt werden können, dann gilt dies auch für Administrationsprogramme, die KDCADMI mit Operationscode KC_MODIFY_OBJECT und Objekttyp KC_LOAD_MODULE absetzen.

Eine Anwendung mit Shared Objects kann auch als Ganzes ausgetauscht werden.

9.2.1 Shared Objects bereitstellen und generieren

Ein Shared Object in C müssen Sie auf Unix- und Linux-Systemen immer so übersetzen, dass dabei das jeweilige Laufzeitsystem mit eingebunden wird.

Näheres über das Übersetzen auf Windows-Systemen siehe "Anwendung übersetzen und binden (Windows-Systeme)".

Versionskonzept von Shared Objects

Shared Objects können mit oder ohne Versionen erstellt werden.

- Ohne Versionen

  Wenn Sie ein Shared Object ohne Version bereitstellen wollen, dann müssen Sie genau eine Datei mit dem Shared Object bereitstellen. Beim Austauschen mit dem Administrationskommando KDCPROG reicht die Angabe des Dateinamens. Shared Objects ohne Version können nur beim Start der Anwendung nachgeladen werden.

- Mit Versionen

  Soll ein Shared Object in mehreren Versionen verfügbar sein, dann müssen Sie zuerst ein Verzeichnis anlegen und anschließend die einzelnen Versionen des Shared Object in dieses Verzeichnis kopieren. Sie können beliebig viele Versionen einbringen. Beim Austauschen geben Sie sowohl den Verzeichnisnamen des Shared Objects als auch den Versionsnamen an.

  Auf Windows-Systemen sollten Sie Shared Objects immer **mit** Versionen erstellen.

Shared Objects generieren

Jedes Shared Object muss mit der KDCDEF-Anweisung SHARED-OBJECT generiert werden (siehe openUTM-Handbuch „Anwendungen generieren“) . Dabei geben Sie Folgendes an:

- Den Namen, den das Shared Object besitzt. Ist es ein Shared Object ohne Versionen, dann geben Sie den Dateinamen an, unter dem es abgespeichert ist. Ist es ein Shared Object mit Versionen, dann geben Sie den Namen des Verzeichnisses an, in dem die Versionen stehen.

  Auf Windows-Systemen muss der Name des Shared Object die Erweiterung `.dll` haben.

  Pro Shared Object kann nur eine Version generiert werden. Die Version kann per UTM- Administration geändert werden.
- Den Dateinamen der jeweiligen Version, falls es ein Shared Object mit Version ist (Operand VERSION).

  Auf Windows-Systemen muss die Version angegeben werden.
- Den Pfadnamen, unter dem das Shared Object zu finden ist (Operand DIRECTORY).

  Auf Windows-Systemen sollten Sie immer den kompletten Pfad angeben, da die Umgebungsvariablen PATH und LD_LIBRARY_PATH für Shared Objects nicht ausgewertet werden.
- Ob das Shared Object beim Start der Anwendung (LOAD-MODE=STARTUP) oder beim ersten Aufruf (LOAD-MODE=ONCALL) geladen werden soll.

In der zu dem Teilprogramm gehörenden PROGRAM-Anweisung müssen Sie den Namen des Shared Object angeben (Operand SHARED-OBJECT, siehe auch die Beispiele auf "Beispiele für Austausch von Shared Objects").

9.2.2 Start der Anwendung

Beim Start der Anwendung lädt openUTM alle Shared Objects, die mit LOAD=STARTUP generiert sind, und zwar in der Reihenfolge, in der die SHARED-OBJECT-Anweisungen gegeben wurden. Shared Objects mit LOAD-MODE=ONCALL werden erst beim erstmaligen Aufruf geladen.

Kann ein Shared Object nicht geladen werden, dann wird der Start dennoch fortgesetzt. Wird ein solches Shared Object später aufgerufen, dann führt dies zu einem BADTAC oder einem PEND ER.

Können die Event-Exits START, SHUT oder INPUT bzw. die Event-Services MSGTAC oder SIGNON oder das Administrationsteilprogramm KDCADM nicht geladen werden, dann wird der Start mit einer Fehlermeldung abgebrochen.

9.2.3 Ablauf des Austausches

Den Austausch eines Shared Objects muss der openUTM-Administrator veranlassen, z.B. mit dem Administrationskommando KDCPROG. Dabei werden die Event-Exits START und SHUT nicht durchlaufen, es sei denn, durch den Austausch wird das Anwendungsprogramm beendet und neu geladen.

Wenn Sie ein Shared Object austauschen, das mit Versionen generiert wurde, dann müssen Sie den Verzeichnisnamen und den Versionsnamen angeben. Bei Shared Objects ohne Versionen geben Sie den Namen des Shared Objects selber an, eine Versionsangabe wird ignoriert.

Der Austausch verläuft unterschiedlich, je nachdem, zu welchem Zeitpunkt (STARTUP oder ONCALL) das Shared Object geladen wurde.

9.2.3.1 Shared Objects mit LOAD-MODE=STARTUP austauschen

Beim Austausch von Anwendungsteilen, die mit LOAD-MODE=STARTUP generiert wurden, läuft der Workprozess weiter. Das Shared Object wird entladen und die angegebene Version neu geladen.

Diesen Programmaustausch können mehrere Workprozesse einer Anwendung gleichzeitig ausführen. Während des Programmaustauschs sind in den Workprozessen der UTM-Anwendung unterschiedliche Stände des Anwendungsprogramms geladen. Jeder Workprozess der Anwendung führt den angeforderten Programmaustausch am Ende der Bearbeitung des aktuellen Auftrags durch. Das Ende des Programmaustauschs wird durch eine Meldung angezeigt.

Bis der Programmaustausch abgeschlossen ist, kann kein weiterer Programmaustausch gestartet werden. Ein erneuter Aufruf von KDCPROG wird von openUTM abgewiesen.

9.2.3.2 Shared Objects mit LOAD-MODE=ONCALL austauschen

Shared Objects, die mit LOAD-MODE=ONCALL generiert sind, können Sie nur austauschen, wenn diese **mit** Versionen erzeugt wurden.

Wenn Sie ein solches Shared Object austauschen, dann wird bei der Bearbeitung des Administrationskommandos KDCPROG nur der neu zu ladende Versionsbezeichner des betroffenen Shared Objects in die openUTM-Tabellen eingetragen.

Die neue Version wird von jedem Workprozess der Anwendung erst dann geladen, wenn dieser Workprozess das nächste Mal ein Teilprogramm aufruft, das in diesem Shared Object enthalten ist. Diesen Programmaustausch können mehrere Workprozesse einer Anwendung gleichzeitig ausführen. Bis der angeforderte Programmaustausch von allen Workprozessen der UTM-Anwendung durchgeführt wurde, sind in den einzelnen Workprozessen unterschiedliche Stände des Anwendungsprogramms geladen. Es ist jedoch sichergestellt, dass jeder Workprozess den angeforderten Austausch durchführt, bevor erneut ein Teilprogramm aktiviert wird, das in dem auszutauschenden Shared Object enthalten ist.

Der Austausch eines mit ONCALL generierten Shared Objects wirkt nicht blockierend auf nachfolgende Kommandos zum Programmaustausch. D.h. der Administrator kann unmittelbar nach der Bearbeitung des Kommandos KDCPROG mit einem weiteren Kommando KDCPROG einen neuen Programmaustausch veranlassen.

Stimmen die Versionsbezeichner von neuem und altem Shared Object überein, wird kein Programmaustausch durchgeführt.

9.2.4 Beispiele für Austausch von Shared Objects

Beispiel 1 (Unix- und Linux-Systeme)

Auf einem Unix- oder Linux-System soll ein einzelnes Modul mit Namen EKSTEUER dynamisch zu einem bestimmten Stichtag ausgetauscht werden. Dazu gehen Sie wie folgt vor:

1. Erstellen und übersetzen Sie das Modul mit den Optionen, die für Shared Objects notwendig sind.
2. Bringen Sie es als Shared Object ohne Version unter dem Namen EKSTEUER in das Verzeichnis, in dem sich die benutzereigenen Programme befinden. Im Beispiel wird für das Verzeichnis der Platzhalter *so-lib* angegeben, das kann z.B. ein Verzeichnis `/usr/proglib` sein.
3. Generieren Sie das Modul mit folgenden KDCDEF-Anweisungen als Shared Object:

   `SHARED-OBJECT EKSTEUER,DIRECTORY=`*so-lib*`,LOAD-MODE=STARTUP`

   `PROGRAM .... ,SHARED-OBJECT=EKSTEUER`

   Das Shared Object wird damit beim Anwendungsstart geladen (Pflicht bei Shared Objects ohne Version).
4. Binden Sie den Workprozess, wobei Sie die dynamische Bibliothek mit dem Shared Object EKSTEUER angeben müssen.
5. Starten Sie die Anwendung wie gewohnt.
6. Ändern Sie das Modul und speichern Sie es vor dem Stichtag in die Datei:

   so-lib/EKSTEUER
7. Geben Sie folgendes Administrationskommando ein:

   `KDCPROG SHARED-OBJECT=EKSTEUER`

   Das Shared Object wird in den einzelnen Workprozessen ausgetauscht, sobald diese den aktuellen Auftrag abgearbeitet haben.

Beispiel 2

Auf einem Unix- bzw. Windows-System soll ein Shared Object mit Namen MONATSABSCHLUSS in 12 Versionen (BIL01,... BIL12) vorhanden sein und zu jedem Monatswechsel ausgetauscht werden. Es soll immer erst beim erstmaligen Aufruf geladen werden. Dazu gehen Sie wie folgt vor:

1. Erstellen und übersetzen Sie das Modul mit den Optionen, die für Shared Objects notwendig sind (für Windows-Systeme siehe "Anwendungsprogramme als DLLs erstellen (Windows-Systeme)").
2. Richten Sie im Verzeichnis mit den benutzereigenen Programmen (im Beispiel *so-lib*) das Verzeichnis MONATSABSCHLUSS ein und kopieren Sie zumindest die beim ersten Anwendungslauf benötigte Version in dieses Verzeichnis.
3. Generieren Sie jede Version des Shared Object mit folgender KDCDEF-Anweisung:

   ```
   SHARED-OBJECT MONATSABSCHLUSS
                 ,DIRECTORY=so-lib
                 ,VERSION=BILxx       (xx=01,... ,12)
                 ,LOAD-MODE=ONCALL
   ```

   Das Shared Object wird damit erst beim Aufruf des Teilprogramms geladen.

   Geben Sie für das Shared Object folgende PROGRAM-Anweisung:

   `PROGRAM .... ,SHARED-OBJECT=MONATSABSCHLUSS`

4. Auf Unix- und Linux-Systemen binden Sie den Workprozess, indem Sie die dynamische Bibliothek mit den Shared Objects angeben.
   Auf Windows-Systemen gibt es an dieser Stelle keine Besonderheiten.
5. Starten Sie die Anwendung wie gewohnt.
6. Geben Sie z.B. am 1. Juli folgendes Administrationskommando ein:

   ```
   KDCPROG SHARED-OBJECT=MONATSABSCHLUSS,VERSION=BIL07
   ```

   Achten Sie bitte darauf, dass diese Version zum genannten Zeitpunkt auch im Verzeichnis vorhanden ist.

   Das Shared Object wird in den einzelnen Workprozessen erst ausgetauscht, wenn das entsprechende Teilprogramm erstmals aufgerufen wird.

9.2.5 Anwendung mit Shared Objects austauschen

Die gesamte Anwendung können Sie mit dem Administrationskommando KDCAPPL PROG=NEW austauschen. Der Austausch kann entweder über das Dateigenerationsverzeichnis PROG erfolgen, das Sie mit dem Tool KDCPROG vorbereiten müssen (siehe Abschnitt „Voraussetzungen für den Anwendungsaustausch“) oder UTM lädt das Anwendungsprogramm *utmwork* direkt aus dem Basisverzeichnis.

In beiden Fällen wird beim Austausch der gesamten Anwendung nacheinander jeder Workprozess der Anwendung entladen und anschließend neu geladen. Beim Neuladen werden die neuen Versionen der Shared Objects geladen. Um den Betrieb der Anwendung möglichst wenig zu stören, tauscht zu einer Zeit immer nur ein Workprozess der Anwendung.

9.2.6 Programme dynamisch hinzufügen

Per dynamischer Administration können u.a. Programme im Betrieb der Anwendung neu generiert werden. Näheres zur dynamischen Administration siehe openUTM-Handbuch „Anwendungen administrieren“.

Bevor diese Programme aufgerufen werden können, müssen sie zunächst geladen werden. Dazu muss das Programm zu dem zugeordneten Shared Object gebunden und mit einer neuen Version im Verzeichnis bereitgestellt werden, das in der SHARED-OBJECT-Anweisung beim Generieren angegeben wurde.

Anschließend muss der Administrator dieses Shared Object durch das Kommando KDCPROG oder durch Programmaufruf austauschen.

Der UTM-Administrator muss die neuen Teilprogramme und die zugehörigen Transaktionscodes dynamisch in die KDCFILE-Tabellen eintragen.

10 Fehlertoleranz von openUTM

Fehlertoleranz heißt hier, dass eine UTM-Anwendung auch dann betriebsbereit bleibt, wenn in einzelnen Teilprogrammen Fehler auftreten, die openUTM zum Abbruch einer Transaktion zwingen. openUTM sorgt dann dafür, dass das Anwendungsprogramm beendet und neu geladen wird, so dass sich der Fehler nicht weiter ausbreitet und andere Benutzer der Anwendung und deren Daten dadurch nicht beeinträchtigt werden.

Beim Fehlerverhalten von openUTM unterscheidet man:

- Interne UTM-Fehler und Fehler in der Systemumgebung
  Sie führen zum abnormalen Beenden der Anwendung, ebenso wie das Administrationskommando KDCSHUT KILL oder das Absetzen eines KDCADMI-Aufrufs mit Operationscode KC_SHUTDOWN und Subcode KC_KILL. openUTM erzeugt für jeden Prozess der Anwendung einen UTM-Dump. Der UTM-Dump wird mit dem Tool KDCDUMP aufbereitet. Wie das geht, ist im openUTM-Handbuch „Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen“ beschrieben.
  - Bei schweren Fehlern im Dialog-Terminalprozess beendet sich dieser und schreibt unter dem aktuellen Dateiverzeichnis einen Core-Dump. Ein erneutes Anmelden während dieses Anmeldungslaufes ist vom zugeordneten Terminal aus nicht mehr möglich. Bei leichten Fehlern meldet sich der Dialog-Terminalprozess ordnungsgemäß von der Anwendung ab.
  - Ein Druckerprozess verhält sich bei Fehlern ähnlich wie ein Dialog-Terminalprozess. Er kann ggf. über Administrationskommando neu gestartet werden.
  - Bei Fehlern im Timerprozess wird die Anwendung abnormal beendet, sobald von den Workprozessen ein Auftrag an den Timerprozess ergeht.

- Fehler im Anwendungsprogramm
  Dabei handelt es sich um Fehler in Teilprogrammen, die sich in zwei Gruppen einteilen lassen:
  - Fehler, die zum Neuladen des Anwendungsprogramms führen
  - Fehler, die gegebenenfalls eine Fortsetzung des Programms erlauben

10.1 Fehler, die openUTM erkennt

Ein Teilprogramm wird in folgenden Fällen durch openUTM abnormal beendet:

- Es wurde ein PEND ER oder FR programmiert.
- Ein UTM-Aufruf hat einen KDCS-Returncode >= 70Z geliefert. In diesem Fall setzt openUTM intern PEND ER.

In allen Fällen bricht openUTM also den Vorgang ab. Falls PEND FR programmiert war, veranlasst openUTM keine weitere Aktionen.

Falls der Vorgang durch PEND ER (per Programm oder intern) beendet wurde, erzeugt openUTM einen UTM-Dump mit REASON=PENDER, der nur die Daten des KDCROOT wiedergibt. Anschließend wird der betroffene Workprozess beendet. Der Mainprozess startet danach einen neuen Workprozess, der das Anwendungsprogramm neu lädt. Dadurch wird auf einem neuen Stand der statischen Datenbereiche aufgesetzt und es werden Folgefehler durch überschriebene Daten vermieden.

10.2 Signalbehandlung von openUTM

Beim Auftreten eines Signals sind die folgenden Aktionen möglich:

- Ignorieren dieses Signals (siehe Tabelle).
- Wenn das Signal während des Ablaufs von Anwender-Programmteilen auftritt: Es wird ein PEND ER mit KCRCCC=70Z und KCRCDC=XT*xx* aufgerufen (*xx* ist die Signalnummer).
  Der betreffende Vorgang bzw. Workprozess wird beendet.

  Ausnahme: Ist auf Unix- und Linux-Systemen eine registrierte User Signal Routine vorhanden, dann wird stattdessen diese Routine aufgerufen. Details siehe openUTM-Handbuch „Anwendungen programmieren mit KDCS“.
- Wenn das Signal während des Ablaufs von UTM-Systemteilen auftritt: Es führt zum abnormalen Anwendungsende mit REASON=SIG*xxx*
  (*xxx* ist die Signalnummer).

Die folgende Tabelle zeigt die Reaktion eines Workprozesses auf die einzelnen Signale. Detaillierte Informationen zu den Signalen sind dem C-Headerfile für Signale (`signal.h`) zu entnehmen.

Empfangenes Signal	**Reaktion**
SIGHUP	wird ignoriert
SIGINT	wird ignoriert
SIGQUIT	wird ignoriert
SIGILL	PEND ER/TRMA[1]
SIGTRAP	wird ignoriert
SIGABRT	wird ignoriert
SIGEMT	wird ignoriert
SIGFPE	PEND ER/TRMA[1]
SIGBUS	PEND ER/TRMA[1]
SIGSEGV	PEND ER/TRMA[1]
SIGSYS	PEND ER/TRMA[1]
SIGPIPE	wird ignoriert
SIGALRM	PEND ER[2]/ wird ignoriert
SIGTERM	wird ignoriert
SIGUSR1	wird ignoriert
SIGUSR2	wird ignoriert

SIGCHLD	wird ignoriert
SIGPWR	PEND ER/TRMA[1]
SIGWINCH	wird ignoriert
SIGURG	wird ignoriert
SIGIO	wird ignoriert
SIGTSTP	wird ignoriert
SIGCONT	wird ignoriert
SIGTTIN	wird ignoriert
SIGTTOU	wird ignoriert
SIGVTALRM	wird ignoriert
SIGPROF	wird ignoriert
SIGXCPU	PEND ER/TRMA[1]
SIGXFSZ	PEND ER/TRMA[1]

[1]TRMA steht für Term Application (= Anwendungsabbruch)

[2]PEND ER wenn Timer von UTM aufgezogen (KDCDEF Generierung TAC RTIME=<rtime>)

Tritt während der durch ein Signal hervorgerufenen Endebehandlung eine Unterbrechung durch ein weiteres Signal auf, dann wird dieses Signal nicht von openUTM abgefangen. In diesem Fall übernimmt das Betriebssystem die Behandlung der Signalunterbrechung.

> **!** **Signale in User Signal Routine**
>
> Tritt beim Ablauf der User Signal Routine ein weiteres Signal auf, wird die Standardbehandlung des Betriebssystems für dieses Signal aktiviert. Dies führt in der Folge zu einer abnormalen Beendigung der UTM-Anwendung.

10.3 Anwendungsende durch System-Absturz/Shutdown

Ein System-Absturz bzw. ein Betriebssystem-Shutdown führt zu einem abnormalen Anwendungsende, bei dem jedoch kein UTM-Dump erzeugt wird. Sämtliche Prozesse der Anwendung werden vom Betriebssystem beendet. Bevor die Anwendung in solchen Fällen wieder gestartet wird, muss das Tool KDCREM aufgerufen werden. Sehen Sie dazu den Abschnitt „Das Tool KDCREM“.

11 Accounting

openUTM stellt Accounting-Funktionen zur Verfügung, die es dem Betreiber einer UTM-Anwendung ermöglichen, den Benutzern der UTM-Anwendung die in Anspruch genommenen Betriebsmittel zu verrechnen. Die Accounting-Funktionen, die das jeweilige Betriebssystem zur Verfügung stellt, können die Betriebsmittelauslastung und Leistung einer UTM-Anwendung nur als Ganzes erfassen. Wenn Sie die DV-Leistungen aber den einzelnen Benutzern der UTM-Anwendung zuordnen und in Rechnung stellen wollen, muss für das UTM-Accounting Folgendes berücksichtigt werden:

- Die Benutzer einer UTM-Anwendung werden durch Benutzerkennungen der UTM-Generierung repräsentiert und nicht durch Benutzerkennungen des Betriebssystems. Die von einem Benutzer in Anspruch genommenen Leistungen müssen also den einzelnen UTM-Benutzerkennungen zugeordnet werden können.
- In einer UTM-Anwendung ist eine Gruppe homogener Prozesse aktiv. Jeder Prozess bearbeitet nacheinander Aufträge für wechselnde Benutzer. Die innerhalb eines Prozesses in Anspruch genommenen Leistungen müssen deshalb pro aufgerufenen Service (d.h. für einzelne Teilprogrammläufe) ermittelt werden.
- Die Zeitbedingungen des OLTP-Betriebs erfordern eine Form der Leistungserfassung, die die Performance der Anwendung nicht beeinträchtigt.

Beim UTM-Accounting wird also der Verbrauch an Betriebsmitteln erfasst, der von den einzelnen Teilprogrammen beansprucht wird. Damit kann der Betriebsmittelverbrauch dem Transaktionscode (TAC) des jeweiligen Teilprogramms und somit auch dem UTM-Benutzer, der den zugehörigen Service gestartet hat, zugeordnet werden.

Über die vom UTM-Accounting erfassten Leistungen hinaus gibt es einen Grundbedarf an Betriebsmitteln, der beim Ablauf einer UTM-Anwendung anfällt, aber nicht direkt einem Benutzer zugeordnet werden kann. Das sind:

- Plattenbelegung für KDCFILE-, SYSLOG- und USLOG-Dateien sowie
- CPU-Verbrauch für
  - Starten und Beenden der UTM-Prozesse
  - Verbindungsbehandlung der zugeordneten Clients
  - LPUT-Behandlung (Übertragung auf USLOG-Datei)
  - Aufbereitung von Druckerausgaben (Unix- und Linux-Systeme)

Sollen diese Leistungen bei der Abrechnung berücksichtigt werden, dann müssen Sie sie den Benutzern als Pauschale in Rechnung stellen.

11.1 Begriffsdefinitionen

In diesem Abschnitt werden einige Begriffe genauer erläutert, die für das UTM-Accounting relevant sind.

Benutzer im Sinne des UTM-Accounting

Der Benutzer einer UTM-Anwendung, für den eine Abrechnung erstellt werden soll, wird i.A. durch die UTM-Benutzerkennung repräsentiert.

Bei Anwendungen oder Clients, die sich nicht explizit mit einer echten Benutzerkennung angemeldet haben, wird der Name der Verbindungs-Benutzerkennung (TS-Anwendungen und UPIC-Clients), der LU6-Sessionname (LU6-Partner) bzw. der OSI-Association-Name (OSI TP-Partner) verwendet.

In UTM-Anwendungen ohne Benutzerkennungen ordnet openUTM die Leistungen, die von Terminals, UPIC-Clients oder TS-Anwendungen in Anspruch genommen wurden, ersatzweise den LTERM-Partnern zu.

Abrechnungsdatei

Alle Informationen, die das UTM-Accounting für die Benutzer-spezifische Abrechnung der Leistungen sammelt, schreibt openUTM in die Abrechnungsdatei.

Die Abrechnungsdatei wird Anwendungs-spezifisch geführt und vom Administrator der UTM-Anwendung verwaltet, näheres siehe Abschnitt „Auswertung“.

Betriebsmittel

Darunter werden folgende Leistungen zusammengefasst:

- technische DV-Leistungen, insbesondere CPU-Verbrauch
- der Aufruf eines bestimmten Programms (Programmgebühr)

Kalkulationsphase

Die Kalkulationsphase dient als Orientierung für den Einsatz des Abrechnungsverfahrens.

In der Kalkulationsphase ermittelt openUTM für jedes aufgerufene Teilprogramm den Verbrauch der einzelnen Betriebsmittel und schreibt die Werte als Kalkulationssatz in die Abrechnungsdatei. Näheres ist in Abschnitt „Kalkulationsphase“ beschrieben.

Kalkulationssatz

Ein Kalkulationssatz ist ein Satz, den openUTM in der Kalkulationsphase pro Teilprogrammlauf in die Abrechnungsdatei schreibt. Er hat den Accounting-Satztyp UTMK. Die Datenfelder des Kalkulationssatzes UTMK sind im Anhang auf "Aufbau des Kalkulationssatzes" beschrieben.

Gewicht

Pro Betriebsmittel können Sie ein Gewicht (Faktor) festlegen. Dieses Gewicht gibt an, wie das Betriebsmittel im Vergleich mit anderen Betriebsmitteln zu werten ist. Der Verbrauch eines Betriebsmittels geht dann als Produkt „Gewicht * Betriebsmittelverbrauch“ in die Abrechnung ein. Die Gewichte für die einzelnen Betriebsmittel geben Sie bei der KDCDEF-Generierung in ACCOUNT an, siehe Abschnitt „Variante des Abrechnungsverfahrens festlegen“.

Abrechnungsphase

In der Abrechnungsphase ermittelt openUTM für jedes Teilprogramm den Betriebsmittelverbrauch. Nach Beendigung eines Teilprogrammlaufs ermittelt openUTM anhand der generierten Gewichte und des generierten Festpreises die Summe der Verbrauchswerte.

Folgenden Leistungen werden berücksichtigt:

- CPU-Verbrauch
- erzeugte Ausgabeaufträge für Drucker (Unix- und Linux-Systeme)
- Festpreis für den Aufruf eines Teilprogramms

Das Ergebnis ist eine Anzahl von Verrechnungseinheiten, die auf den Benutzerspezifischen Verrechnungseinheitenzähler aufaddiert wird.

openUTM schreibt erst dann einen Satz mit dem Inhalt dieses Zählers in die Abrechnungsdatei,

- wenn der Benutzer sich abmeldet und über keine andere Verbindung mehr bei der UTM-Anwendung angemeldet ist,
- oder wenn die Anwendung normal beendet wird
- oder wenn ein bestimmter Maximalwert überschritten wird. Diesen Maximalwert legen Sie bei der KDCDEF-Generierung mit ACCOUNT ...,MAXUNIT= fest.

Gewichte und Festpreise müssen Sie vor dem Start der Abrechnungsphase in die UTM-Generierung der Anwendung einbringen. Dabei können Sie wählen zwischen

- Festpreis-Abrechnung,
- verbrauchsorientierter Abrechnung,
- und einer Kombination der beiden Varianten.

Eine detaillierte Beschreibung der Abrechnungsphase finden Sie in Abschnitt „Abrechnungsphase“.

Die Abrechnungsphase des UTM-Accounting kann im laufenden Betrieb der UTM-Anwendung ein- und ausgeschaltet werden.

Abrechnungssatz

Ein Abrechnungssatz ist ein Satz, den openUTM in der Abrechnungsphase in die Abrechnungsdatei schreibt. Er hat den Accounting-Satztyp UTMA.
Die Datenfelder des Abrechnungssatz UTMA sind im Anhang auf ''Aufbau des Abrechnungssatzes'' beschrieben.

Verrechnungseinheiten

Verrechnungseinheiten sind das Produkt aus Verbrauch und Gewicht des jeweiligen Betriebsmittels. In der UTM-Abrechnung werden nur Verrechnungseinheiten gezählt.

Verrechnungseinheitenzähler

openUTM führt in einer UTM-Anwendung pro Benutzer einen Verrechnungseinheitenzähler und akkumuliert dort den Verbrauch der Verrechnungseinheiten pro Benutzer.

Festpreis-Abrechnung

Mit dieser Variante des Abrechnungsverfahrens wird für einen Teilprogrammlauf eine konstante Anzahl von Verrechnungseinheiten in Rechnung gestellt. Diese Anzahl wird bei der Anwendungsgenerierung dem Transaktionscode zugeordnet. Die Gewichte der anderen Betriebsmittel sind dabei Null. Dabei können Sie einzelne Services auch kostenlos anbieten, z.B. Auskunftsfunktionen.

Verbrauchsorientierte Abrechnung

Mit dieser Variante des Abrechnungsverfahrens wird für einen Teilprogrammlauf der Verbrauch an Betriebsmitteln in Rechnung gestellt, der aktuell ermittelt wird. Die Verbrauchswerte für die Betriebsmittel werden entsprechend der generierten Gewichte gewichtet. Für den Aufruf eines Teilprogramms wird kein Festpreis berechnet.

11.2 Phasen des Accounting

Für die Durchführung des Accounting in UTM-Anwendungen sind folgende Schritte erforderlich:

- Kalkulationsphase
- Abrechnungsverfahren festlegen
- Abrechnungsphase
- Auswertung

11.2.1 Kalkulationsphase

Die Kalkulationsphase liefert Orientierungswerte, mit deren Hilfe Sie die einzelnen Betriebsmittel gewichten und Festpreise für die Inanspruchnahme eines Services festlegen können. Dabei ermittelt openUTM für jeden Teilprogrammlauf den Betriebsmittelverbrauch, stellt am Programmlaufende einen Kalkulationssatz vom Satztyp UTMK zusammen und schreibt ihn in die Abrechnungsdatei.

Die Kalkulationsphase kann im laufenden Betrieb jederzeit durch die UTM-Administration ein- und ausgeschaltet werden, z.B. um die generierten Gewichte zu überprüfen und gegebenenfalls bei der Neugenerierung zu aktualisieren.

Sie sollten jedoch beachten, dass openUTM bei eingeschalteter Kalkulationsphase nach jedem Teilprogrammlauf einen Satz in die Abrechnungsdatei schreibt. Dadurch wird die Performance der Anwendung belastet.

Kalkulationsphase einschalten

Die Kalkulationsphase können Sie bei der KDCDEF-Generierung oder per Administration einschalten, siehe auch openUTM-Handbuch „Anwendungen generieren“ und openUTM-Handbuch „Anwendungen administrieren“:

- per KDCDEF-Anweisung ACCOUNT ACC=CALC
- oder per UTM-Administration
  - durch das Kommando KDCAPPL CALC=ON
  - oder über WinAdmin/WebAdmin
  - oder per KDCADMI-Programmaufruf KC_MODIFY_OBJECT mit obj_type=KC_DIAG_AND_ACCOUNT_PAR

Kalkulationsphase ausschalten

Die Kalkulationsphase können Sie nur per UTM-Administration ausschalten:

- durch das Kommando KDCAPPL CALC=OFF
- oder über WinAdmin/WebAdmin
- oder per KDCADMI-Programmaufruf KC_MODIFY_OBJECT mit obj_type=KC_DIAG_AND_ACCOUNT_PAR

Daten eines Kalkulationssatzes

Ein Kalkulationssatz enthält folgende Daten:

- Name der UTM-Anwendung
- Transaktionscode (TAC) des Teilprogramms
- CPU-Verbrauch im UTM-Prozess in msec
- Länge der Eingabe-Nachricht in Byte
- Länge der Ausgabe-Nachricht in Byte
- Anzahl Ausgabeaufträge an Drucker (Unix- und Linux-Systeme)
- Verrechnungseinheiten für LTAC-Aufrufe
- UTM-Benutzer, der den Service aufgerufen hat
- Name des LTERM-Partners, über den der Benutzer angemeldet ist
- Realzeit des Teilprogrammlaufs (msec)

Bei den Ausgabe-Nachrichten werden auch die mitgerechnet, die an ein Folgeteilprogramm gerichtet sind (z.B. nach PEND PR).

11.2.2 Variante des Abrechnungsverfahrens festlegen

Zunächst müssen Sie festlegen, ob Sie über Festpreis, über den Verbrauch oder über eine Kombination aus diesen beiden Varianten abrechnen. Ihre Entscheidung hängt davon ab, ob Sie bestimmte Leistungen der Anwendung mit festen Preisen anbieten oder den tatsächlichen Betriebsmittelverbrauch in Rechnung stellen wollen.

Festpreis-Abrechnung

Bei der Festpreis-Abrechnung kostet ein Teilprogrammlauf eine konstante Anzahl von Verrechnungseinheiten. Orientierungswerte dafür liefert die Kalkulationsphase. Daher ist eine Festpreis-Abrechnung die einfachste Lösung.

Die Anzahl der Verrechnungseinheiten geben Sie bei der KDCDEF-Generierung in der TAC-Anweisung beim Operanden TACUNIT an, siehe openUTM-Handbuch „Anwendungen generieren“.

`TAC` *tacname*`,PROGRAM=`*progname*`,TACUNIT=`*anzahl_verrechnungseinheiten*

Für jeden vom Benutzer aufgerufenen Transaktionscode wird der in TACUNIT angegebene Wert auf den Benutzer-spezifischen Verrechnungseinheitenzähler aufaddiert.

Bei der Festpreis-Abrechnung können Sie einige Services (z.B. Auskunftsfunktionen) auch kostenfrei zur Verfügung stellen. Dazu müssen Sie die zugehörigen Transaktionscodes wie folgt generieren:

```
TAC ... TACUNIT=0
```

Bei Verteilter Verarbeitung gilt Entsprechendes für die Anweisung LTAC und den Operanden LTACUNIT, siehe Abschnitt „Abrechnung bei verteilter Verarbeitung“.

Bei einer Festpreis-Abrechnung müssen Sie die Gewichte der Betriebsmittel in der KDCDEF-Anweisung ACCOUNT auf 0 setzen (=Standardwert).

Verbrauchsorientierte Abrechnung

Bei dieser Variante wird dem Benutzer der Verbrauch an Betriebsmitteln in Rechnung gestellt, der aktuell in der Abrechnungsphase ermittelt wird. Für die einzelnen Betriebsmittel müssen Sie Gewichte festlegen. Ein Gewicht ist ein Faktor, mit dem die verbrauchten Einheiten multipliziert werden. Bei der Wahl dieser Gewichte können Sie sich an den Verbrauchsdaten orientieren, die Sie in der Kalkulationsphase ermittelt haben.

Die Gewichte werden Anwendungs-spezifisch in der KDCDEF-Anweisung ACCOUNT festgelegt, d.h. sie gelten für alle Teilprogrammläufe.

Die Festlegung der Gewichte ist zwangsläufig subjektiv und hängt von der Installationsumgebung ab. Folgende Betriebsmittel können gewichtet werden:

- CPU-Verbrauch (ACCOUNT-Operand CPUUNIT)
- Ein-/Ausgaben auf Hintergrundspeicher (ACCOUNT-Operand IOUNIT)
- Druckerausgaben (ACCOUNT-Operand OUTUNIT) (Unix- und Linux-Systeme)

Näheres siehe openUTM-Handbuch „Anwendungen generieren“.

Beispiel für die KDCDEF-Generierung dieser Variante

```
ACCOUNT ACC=ON,CPUUNIT=15,IOUNIT=5,OUTUNIT=20

TAC tacname,PROGRAM=progname,TACUNIT=0

TAC ....
```

Pro Aufruf eines Transaktionscodes wird dann die folgende Summe auf den Verrechnungseinheitenzähler des Benutzers aufaddiert:

15 * CPU-Verbrauch + 5 * I/O-Verbrauch + 20 * Druckausgaben-Verbrauch

Kombination aus Festpreis- und verbrauchsorientierter Abrechnung

Sie können für Ihre Abrechnung die beiden obigen Varianten auch kombinieren, indem Sie für den Aufruf eines Transaktionscodes einen bestimmten Festpreis festlegen und zusätzlich den Verbrauch der Betriebsmittel (z.B. den CPU-Verbrauch) verrechnen.

In der Abrechnungsphase wird beim Aufruf eines Transaktionscodes die folgende Summe gebildet und auf den Verrechnungseinheitenzähler des Benutzers aufaddiert:

TACUNIT (Festpreis für den Aufruf des Teilprogramms)
+ CPUUNIT * CPU-Verbrauch + IOUNIT* I/O-Verbrauch
+ OUTUNIT * Druckausgaben-Verbrauch

Beispiel für die KDCDEF-Generierung dieser Variante:

```
ACCOUNT ACC=ON,CPUUNIT=15

TAC tacnam1,PROGRAM=progname1,TACUNIT=1
TAC tacnam2,PROGRAM=progname2,TACUNIT=2

:

:
```

11.2.3 Abrechnungsphase

In der Abrechnungsphase ermittelt openUTM die pro Teilprogrammlauf verbrauchten Betriebsmittel, berechnet daraus und aus den generierten Gewichten und Festpreisen eine gewichtete Summe. Diese Summe addiert openUTM auf den Verrechnungseinheitenzähler des UTM-Benutzers. Der Wert dieses Zählers ist im Abrechnungssatz enthalten, den openUTM in die Abrechnungsdatei schreibt.

openUTM schreibt immer dann einen Abrechnungssatz, wenn für den Benutzer eine bestimmte Anzahl von Verrechnungseinheiten aufaddiert sind oder wenn sich der Benutzer abmeldet und über keine andere Verbindung mehr bei der UTM-Anwendung angemeldet ist. Die Anzahl von Verrechnungseinheiten, bei der openUTM einen Abrechnungssatz schreibt, legen Sie bei der KDCDEF-Generierung in ACCOUNT MAXUNIT= fest. Dabei müssen Sie Folgendes beachten:

- Den Wert von MAXUNIT sollten Sie nicht zu klein wählen, da ein zu häufiges Schreiben von Abrechnungssätzen die Performance der Anwendung beeinträchtigen könnte.
- Den Wert von MAXUNIT sollten Sie nicht zu groß wählen, da bei einem Abbruch der Anwendung die Verrechnungseinheiten, die noch nicht in die Abrechnungsdatei geschrieben wurden, verloren gehen können (die Abrechnung unterliegt nicht der Transaktionssicherung).

Nachdem der Abrechnungssatz in die Abrechnungsdatei geschrieben wurde, werden der Verrechnungseinheitenzähler und der Zähler für die Anzahl der aufgerufenen TACs auf Null gesetzt.

Abrechnungsphase einschalten

Mit der KDCDEF-Steueranweisung ACCOUNT ACC=ON wird bei der UTM-Generierung die Abrechnung für die UTM-Anwendung eingeschaltet.

Zusätzlich kann die Abrechnungsphase auch im laufenden Betrieb durch die UTM-Administration ein- und ausgeschaltet werden

- durch das Kommando KDCAPPL ACCOUNT=ON
- oder durch WinAdmin/WebAdmin
- oder per KDCADMI-Programmaufruf KC_MODIFY_OBJECT mit obj_type=KC_DIAG_AND_ACCOUNT_PAR

Abrechnungsphase ausschalten

Die Abrechnungsphase können Sie nur per Administration ausschalten:

- durch das Kommando KDCAPPL ACCOUNT=OFF
- oder über WinAdmin/WebAdmin
- oder per KDCADMI-Programmaufruf KC_MODIFY_OBJECT mit obj_type=KC_DIAG_AND_ACCOUNT_PAR

Daten des Abrechnungssatzes

Der Abrechnungssatz ist vom Satztyp UTMA. Er enthält folgende Daten:

- Name der UTM-Anwendung
- UTM-Benutzerkennung
- Zeitpunkt der Anmeldung des Benutzers auf der aktuellen Verbindung
- Wert des Verrechnungseinheitenzählers

- Anzahl der aufgerufenen TACs mit TACUNIT > 0 seit der Anmeldung oder seitdem der letzte Satz geschrieben wurde

Sie können Kalkulationsdaten auch bei laufender Abrechnungsphase erfassen. Damit können Sie die Gewichte jederzeit überprüfen.

11.2.4 Auswertung

Das Ergebnis der Abrechnungsphase sind die Abrechnungssätze in der Abrechnungsdatei. Die Abrechnungsdatei legt openUTM im Unterverzeichnis ACCNT des Basisverzeichnisses *filebase* der UTM-Anwendung an. Die Abrechnungsdatei hat den Namen 0001.*pid*. Dabei ist *pid* die Prozess-ID des Logging-Prozesses der UTM-Anwendung, in dem das Programm *utmlog* abläuft.

In die Datei 0001.*pid* schreibt openUTM sowohl die Abrechnungssätze als auch die Kalkulationssätze.

Nach dem Aus- und Wiedereinschalten der Abrechnungsphase schreibt openUTM in derselben Datei weiter. Die vor dem Ausschalten erfassten Kalkulations- und Abrechnungssätze werden nicht überschrieben.

Die Datei können Sie selbst auswerten.

Der Aufbau der Abrechnungssätze ist im Anhang auf "Aufbau des Abrechnungssatzes" beschrieben.

11.2.5 Fehlersituationen

Kann das Accounting auf Grund eines Fehlers einen Abrechnungssatz und/oder einen Kalkulationssatz nicht schreiben, z.B. weil nicht genügend Platz auf der Platte ist, erzeugt openUTM die Meldung K079 und beendet die Kalkulations- und/oder Abrechnungsphase. Ein Insert der Meldung K079 gibt die Ursache des Fehlers an. Die Anwendung läuft weiter.

Nach Behebung des Fehlers kann die Kalkulations- und/oder Abrechnungsphase durch die UTM-Administration (z. B. mit dem Administrationskommando KDCAPPL) wieder eingeschaltet werden.

11.3 Abrechnung bei verteilter Verarbeitung

Bei verteilter Verarbeitung kann im Prinzip jede der beteiligten Anwendungen Vorgänge in anderen Anwendungen starten. Die Abrechnung bei verteilter Verarbeitung ist vor allem dann sinnvoll, wenn die Rollen ungleich verteilt sind, d.h. eine Anwendung spielt ganz die Rolle des Auftraggebers und andere Anwendungen die Rollen der Auftragnehmer. Die Anwendungen werden in diesem Abschnitt daher als **Auftraggeber-Anwendung** und **Auftragnehmer-Anwendung** bezeichnet.

Die Auftraggeber-Anwendung nutzt Leistungen, die Teilprogramme in fernen Partner-Anwendungen (Auftragnehmer) für sie erbringen. In diesem Fall kann in der Auftraggeber-Anwendung der dabei anfallende Betriebsmittelverbrauch als Festpreis in Rechnung gestellt werden. Dazu ordnen Sie den LTACs in der Auftraggeber-Anwendung Verrechnungseinheiten als Festpreise zu. LTACs sind die Transaktionscodes, die in der Auftraggeber-Anwendung für einen Vorgang in einer Auftragnehmer-Anwendung definiert werden.

Näheres siehe openUTM-Handbuch „Anwendungen generieren“, Anweisung LTAC, Operand LTACUNIT.

Kalkulationsphase (Festlegen der Festpreise)

In der **Auftragnehmer-Anwendung** wird in der Kalkulationsphase der mittlere Betriebsmittelverbrauch der Teilprogramme ermittelt, die Services für die Auftraggeber-Anwendung erbringen. Anhand der hier ermittelten Verbrauchswerte können Sie die Festpreise festlegen, die den Benutzern der LTACs in der Auftraggeber-Anwendung berechnet werden sollen.

In der **Auftraggeber-Anwendung** zählt openUTM in einem Feld des Kalkulationssatzes die Verrechnungseinheiten, die bei den Aufrufen von LTACs anfallen.

Abrechnungsphase

In der **Auftragnehmer-Anwendung** werden alle Verbrauchswerte, die bei der Bearbeitung von Aufträgen für eine Auftraggeber-Anwendung anfallen, wie folgt zugeordnet:

- Bei LU6.1 den Sessions (LSES) zum Auftraggeber
- Bei OSI TP den Associations (OSI-LPAP ... ‚ASSOCIATION-NAME=), falls sich der OSI TP-Auftraggeber nicht unter einer echten Benutzerkennung angemeldet hat.

Die erbrachten Leistungen werden also der Auftraggeber-Anwendung insgesamt in Rechnung gestellt. Die Aufwände für die einzelnen Benutzer der Auftraggeber-Anwendung können nicht ermittelt werden.

In der **Auftraggeber-Anwendung** addiert openUTM beim Aufruf eines LTACs die Verrechnungseinheiten auf den Verrechungseinheitenzähler des Benutzers auf, die in der KDCDEF-Generierung in der LTAC-Anweisung angegeben wurden.

11.4 Einschränkungen

Bei der Nutzung des UTM-Accounting ist Folgendes zu beachten:

- Das Schreiben von Abrechnungsinformationen unterliegt nicht der Transaktionssicherung, deshalb können beim Abbruch einer Anwendung Verrechnungseinheiten verloren gehen. Der Maximalwert pro Benutzer kann per UTM-Generierung begrenzt werden.
- Für Anwendungen mit verteilter Verarbeitung wird in der Kalkulationsphase jeder LTAC-Aufruf mitgezählt. Es wird nicht berücksichtigt, ob nach der PEND-Verarbeitung eine Session belegt werden konnte oder nicht.
- Die Erfassung des Betriebsmittelverbrauchs beginnt vor dem Start eines Teilprogramms, sie endet in der Verarbeitung des PEND-Aufrufs. Die übrige Verarbeitungsleistung (Grundverbrauch) der UTM-Prozesse wird den Benutzern nicht in Rechnung gestellt.
- Das Rücksetzen einer Transaktion hat folgende Auswirkungen: Alle Werte bis auf CPU werden zurückgesetzt. Da openUTM die Verbrauchswerte in der PEND-Verarbeitung aufsummiert, kann eine Rücksetzaktion Verbrauchswerte nur dann zurücksetzen, wenn sie im aktuellen Teilprogrammlauf entstehen.
- Sind seit dem letzten Start der Anwendung für den Benutzer nur Asynchron-Aufträge verarbeitet worden, steht im Abrechnungssatz als Zeitpunkt der Anmeldung an die Anwendung der Wert Null.
- Für den Event-Exit VORGANG wird der Betriebsmittelverbrauch nur am Anfang des Vorgangs erfasst.
- Für den Event-Service BADTACS kann in der Abrechnungsphase kein Teilprogramm-Gewicht berücksichtigt werden.

12 Leistungskontrolle mit openSM2 und KDCMON

Die Performance einer UTM-Anwendung wird von verschiedenen Faktoren beeinflusst. Die Einflussfaktoren liegen zum einen im System-Umfeld einer UTM-Anwendung (Ausbau des Arbeitsspeichers, Leistungsfähigkeit der Peripherie) und zum anderen in der UTM-Anwendung selbst (Konfiguration der Anwendung und Aufbau der Teilprogramme). Im laufenden Betrieb einer Anwendung sollten Sie regelmäßig Leistungskontrollen durchführen, damit Sie rechtzeitig Hinweise auf Leistungsengpässe erhalten. Für UTM-Anwendungen stehen Ihnen die folgenden Instrumente zur Leistungskontrolle zur Verfügung:

- Software Monitor openSM2
- UTM-Messmonitor KDCMON mit dem Auswertungstool KDCEVAL
- Informationsservices der UTM-Administration

Software Monitor openSM2

Eine Leistungskontrolle der UTM-Anwendung können Sie zusammen mit dem Software Monitor openSM2 erstellen. Details dazu finden Sie im Abschnitt „Messdatenerfassung mit openSM2“.

UTM-Messmonitor KDCMON

Für UTM-Anwender steht der UTM-Messmonitor KDCMON zur Verfügung. KDCMON ist eine in openUTM integrierte Funktion und zeichnet Informationen über den Ablauf von UTM-Anwendungen und Anwenderteilprogrammen auf. Zeichnen sich Leistungsengpässe ab, können Sie mit KDCMON Daten erfassen. Die gesammelten Daten müssen Sie mit dem Tool KDCEVAL auswerten. Anhand dieser Auswertung können Sie eine detaillierte Analyse durchführen. Siehe ab "Auswertungslisten".

KDCMON ist damit ein wichtiges Werkzeug zur Beurteilung der Performance in einer UTM-Anwendung. KDCMON kann z.B. zur detaillierten Leistungsuntersuchung eingesetzt werden, wenn Messungen mit Hilfe der UTM-Administration auf Leistungsengpässe hinweisen.

Informationsservices der UTM-Administration

Einige Informationen zur Diagnose von Leistungsengpässen können Sie auch per UTM-Administration abfragen, z. B. über das Administrationskommando KDCINF oder über die grafischen Administrationstools WinAdmin /WebAdmin.

Das Kommando KDCINF STATISTICS liefert Daten über die Auslastung einzelner ausgewählter Komponenten Ihrer UTM-Anwendung, wie beispielsweise der Clients. Mit dem Kommando KDCINF STATISTICS können Sie gleichzeitig allgemeine Statistikinformationen über die Auslastung der gesamten Anwendung abfragen und Kenngrößen für die Leistungskontrolle und die Beurteilung der Performance Ihrer UTM-Anwendungen im laufenden Betrieb ermitteln, wie beispielsweise die Auslastung der Anwendung, die Belegung des Pagepools, Anzahl der aktuell angemeldeten Benutzer, Anzahl der durchgeführten Dialog- oder Asynchron-Transaktionen pro Sekunde, offene Dialog- und Asynchron-Vorgänge usw..

Das Kommando KDCINF PAGEPOOL liefert weitere, detailliertere Daten über die aktuelle Belegung des Pagepools.

Details zu KDCINF finden Sie im openUTM-Handbuch „Anwendungen administrieren“.

Wenn Sie die UTM-Anwendung mit dem grafischen Administrationsarbeitsplatz WinAdmin oder WebAdmin administrieren, können die Statistikdaten auch grafisch dargestellt werden.

12.1 Messdatenerfassung mit openSM2

Der Software Monitor openSM2 liefert umfassende Messdaten zur Leistungsüberwachung von Server- und Speichersystemen. Ab openSM2 V9.0 wird auch die Erfassung von UTM-Anwendungs-spezifischen Daten unterstützt.

Sie sollten die Funktionalität von openSM2 nutzen, um die Auslastung des Systems insgesamt und das Verhalten einer UTM-Anwendung im Besonderen zu überwachen und Leistungsengpässe aufzudecken.

Die Messdaten von openSM2 erlauben jedoch keine Aussage über einzelne Objekte der UTM-Anwendung, z.B. Teilprogramme. Sie zeigen vielmehr das Verhalten der gesamten Anwendung auf, z.B. Mittelwerte zur Transaktionsrate, Durchsatz und Bearbeitungszeit.

Damit openUTM Daten an openSM2 liefern kann und openSM2 UTM-Daten sammeln, speichern und aufbereiten kann, müssen die im Folgenden beschriebenen Voraussetzungen erfüllt sein.

Generierung von openUTM

Die Datenlieferung von openUTM an SM2 muss in der UTM-Anwendung generiert sein. Dazu dient der Operand SM2 in der MAX-Anweisung. In diesem Operanden muss einer der Werte ON oder OFF angegeben werden:

- Wird MAX...,SM2=ON angegeben, so wird die Datenlieferung an openSM2 beim Start der Anwendung eingeschaltet. Sie kann dann im laufenden Betrieb bei Bedarf per UTM-Administration aus- und wieder eingeschaltet werden.
- Wird MAX...,SM2=OFF angegeben, so ist die Datenlieferung an openSM2 für diese Anwendung erlaubt. Sie muss aber im laufenden Betrieb durch die UTM-Administration explizit eingeschaltet werden.

Wenn MAX ...,SM2=NO generiert wird, so kann openUTM für diese Anwendung **keine** Daten an openSM2 liefern. Die Datenlieferung kann dann auch nicht durch die UTM-Administration eingeschaltet werden.

Einschalten der Datenlieferung an openSM2 per UTM-Administration

Der UTM-Administrator kann die Datenlieferung an openSM2 mit dem Kommando KDCAPPL SM2=ON einschalten, wenn dies in der UTM-Generierung vorgesehen wurde (MAX SM2=ON/OFF). Mit KDCAPPL SM2=OFF wird die Datenlieferung ausgeschaltet.

Mit dem Kommando KDCINF SYSPARM kann der UTM-Administrator feststellen, ob die Anwendung Daten an openSM2 liefern kann und ob sie derzeit Daten liefert.

Das Ein- und Ausschalten der Datenlieferung an openSM2 ist auch über die „Administrations-Programmschnittstelle KDCADMI oder über WinAdmin/WebAdmin möglich.

Voraussetzungen bei openSM2

Die Erfassung der Messdaten wird mit der Komponente INSPECTOR von openSM2 realisiert. Hierzu müssen die UTM-Anwendungen in die Konfigurationsdatei des Agenten eingetragen werden. Das genaue Format der Konfigurationszeilen kann der Online-Hilfe im INSPECTOR Manager entnommen werden (Abschnitt „Die Konfigurationsdatei“ des jeweiligen Agenten) entnommen werden.

Die Ausgabe und Auswertung der Messdaten ist in der Dokumentation zu openSM2 beschrieben.

12.2 UTM-Messmonitor KDCMON

Mit KDCMON werden nur UTM-Ereignisse aufgezeichnet. openSM2 und KDCMON können zusammen eingesetzt werden.

KDCMON kann im laufenden Betrieb eingeschaltet und nach der gewünschten Messdauer wieder abgeschaltet werden. Die Daten werden gepuffert auf Datei geschrieben.

Zur Auswertung der von KDCMON erfassten Daten steht das Tool KDCEVAL zur Verfügung.

12.2.1 Erfassung starten und beenden

Die Datenerfassung können Sie mit dem folgenden Administrations-Kommando ein- und ausschalten:

```
KDCDIAG KDCMON={ ON | OFF }
```

Diese Administrationsfunktion steht auch an der Programmschnittstelle KDCADMI und in WinAdmin/WebAdmin zur Verfügung.

Der UTM-Administrator kann jederzeit mit Hilfe des Kommandos `KDCINF SYSPARM` feststellen, ob Daten erfasst werden oder nicht.

Wenn openUTM beim Einschalten feststellt, dass die KDCMON-Funktion nicht verfügbar ist, dann wird folgende Meldung mit dem Standard-Ziel SYSLOG ausgeben:

```
K080 KDCMON-Funktion nicht verfuegbar
```

Mögliche Ursache: Die Kommunikation mit dem Log-Prozess ist gestört (nur bei Unix- und Linux-Systemen).

Wenn openUTM bei laufender Erfassung feststellt, dass die KDCMON-Funktion nicht mehr verfügbar ist, dann schaltet openUTM die Datenerfassung aus und protokolliert dies ebenfalls mit der Meldung K080.

Beim Erfassen werden folgende Dateien erzeugt:

Unix- und Linux-Systeme

Für jedes Messintervall erzeugt openUTM eine Datei mit folgendem Namen:

filebase`/KDCMON/`*nnnn.pid*

Dabei ist *nnnn* die laufende Nummer des Messintervalls, beginnend bei 0001 nach dem Anwendungsstart, und *pid* die Prozess-ID des utmlog-Prozesses, der zum Anwendungslauf gehört.

Windows-Systeme

openUTM erzeugt eine Datei mit dem festen Namen *filebase*`\KDCMON\0001.`

Diese Datei bleibt vom ersten Kommando KDCDIAG KDCMON=ON bis zum Ende des Anwendungslaufs geöffnet. Sie können diese Datei kopieren, nachdem Sie das Kommando KDCDIAG KDCMON=OFF eingegeben haben.

Bitte beachten Sie, dass die Datei nach dem nächsten Kommando KDCDIAG KDCMON=ON oder nach dem nächsten Start der Anwendung wieder leer ist.

12.2.2 Daten auswerten mit KDCEVAL

Mit dem Tool KDCEVAL werden die mit KDCMON aufgezeichneten Daten ausgewertet. In einem Auswertungslauf können nur die Daten *einer* Anwendung ausgewertet werden. KDCEVAL benötigt zur Steuerung einige Parameter, die Sie nach dem Starten von KDCEVAL eingeben müssen.

Starten von KDCEVAL

Vor dem Starten von KDCEVAL müssen Sie die Datei, die Sie auswerten möchten (*nnnn.pid* im Verzeichnis `KDCMON`, s.o.), in das aktuelle Verzeichnis kopieren und unter dem Name`evalin` abspeichern.

Danach rufen Sie KDCEVAL wie folgt auf:

utmpfad`/ex/kdceval` (Unix- und Linux-Systeme) bzw.

utmpfad`\ex\kdceval` (Windows-Systeme)

Nach dem Start des Auswertungsprogramms im Dialog fordert KDCEVAL mit der folgenden Meldung die Eingabe von Steuerparametern an:

```
PLEASE ENTER COMMANDS OR 'HELP' OR 'END'
```

Steuerparameter von KDCEVAL

Das Programm liest die Parameter von *stdin*. Die einzelnen Kommandos, mit denen Sie die Auswertung steuern können, haben folgendes Format:

APPLINAME appliname
Name der Anwendung, für die die Auswertung durchgeführt werden soll.

TIME FROM={ t1 | START }, TO={ t2 | STOP }

Zeitangabe zur Definition des Auswertungsintervalls.

FROM=t1

Startzeitpunkt der Auswertung in Sekunden.
Die Zeitangabe erfolgt relativ zum Einschalten der Erfassung (z.B. mit dem Kommando KDCDIAG).

FROM=START

Es wird vom Anfang der Datei an ausgewertet.

TO=t2

Endzeitpunkt der Auswertung in Sekunden.
Die Zeitangabe erfolgt relativ zum Einschalten der Erfassung (z.B. mit dem Kommando KDCDIAG).

TO=STOP

Es wird bis zum Dateiende ausgewertet.
Für *t1* und *t2* gilt:
Minimalwert: 0
Maximalwert: 99999999

LIST { (liste$_1$, liste$_2$,...,liste*n* [,TABLE]) | (STD [,TABLE]) | (ALL [,TABLE]) }

liste$_1$, liste$_2$,...,liste$_n$

Namen der Einzellisten, die aufbereitet werden sollen. Die Namen, die Sie hier angeben können, finden Sie auf "Auswertungslisten". Die Listen TRACE und TRACE2 dürfen nicht zusammen angegeben werden.

STD

Diese Auswertung umfasst die Listen TASKS, SUMM, TIMES und TCLASS.

ALL

Die Auswertung umfasst alle Listen außer TRACE und TRACE2.
Werden ALL oder STD ohne TABLE angegeben, dann können die runden Klammern weggelassen werden.

TABLE

Wird zusätzlich TABLE angegeben, dann werden die Listen in einem Tabellenformat erzeugt, das auf einem PC mit Excel oder einem anderen Tabellenkalkulator weiterverarbeitet werden kann, siehe "Auswertungsdaten auf dem PC bearbeiten". TABLE wirkt nur auf die Einzellisten TASKS, TIMES, TCLASS, TACCL, TACPT und TACLIST.

OPTION **D**ECIMAL-**S**EPARATOR={ **C**OMMA | **P**OINT }, SHOW-TSN={ **F**IRST | **A**LL }

Definiert das Dezimaltrennzeichen.

DECIMAL-SEPARATOR=COMMA

Als Dezimaltrennzeichen wird das Komma verwendet.

DECIMAL-SEPARATOR=POINT

Als Dezimaltrennzeichen wird der Punkt verwendet, Standardwert.

SHOW-TSN=FIRST

In der Liste TRACE2 wird in aufeinanderfolgenden Sätzen mit identischer PID die PID nur im ersten Satz dieser Folge ausgegeben. Die Folgesätze einer solchen Folge von Sätzen mit gleicher PID enthalten im PID-Feld Anführungszeichen (").Dies ist der Standardwert.

SHOW-TSN=ALL

In der Liste TRACE2 wird die PID in jedem Satz ausgegeben. Dies kann dann sinnvoll sein, wenn die Ausgabeliste mit einem Programm bearbeitet oder ausgewertet werden soll.

END

Mit diesem Kommando wird die Parametereingabe beendet.

Bei Auswertungen im Dialog kann auch das Kommando HELP eingegeben werden. Es werden dann die Syntax der Kommandos und die möglichen Listennamen ausgegeben.

Fehler und Meldungen

- Fehlt eines der Kommandos APPLINAME, TIME oder LIST, so wird die Auswertung mit folgender Fehlermeldung abgebrochen:

  ```
  MANDATORY COMMAND MISSING
  ```

- Bei einem Syntaxfehler erscheint die folgende Meldung und das fehlerhafte Kommando wird angezeigt:

  ```
  ERROR IN COMMAND
  ```

- Sind die Zeitangaben *t1* und *t2* inkonsistent, so wird folgende Meldung ausgegeben:

  `KDCEVAL: WRONG TIME INPUT`
- Werden in der Datei keine Sätze für die Anwendung gefunden oder sind für das Auswertungsintervall keine Daten vorhanden, so wird eine der folgenden Meldungen ausgegeben:

  `NO EVALUATION : NO RECORD WITH APPLINAME FOUND`

  oder

  `NO EVALUATION : NO RECORD IN TIME_INTERVAL`
- Bei einem DMS-Fehler werden folgende Meldungen ausgegeben:
  - Wenn KDCEVAL die Datei `evalin` nicht gefunden hat:

    `KDCEVAL: NO KDCMON FILE`

    `KDCEVAL: NO EVALUATION`
  - Wenn die Datei `evalin` nicht durch KDCMON erzeugt wurde:

    `NO EVALUATION: NO VALID KDCMON FILE`
- Versionsprüfung:

  Die Auswertung der KDCMON-Daten durch KDCEVAL ist nur möglich, wenn KDCEVAL die gleiche openUTM-Version wie der UTM-Systemcode hat. KDCEVAL prüft die Version der KDCMON-Daten. Erkennt KDCEVAL eine unzulässige Version, bricht KDCEVAL die Auswertung mit der folgenden Meldung ab:

  `NO EVALUATION: INPUT FILE FROM INVALID UTM VERSION`

Ergebnis der Auswertung mit KDCEVAL

KDCEVAL schreibt die Auswertung in die Datei

`kdcmon.`*`appliname`*

Diese Datei wird im aktuellen Verzeichnis abgelegt.

12.2.3 Auswertungsdaten auf dem PC bearbeiten

Wenn Sie bei KDCEVAL im Steuerparameter LIST zusätzlich zu den Listennamen den Operanden TABLE angeben, dann werden diese Listen in Tabellenform erzeugt. Diese Art der Aufbereitung ist nur für Listen TASKS, TIMES, TCLASS, TACCL, TACPT, und TACLIST möglich.

Diese so erzeugten Listen können Sie auf dem PC mit einem Tabellenkalkulator wie z.B. Excel bearbeiten und grafisch aufbereiten. Für Excel wird dazu das Makro `kdceval.xls` ausgeliefert.

Dazu gehen Sie wie folgt vor:

1. Übertragen Sie die von KDCEVAL erzeugte Listen-Datei sowie das Makro `kdceval.xls` auf einen PC.

   Das Makro verlangt, dass die auszuwertende Datei die Endung `.txt` besitzt!
2. Rufen Sie das Makro `kdceval.xls` auf und lesen Sie die Listen-Datei in Excel ein. Excel erzeugt dann für jede Liste ein eigenes Tabellenblatt sowie ein Zusatzblatt mit Übersichtsinformationen (Summary-Blatt).
3. Bearbeiten Sie die einzelnen Listen nach Ihren Wünschen, indem Sie z.B. eine Liste sortieren und anschließend in ein Kurven- oder Balkendiagramm umwandeln.

12.2.4 Auswertungslisten

Jede Auswertungsliste besteht aus

- einer Überschrift, die den Namen der Auswertungsliste enthält
- einem Kopf, der für alle Listen identisch ist
- der spezifischen Auswertungsliste

Der Listenkopf hat folgenden Aufbau:

```
NAME OF APPLICATION : appliname      DATE           : day month dd yyyy hh:mm:ss
COMMENCEMENT TIME   : rel-sec SEC.   KDCEVAL VERSION: V07.0A00
END TIME            : rel-sec SEC.   openUTM VERSION: V07.0A00
SYSTEM INFORMATION  : system info
APPLICATION INFO    : Appli-Mode S   Cache size,loc   BLKSIZE : 4K   Test-Mode OFF
Data Compression OFF
```

mit folgender Bedeutung der Felder (soweit nicht selbsterklärend):

NAME OF APPLICATION	Name der Anwendung
DATE	Datum der Datenerfassung mit KDCMON
COMMENCEMENT TIME	Anfangszeitpunkt des ausgewählten Auswertungszeitraums (relativ zum Startzeitpunkt der Datenerfassung)
END TIME	Endzeitpunkt des ausgewählten Auswertungszeitraums (relativ zum Startzeitpunkt der Datenerfassung)
SYSTEM INFORMATION	Name und Betriebssystemversion des Rechners sowie Ablaufmodus (Bitmode: 32 Bit oder Bitmode: 64 Bit)
Cache size, loc	Größe und Speicherort des Cache. Für den Speicherort wird immer PS (Programmraum) ausgegeben.

Bei der TRACE- und TRACE2-Liste enthält END TIME den Wert 999999, wenn man die ganze Datei auswertet (Parameter TIME FROM=START,TO=STOP).

Bei den Bearbeitungszeiten handelt es sich stets um die ELAPSED TIME (Realzeit).

Es sind folgende Einzel-Auswertungen und Kombinationen dieser Auswertungen möglich:

TASKS	UTILIZATION OF THE UTM TASKS
SUMM	TRANSACTION EVALUATION
TIMES	DISTRIBUTION OF PROCESSING TIMES
KCOP	KDCS CALLS STATISTIC

WAIT	WAITING TIMES
TCLASS	EVALUATION OF THE TAC CLASSES
TACCL	TAC SPECIFIC TAC CLASS EVALUATION
TACPT	TAC SPECIFIC DISTRIBUTION OF PROCESSING TIMES
TACLIST	TAC SPECIFIC STATISTICS
TRACE	TASK SPECIFIC TRACES
TRACE2	TASK PERFORMANCE TRACES

Im Folgenden werden die einzelnen Auswertungslisten beschrieben.

12.2.4.1 TASKS: UTILIZATION OF THE UTM TASKS

In dieser Liste wird ein Überblick über die Auslastung der Prozesse der Anwendung gegeben. Außerdem wird der CPU-Verbrauch und die Anzahl der Ein- und
Ausgabeoperationen (I/O's) für jeden einzelnen UTM-Prozess aufsummiert und für alle Prozesse der Anwendung angezeigt.

```
                        1 = Program
                        2 = System code
                        4 = Bourse Wait

|---------------------|
| PID  | START TIME   | TASK UTILIZATION , Number Used Tasks: 4 , Number System Tasks: 0
| 9253 | 09:33:05.542 | <1><---2--><-----------------------------------4--------------------------------->
| 9250 | 09:32:41.114 | 1><-----2-----><---------------------------------4------------------------------->
| 9217 | 09:32:41.114 | <-1-><------2------><-------------------------------4----------------------------->
| 9144 | 09:32:39.010 | <1><----2---><-----------------------------------4--------------------------------->
|------|--------------|

PID    CPU-time  Number I/O   Program   System    Bourse Wait   System Task
9253    19655       16677      13322     32766      323895           N
9250    16683       14145      11206     54501      328704           N
9217    33065       27990      22564     60260      311587           N
9144    19863       16575      13607     41018      341891           N
Summ    89266       75387
```

In der Liste bedeuten:

PID	Prozess ID des UTM-Prozesses.
START TIME	Zeit des ersten Satzes dieses Prozesses (absolut).
Program	Zeitanteil des Anwendungsprogramms im UTM-Prozess.
System	Zeitanteil des UTM-Systemcodes.
Bourse Wait	Zeitanteil, den der Prozess auf neue Aufträge an der Auftragswarteschlange gewartet hat.
System Task	Gibt an, ob dieser Prozess ein UTM-System-Prozess ist.

Bei den in den Spalten Program, System, Database und Bourse Wait ausgegebenen Zeiten handelt es sich um Real-Zeiten. Die Einheit ist in Millisekunden (genauso wie für die CPU-Zeit).

Für die Auswertung TASKS ist ein Herabsetzen der Prozess-Anzahl während des Auswertungsintervalls zu vermeiden, denn es führt zu verzerrten Ergebnissen. In diesem Fall sollten Sie weitere Auswertungsintervalle verwenden.

12.2.4.2 SUMM: TRANSACTION EVALUATION

In dieser Liste wird ein Überblick über die Vorgänge und Transaktionen für den Auswertungszeitraum gegeben. Es werden nur solche Transaktionen berücksichtigt, die vollständig im Auswertungszeitraum liegen. Zusätzlich gibt das Auswertungstool
KDCEVAL den CPU-Verbrauch von allen Teilprogrammläufen aus, die im Auswertungsintervall beendet wurden.
Die Liste hat folgendes Format:

```
COUNT OF TRANSACTIONS                          :      19126
COUNT OF SERVICES                              :       3059  1)
COUNT OF DIALOG STEPS                          :      19126
NUMBER OF DIALOG STEPS PER SECOND              :      59,91
TOTAL CPU-TIME USED IN MSEC                    :      89094  2)
```

1)SERVICE = Vorgang

2)In dieser Zeile wird die Summe des CPU-Verbrauchs der Teilprogrammläufe ausgegeben. Darin ist auch der Verbrauch im UTM- und Betriebssystemcode enthalten, soweit er innerhalb der Teilprogrammläufe anfällt, sowie die Start- und Endeverarbeitung für die Teilprogrammläufe in openUTM. Andere Aktionen der UTM-Prozesse, die nicht direkt zu Teilprogrammläufen gehören, sind nicht enthalten.

12.2.4.3 TIMES: DISTRIBUTION OF PROCESSING TIMES

In dieser Liste wird in tabellarischer Form eine Verteilung der Bearbeitungszeiten für die Teilprogramme ausgegeben. In diesen Zeiten ist die Wartezeit vor der Bearbeitung durch openUTM nicht enthalten.

Die Liste hat folgendes Format:

```
|----------------------------------------|
| PROCESSING TIMES |   NUMBER   | PERCENT |
| (MSEC)           |            |         |
|----------------------------------------|
|     0 -      100 |      21721 |   99,62 |
|   101 -      200 |          2 |    0,00 |
|   201 -      500 |          0 |    0,00 |
|   501 -     1000 |          0 |    0,00 |
|  1001 -     2000 |         80 |    0,36 |
|  2001 -     5000 |          0 |    0,00 |
|  5001 -    10000 |          0 |    0,00 |
| 10001 -    20000 |          0 |    0,00 |
| 20001 -    50000 |          0 |    0,00 |
| 50001 -   100000 |          0 |    0,00 |
|       >   100000 |          0 |    0,00 |
|----------------------------------------|
```

In dieser Liste ist die Anzahl der vollständigen Teilprogrammläufe und der prozentuale Anteil für die jeweilige Zeitklasse angegeben.

12.2.4.4 KCOP: UTM CALLS STATISTIC

In dieser Tabelle ist für die UTM-Aufrufe angegeben, wie oft sie im Auswertungszeitraum aufgetreten sind.

Unter *others* sind Aufrufe aufgeführt, die nicht in der Liste der in KDCEVAL bekannten Aufrufe enthalten sind.

In dieser Liste sind Aufrufe enthalten, die openUTM für interne Bearbeitungen aufruft und die dem Anwender nicht zur Verfügung stehen:

CONT	Aufruf nach einer Formatierung oder einer Datenbankkommunikation
ADMI	UTM-Administrationsaktion
WAIT	Ende der Bearbeitung eines Programmlaufes
NOOP	Pufferbereich der MESSAREA muss geleert werden

Die KCOP-Liste hat folgendes Format:

OP	OM	NUMBER	OP	OM	NUMBER
ADMI		7	MPUT	HM	0
APRO	AM	0	MPUT	ID	0
APRO	DM	0	MPUT	NE	6
APRO	IN	0	MPUT	NT	24378
CONT		19019	MPUT	PM	0
CTRL	AB	0	MPUT	RM	0
CTRL	EC	0	NOOP		0
CTRL	PE	0	PADM	AC	0
CTRL	PR	0	PADM	AI	0
CTRL	SC	0	PADM	AT	0
DADM	CS	0	PADM	CA	0
DADM	DA	0	PADM	CS	0
DADM	DL	0	PADM	OK	0
DADM	MA	0	PADM	PI	0
DADM	MV	0	PADM	PR	0
DADM	RQ	0	PEND	ER	0
DADM	UI	0	PEND	FC	0
DGET	BF	0	PEND	FI	3060
DGET	BN	0	PEND	FR	0
DGET	FT	0	PEND	KP	0
DGET	NT	0	PEND	PA	2677
DGET	PF	0	PEND	PR	0
DGET	PN	0	PEND	PS	0
DPUT	NE	0	PEND	RE	16067
DPUT	NI	0	PEND	RS	0
DPUT	NT	0	PEND	SP	0
DPUT	RP	0	PGWT	CM	0
DPUT	QE	0	PGWT	KP	0
DPUT	QI	0	PGWT	PR	0
DPUT	QT	0	PGWT	RB	0
DPUT	+I	0	PGWT	RT	0
DPUT	-I	0	PGWT	ST	0
DPUT	+T	0	PTDA		144
DPUT	-T	0	QCRE	NN	0
FGET		145	QCRE	WN	0
FPUT	NE	225	QREL	RL	0

```
| FPUT  NT                2   |    | RSET                    4   |
| FPUT  RP                0   |    | SGET  GB             2678   |
| FPUT  UF                0   |    | SGET  KP                0   |
| GTDA                    1   |    | SGET  RL                0   |
| INFO  CD                0   |    | SGET  US                0   |
| INFO  CK                0   |    | SIGN  CK                0   |
| INFO  DT                0   |    | SIGN  CL                0   |
| INFO  FH                0   |    | SIGN  CP                0   |
| INFO  GN                0   |    | SIGN  OB                0   |
| INFO  LO                0   |    | SIGN  OF                0   |
| INFO  PC                0   |    | SIGN  ON                0   |
| INFO  SI                0   |    | SIGN  ST                0   |
| INIT                21804   |    | SMSG                    0   |
| INIT  PU                0   |    | SPUT  DL                0   |
| INIT  MD                0   |    | SPUT  ES                0   |
| LPUT                    0   |    | SPUT  GB             2677   |
| MCOM  BC                0   |    | SPUT  MS                0   |
| MCOM  EC                0   |    | SPUT  US                0   |
| MGET                18981   |    | SREL  GB                0   |
| MGET  NT                0   |    | SREL  LB                0   |
| MPUT  CM                0   |    | UNLK  DA                0   |
| MPUT  EM                0   |    | UNLK  GB                0   |
| MPUT  ES                0   |    | UNLK  US                0   |
| MPUT  GC                0   |    | WAIT                19153   |
| OTHERS                  0   |    |                             |
|-----------------------------|    |-----------------------------|
```

12.2.4.5 WAIT: WAITING TIMES

Zur Ermittlung von Stausituationen fügt openUTM bei eingeschaltetem KDCMON in regelmäßigen Zeitabständen Messaufträge in die Auftragswarteschlange ein. Mit Hilfe des Zeitpunktes, zu dem der Auftrag eingefügt wurde (Zeitstempel absolut) und dem Bearbeitungszeitpunkt kann die Wartezeit der Aufträge in der UTM-Warteschlange ermittelt werden. Der Zeitabstand zwischen den einzelnen Pseudoaufträgen beträgt etwa 10 Sekunden.

In der WAIT-Liste wird Folgendes protokolliert:

- In der Spalte WAITING TIME wird die ermittelte Wartezeit für jeden Pseudoauftrag in Sekunden ausgegeben.
- Für diese Wartezeiten berechnet das Auswertungstool KDCEVAL zusätzlich den Maximal-, Minimal- und Mittelwert in Sekunden und zeigt diese Werte unter UTM WAITING TIMES an.
- In der Spalte NUMBER OF TASKS wird die Anzahl der Prozesse, die zu diesem Zeitpunkt in der Anwendung zur Verfügung standen, ausgegeben. Die UTM-System-Prozesse sind in dieser Zahl nicht berücksichtigt.

Wenn die Wartezeit zu lang wird, sollte die Anzahl der UTM-Prozesse erhöht werden.

Die WAIT-Liste hat folgendes Format:

```
+-------------------------------------------------+
|  TIME STAMP      | WAITING TIME | NUMBER OF TASKS |
+-------------------------------------------------+
|  09:32:41.114    |        0,000 |               4 |
+-------------------------------------------------+
|  09:32:51.114    |        0,000 |               4 |
+-------------------------------------------------+
|  09:33:01.114    |        0,018 |               4 |
+-------------------------------------------------+
|  09:33:11.534    |        0,000 |               4 |
+-------------------------------------------------+
|  09:33:21.534    |        0,008 |               4 |
+-------------------------------------------------+
|  09:33:31.534    |        0,000 |               4 |
+-------------------------------------------------+
|  09:33:41.534    |        0,000 |               4 |
+-------------------------------------------------+
                  UTM WAITING TIMES:
TIME STAMP :  09:33:01.114    WAITING TIME MAXIMUM :        0,018
TIME STAMP :  09:33:41.534    WAITING TIME MINIMUM :        0,000
NUMBER OF ENTRIES:       7    WAITING TIME AVERAGE :        0,004
```

12.2.4.6 TCLASS: EVALUATION OF THE TAC CLASSES

Die TCLASS-Liste enthält in tabellarischer Form einen Überblick über die Auftragsbearbeitung von TACs der einzelnen TAC-Klassen (1 bis 16). In der Auswertung sind in der TAC-Klasse 0 alle die Dialog-TACs zusammengefasst, denen bei der UTM-Generierung mit KDCDEF keine TAC-Klasse zugeordnet wurde.

Bei der UTM-Generierung kann der Anwender festlegen, wieviele Prozesse zu einem Zeitpunkt maximal für eine TAC-Klasse arbeiten dürfen. Ist diese Anzahl erreicht, so werden weitere Aufträge in eine TAC-Klassen-spezifische Warteschlange eingereiht.

TAC-CLASS	NUMBER CALLS	DISTRIBUTION IN PERCENT			AVERAGE WAIT TIME (IN MSEC)	MAXIMUM WAIT TIME (IN MSEC)	MINIMUM WAIT TIME (IN MSEC)
		NUMBER CALLS	WAIT-TIME=0	WAIT-TIME>0			
0	10	0,04					
1	21646	99,27	97,90	2,10	184	1010	1
2	0	0,00			0	0	0
3	3	0,01	66,66	33,34	296	296	296
4	0	0,00			0	0	0
5	0	0,00			0	0	0
6	0	0,00			0	0	0
7	0	0,00			0	0	0
8	0	0,00			0	0	0
9	145	0,66	2,75	97,25	1	2	1
10	0	0,00			0	0	0
11	0	0,00			0	0	0
12	0	0,00			0	0	0
13	0	0,00			0	0	0
14	0	0,00			0	0	0
15	0	0,00			0	0	0
16	0	0,00			0	0	0
		97,26	2,74		141		

```
   21659 DIALOG TACS WERE CALLED
     145 ASYNCHRONOUS TACS WERE CALLED
```

Die TCLASS-Liste enthält folgende Angaben:

- In der Spalte NUMBER CALLS wird für eine TAC-Klasse die Anzahl der TAC-Aufrufe im Auswertungszeitraum angegeben.
- Die Spalte DISTRIBUTION IN PERCENT enthält Prozentwerte.
  Die Unterspalte NUMBER CALLS gibt den prozentualen Anteil der Aufrufe einer TAC-Klasse zur Anzahl aller TAC-Aufrufe an. Die nächsten beiden Spalten enthalten eine prozentuale Aufteilung der Aufrufe dieser TAC-Klasse in
  - Aufrufe, die sofort bearbeitet wurden (WAITTIME=0) und
  - Aufrufe, die in eine TAC-Klassen-spezifische Warteschlange eingereiht werden mussten (WAITTIME>0)

- Die Werte in den Spalten AVERAGE / MINIMUM / MAXIMUM WAIT TIME beziehen sich auf die Aufträge, die openUTM vorübergehend in eine TAC-Klassen-spezifische Warteschlange verdrängt hat. Es wird die mittlere, minimale bzw. maximale Wartezeit eines Auftrags pro TAC-Klasse angezeigt.

  > **i** Die mittlere Wartezeit von Aufträgen pro TAC-Klasse kann im laufenden Betrieb einer Anwendung auch mit dem Administrationskommando KDCINF TACCLASS oder mit der entsprechenden Funktion bei WinAdmin/WebAdmin oder KDCADMI abgefragt werden.

Wartezeit bei Dialog-Aufträgen

Bei Dialog-Aufträgen ist die Wartezeit die Zeitspanne zwischen der Entgegennahme des Auftrags durch die Anwendung (Abholen des Auftrags von der Warteschlange der Anwendung) und dem Start des Teilprogramms. Es kann auch zwischen einzelnen Teilprogrammen zur Verdrängung kommen.

Wartezeit bei Asynchron-Aufträgen

openUTM erfasst auch die Wartezeit von Asynchron-Aufträgen, die sich wie folgt definiert:

Asynchron-Auftrag	**Definition „Wartezeit“**
Eingabe Asynchron-TAC	Zeitspanne zwischen der Entgegennahme des Auftrags durch openUTM und dem Starten des Asynchron-Vorgangs
FPUT-Aufruf im Teilprogramm	Zeitspanne zwischen dem Ende der Transaktion, in der der FPUT-Auftrag ausgeführt wurde, und dem Starten des Asynchron-Vorgangs
DPUT-Aufruf im Teilprogramm	Zeitspanne zwischen der Umwandlung des DPUT in einen FPUT und dem Starten des Asynchron-Vorgangs

Wurde der Asynchron-Auftrag nicht im aktuellen Anwendungslauf erzeugt, so wird als asynchrone Wartezeit immer die Zeitdifferenz zwischen dem Start der Anwendung und dem Start des Asynchron-Auftrags genommen.

12.2.4.7 TACCL: TAC SPECIFIC TAC CLASS EVALUATION

In der Liste TACCL sind die gleichen Informationen wie in der TCLASS-Liste aufgeführt, nur aufgegliedert auf die einzelnen Transaktionscodes. Es sind alle TACs aufgeführt, die im Auswertungszeitraum aufgerufen wurden. Die TACs sind in der Reihenfolge des ersten Auftretens aufgeführt. Die Bedeutung der einzelnen Spalten kann der Beschreibung des TCLASS-Listenformates entnommen werden.

TAC	TAC-CLASS	NUMBER CALLS	DISTRIBUTION IN PERCENT		AVERAGE WAIT TIME (MSEC)
			NUMBER CALLS	WAIT-TIME=0	WAIT-TIME>0	
CVARL	1	2678	12,28	98,91	1,09	356
CVARL1	1	16066	73,68	98,91	1,09	355
CVAR1	1	2677	12,27	99,02	0,98	358
KDCINF	3	2	0,00	100,00	0,00	0
UPDEMP	0	3	0,01			
PTDA	1	144	0,66	0,69	99,31	11
PTDAA	9	144	0,66	2,77	97,23	1
...	.	.	.	.	.	.
...	.	.	.	.	.	.

Für TACs der TAC-Klasse 0 werden keine Angaben für WAIT TIME eingetragen.

12.2.4.8 TACPT: TAC SPECIFIC DISTRIBUTION OF PROCESSING TIMES

In dieser Tabelle sind für alle im Auswertungszeitraum bearbeiteten TACs die minimale (MIN), die maximale (MAX) und die mittlere (MEAN) Bearbeitungszeit aufgeführt. Es werden nur solche TACs aufgenommen, deren Start- und Beendigungszeitpunkt innerhalb des Auswertungszeitraums liegen. Die Liste hat folgendes Format:

```
|-----------------------------------------------------|
|          | PROCESSING TIME PER TAC (IN MSEC)        |
|    TAC   |    MEAN     |    MIN      |     MAX      |
|-----------------------------------------------------|
| UPDEMP   |          28 |          13 |           58 |
| CVAR1    |          14 |           0 |           50 |
| KDCINF   |          11 |          11 |           12 |
| PTDA     |           8 |           7 |           24 |
| CVARL    |           6 |           6 |          114 |
|-----------------------------------------------------|
| PTDAA    |           5 |           5 |           12 |
| CVARL1   |           5 |           0 |          151 |
| ...      |           . |           . |            . |
| ...      |           . |           . |            . |
|-----------------------------------------------------|
```

Die Tabelle ist nach mittleren Bearbeitungszeiten fallend sortiert. Angezeigt werden nur TACs mit einer mittleren Bearbeitungszeit > 0.

12.2.4.9 TACLIST: TAC SPECIFIC STATISTICS

In dieser Liste werden die folgenden TAC-spezifischen Informationen erfasst:

- die durchschnittliche KB-Größe (Spalte AVERAGE SIZE OF KB)
- die Aufteilung der Bearbeitungszeit in

  1: Program
  2: System code

Die Liste hat folgendes Format:

```
-----------------------------------------------
|   TAC   | NUMBER CALLS | AVERAGE SIZE OF KB |
-----------------------------------------------
| CVARL   |        2678  |              956  | <---------1--------><-------------2------------>
-----------------------------------------------
| CVARL1  |       16066  |              956  | <---------1--------><-------------2------------>
-----------------------------------------------
| CVAR1   |        2677  |              956  | <---1---><-------------------2------------------>
-----------------------------------------------
| KDCINF  |           2  |                0  | <--------1--------><--------------2-------------->
-----------------------------------------------
| UPDEMP  |           3  |                0  | <--------------------1------------------><---2-->
-----------------------------------------------
| PTDA    |         144  |                1  | <-------1-------><---------------2--------------->
-----------------------------------------------
| PTDAA   |         144  |                1  | <-----------1----------><------------2----------->
-----------------------------------------------
| ...     |           .  |                .  |
-----------------------------------------------
| ...     |           .  |                .  |
-----------------------------------------------
```

Die Liste ist nicht sortiert, die TACs sind in der Reihenfolge aufgelistet, in der sie in der Datei zuerst auftreten.

Es werden nur die TACs berücksichtigt, deren Start- und Beendigungszeitpunkt innerhalb des Auswertungsintervalls liegen.

12.2.4.10 TRACE: TASK SPECIFIC TRACES

Zur genaueren Analyse des Ablaufs einer UTM-Anwendung können TRACE-Listen erstellt werden. In diesen Listen werden in zeitlicher Reihenfolge für die einzelnen UTM-Prozesse alle UTM-Aufrufe aufgelistet.

Die TRACE-Liste enthält immer nur die Daten der ersten 6 Prozesse. Existieren für den Auswertungszeitraum Daten von mehr als 6 Prozessen, dann sollte zur Auswertung die Tabelle TRACE2 verwendet werden.

Die Liste ist in zeitlicher Reihenfolge sortiert.

Die Spalte TIME STAMP enthält den Zeitstempel des entsprechenden Aufrufs, der protokolliert wurde (mit einer Genauigkeit von Millisekunden).

Die Liste TRACE erfasst folgende Ereignisse und Daten:

- den aufgerufenen Transaktionscode (TAC)
- die Transaktions-ID. In openUTM wird für jede Transaktion eine eindeutige Transaktions-ID vergeben. Dieses Kennzeichen wird auch den angeschlossenen Datenbanken an der UTM-DB-Schnittstelle übergeben. Damit wird es u.a. möglich, Traces der Datenbank mit diesen Traces von openUTM zu koppeln und Zusammenhänge zwischen UTM- und DB-Abläufen herzustellen. Die Transaktions-ID besteht aus vier Teilen:

SC	Session Counter: er nummeriert die Anwendungsläufe. Nach Neugenerierung ist er 1, bei jedem Start der Anwendung wird er um 1 hochgezählt.
VC	Vorgangs Counter: er nummeriert die Vorgänge innerhalb eines Anwendungslaufes und läuft bis 16 777 216 (2^{24}).
TC	Transaction Counter: er nummeriert die Transaktionen innerhalb eines Vorgangs und läuft bis 32 768 (2^{15}).
VN	Vorgangs Nummer: das ist die Nummer einer UTM-internen Tabelle zur Verwaltung der Vorgänge.

  Diese vier Teile werden nach dem KDCS-Aufruf INIT protokolliert.
  Für den Anwender sind die Angaben zu VC und TC interessant.
- Alle UTM-Aufrufe mit ihren Operationsmodifikationen. Auch UTM-interne Aufrufe (WAIT, CONT, ...) werden aufgeführt. Sehen Sie auch die KCOP-Liste.

  Zusätzlich wird protokolliert:
  - KCMF bei Aufrufen, bei denen KCMF relevant ist
  - KCRN bei Aufrufen, bei denen KCRN relevant ist
  - KCLT bei den Aufrufen PADM/DADM
  - bei einem Abbruch mit PEND ER/ FR als Diagnoseinformation:
    - der TAC des Teilprogramms, das den Abbruch verursacht hat
    - die Returncodes KCRCDC und KCRRCC
    - VC und TC, für die Zuordnung zum abgebrochenen Vorgang
  - bei einem Aufruf PEND RS als Diagnoseinformation:
    - der TAC des aktuellen Teilprogramms
    - VC und TC, für die Zuordnung zum abgebrochenen Vorgang

Solange kein Prozesswechsel stattfindet, sind alle Aufrufe zur Bearbeitung eines Dialogschrittes in der gleichen PID-Spalte hintereinander aufgeführt. Nach PEND PA/PR/SP kann ein Prozesswechsel nur bei einem Wechsel der TAC-Klasse auftreten. Die Unterbrechung eines Prozesses durch das Betriebssystem wird dadurch sichtbar, dass mitten in der Bearbeitung eines Dialogschrittes die Aufrufe in einer anderen Prozess-Spalte fortgesetzt werden.

Beispiel

```
  TIME STAMP|  PID :9144        |  PID :9217        |  PID :9250        |  PID :
9253        |  PID :      ...
09:39:11.454+-                 -+-                -+-                -|
CVARL1   +-            ...
09:39:11.454|                   |                   |                   |
INIT             |             ...
09:39:11.454|                   |                   |                   |   SC :
1 |             ...
09:39:11.454|                   |                   |                   |   VC :
97829 |             ...
09:39:11.454|                   |                   |                   |   TC :
3 |             ...
09:39:11.454+-                 -+-                -+-                -|   VN :
1052 +-            ...
09:39:11.454|                   |                   |                   |  MGET
@@          |             ...
09:39:11.454|  CONT             |                   |
|                   |             ...
09:39:11.454|                   |                   |                   |  MPUT
NT          |             ...
09:39:11.457+-                 -+-                -+-                -|  PEND RE
CVARL1   +-            ...
09:39:11.462|                   |                   |                   |
WAIT             |             ...
09:39:11.463|          GTDA     |                   |
|                   |             ...
09:39:11.463|  INIT             |                   |
|                   |             ...
09:39:11.463|   SC :          1 +-                -+-                -+-
-+-            ...
09:39:11.463|   VC :      97925 |                   |
|                   |             ...
09:39:11.463|   TC :          1 |                   |
|                   |             ...
09:39:11.463|   VN :        149 |                   |
|                   |             ...
09:39:11.463|                   |                   |  CONT
|                   |             ...
09:39:11.463|  MGET @@          +-                -+-                -+-
-+-            ...
09:39:11.463|  FPUT NE GTDAA    |                   |
|                   |             ...
09:39:11.463|  MPUT NT          |                   |
|                   |             ...
09:39:11.465|  PEND FI          |                   |
|                   |             ...
09:39:12.471|  WAIT             +-                -+-                -+-
-+-            ...
09:39:12.472|                   |                   |          CVARL1
|                   |             ...
```

```
09:39:12.472|                  |                  |  INIT
|                  |            ...
09:39:12.472|                  |                  |   SC :          1
|                  |            ...
09:39:12.472+-                 -+-                -|   VC :      97829 +-
-+-            ...
09:39:12.472|                  |                  |   TC :          4
|                  |            ...
09:39:12.472|                  |                  |   VN :       1052
|                  |            ...
09:39:12.472|                  |                  |  MGET @@
|                  |            ...
09:39:12.472|  CONT            |                  |
|                  |            ...
09:39:12.472+-                 -+-                -|  MPUT NT          +-
-+-            ...
09:39:12.474|                  |                  |  PEND RE CVARL1
|                  |            ...
09:39:12.479|                  |                  |  WAIT
|                  |            ...
09:39:12.480|          GTDA     +-                -+-                -+-
-+-            ...
09:39:12.480|  INIT            |                  |
|                  |            ...
09:39:12.480|   SC :          1 |                  |
|                  |            ...
09:39:12.480|   VC :      97957 |                  |
|                  |            ...
09:39:12.480|   TC :          1 |                  |
|                  |            ...
09:39:12.480|   VN :        149 +-                -+-                -+-
-+-            ...
09:39:12.480|                  |                  |                  |
CONT             |             ...
09:39:12.480|  MGET @@         |                  |
|                  |            ...
09:39:12.480|  FPUT NE GTDAA   |                  |
|                  |            ...
09:39:12.480|  MPUT NT         |                  |
|                  |            ...
09:39:12.480|                  |                  |
|                  |            ...
09:39:12.482|  PEND FI         +-                -+-                -+-
-+-            ...
09:39:13.488|  WAIT            |                  |
|                  |            ...
```

12.2.4.11 TRACE2: TASK PERFORMANCE TRACE

Die Auswertungsliste TRACE2 listet die wichtigsten Ereignisse in den Teilprogrammen der Anwendung sequenziell auf. Da die Auswertung nicht wie in der Liste TRACE in Spalten für die UTM-Prozesse erfolgt, kann die Liste TRACE2 beliebig viele Prozesse anzeigen. TRACE2 enthält zusätzlich zu den Einträgen der Auswertung TRACE wichtige Daten zur Performanceanalyse.

Die Einträge in der Auswertung sind in zeitlicher Reihenfolge sortiert. Die Spalte TIME STAMP enthält den Zeitstempel des Ereignisses (Genauigkeit: Millisekunden).

Die Auswertungsliste TRACE2 erfasst folgende Ereignisse und Daten:

- Start eines Teilprogramms als Eintrag `strt >>> `*tac* mit
  - Transaktionscode des Teilprogramms
  - TAC-Klasse
  - aktueller I/O- und CPU-Stempel
  - Wartezeit des Auftrags in der TAC-Klasse
- Alle UTM-Funktionsaufrufe mit Operationscode und -modifikation und zusätzlich folgende Informationen:
  - KCMF bei Aufrufen, bei denen KCMF relevant ist
  - KCRN bei Aufrufen, bei denen KCRN relevant ist
  - KCLT bei den Aufrufen PADM und DADM
  - Bei PEND-Aufrufen mit KCOM = ER/FR/RS die Transaction-ID (SC,VC,TC und VN) für die Zuordnung zum abgebrochenen Vorgang.
  - Wenn KCRCCC != 0, die Returncodes KCRCCC und KCRCDC und die Transaction-ID (SC,VC,TC und VN).
- Ende des Teilprogramms als Eintrag `WAIT end<<<<` mit
  - CPU-Verbrauch im Teilprogramm in Mikrosekunden in Spalte „CPU“
  - I/O-Verbrauch im Teilprogramm in Spalte „I/0“
  - Am Ende der Spalte "TRACE" werden in vier Zeichen die Sedezimal-Werte der beiden ersten Bytes des Börsen-Announcements protokolliert, das diesen Börsen-Zyklus gestartet hat.

    Diese Information dient dem Systemdienst zur besseren Analyse und Diagnose von evtl.Performance-Problemen.

Die Strukturelemente `<<<<<<` in der Liste erleichtern die Lesbarkeit der Einträge.

Beispiel

```
TIME STAMP  | TRACE               | PID  |   SC       VC     TC     VN|KCRN    CPU  |KCMF    I/O  |TACCLASS Q.TIME   |
16:39:38.339+ INIT    -------------+  "   +---------------------------+-------------+-------------+------------------+-----------
16:39:38.339| MGET    -------------|  "   |                           |             |             |                  |
16:39:38.339| ADMI    -------------|  "   |                           |             |             |                  |
16:39:38.339| ADMI    -------------|  "   |                           |             |             |                  |
16:39:38.339+ MPUT NT -------------+  "   +---------------------------+-------------+-------------+------------------+-----------
16:39:38.339| MPUT NT -------------|  "   |                           |             |             |                  |
16:39:38.339| MPUT NT -------------|  "   |                           |             |             |                  |
16:39:38.339| MPUT NT -------------|  "   |                           |             |             |                  |
16:39:38.339+ MPUT NT -------------+  "   +---------------------------+-------------+-------------+------------------+-----------
```

```
16:39:38.339| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:38.339| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:38.339| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:38.339+ MPUT NT -------------+  "   +----------------------------+-------------+-------------+-----------------+-----------
16:39:38.339| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:38.340| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:38.340| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:38.340+ MPUT NT -------------+  "   +----------------------------+-------------+-------------+-----------------+-----------
16:39:38.340| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:38.340| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:38.340| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:38.340+ MPUT NT -------------+  "   +----------------------------+-------------+-------------+-----------------+-----------
16:39:38.340| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:38.340| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:38.340| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:38.340+ MPUT NT -------------+  "   +----------------------------+-------------+-------------+-----------------+-----------
16:39:38.340| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:38.340| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:38.340|                      |  "   |                            |             |             |                 |
16:39:38.341+ PEND FI -------------+  "   +----------------------------+-------------+-------------+-----------------+-----------
16:39:38.342| WAIT    -------------|  "   |                            |             |             |                 |
16:39:38.342| WAIT    end<<<<< 4545|  "   |<<<<<<<<<<<<<<<<<<<<<<<<<<<<|       +1071 |          +2 |<<<<<<<<<<<<<<<  |<<<<<<<<<<<
16:39:44.194| CONT    -------------|  "   |                            |             |             |                 |
16:39:44.194+ strt >>>palget       +  "   +    2       13      1      6+         102 +          15 + 0   *         0 +          0
16:39:44.194|                      |  "   |                            |             |             |                 |
16:39:44.198| INIT    -------------|  "   |                            |             |             |                 |
16:39:44.198| MGET    -------------| 7645 |    2       13      1      6|             |             |                 |
16:39:44.198| ADMI    -------------|  "   |                            |             |             |                 |
16:39:44.198| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:44.198+ MPUT NT -------------+  "   +----------------------------+-------------+-------------+-----------------+-----------
16:39:44.198| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:44.198| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:44.198| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:44.198+ MPUT NT -------------+  "   +----------------------------+-------------+-------------+-----------------+-----------
16:39:44.198| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:44.198| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:44.198| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:44.198+ MPUT NT -------------+  "   +----------------------------+-------------+-------------+-----------------+-----------
16:39:44.198| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:44.198| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:44.198| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:44.198+ MPUT NT -------------+  "   +----------------------------+-------------+-------------+-----------------+-----------
16:39:44.198| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:44.198| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:44.198| MPUT NT -------------|  "   |                            |             |             |                 |
16:39:44.198+ MPUT NT -------------+  "   +----------------------------+-------------+-------------+-----------------+-----------
16:39:44.198|                      |  "   |                            |             |             |                 |
16:39:44.206| PEND FI -------------|  "   |                            |             |             |                 |
```

```
16:39:44.207| WAIT    -------------|  "   |                           |             |             |                 |
16:39:44.207+ WAIT    end<<<<< 4545+  "   |<<<<<<<<<<<<<<<<<<<<<<<<<<<<|        +867 |          +2 |<<<<<<<<<<<<<<   |<<<<<<<<<<<
16:39:44.891| WAIT    end<<<<< F602|  "   |<<<<<<<<<<<<<<<<<<<<<<<<<<<<|         +17 |          +0 |<<<<<<<<<<<<<<   |<<<<<<<<<<<
16:39:44.891| WAIT    end<<<<< F0F7| 7628 |<<<<<<<<<<<<<<<<<<<<<<<<<<<<|         +17 |          +0 |<<<<<<<<<<<<<<   |<<<<<<<<<<<
16:39:54.878| WAIT    end<<<<< F602| 7640 |<<<<<<<<<<<<<<<<<<<<<<<<<<<<|          91 |           6 |<<<<<<<<<<<<<<   |<<<<<<<<<<<
16:39:54.879+ WAIT    end<<<<< 00F7+ 7644 |<<<<<<<<<<<<<<<<<<<<<<<<<<<<|          81 |           6 |<<<<<<<<<<<<<<   |<<<<<<<<<<<
16:39:54.901| WAIT    end<<<<< F602|  "   |<<<<<<<<<<<<<<<<<<<<<<<<<<<<|         +17 |          +0 |<<<<<<<<<<<<<<   |<<<<<<<<<<<
16:39:54.901| WAIT    end<<<<< F0F7| 7640 |<<<<<<<<<<<<<<<<<<<<<<<<<<<<|       +9473 |          +0 |<<<<<<<<<<<<<<   |<<<<<<<<<<<
16:39:57.154| CONT    -------------|  "   |                           |             |             |                 |
16:39:57.154+ strt >>>CSELEMP      +  "   +    2        14      1     6+         100 +           6 + 0   *          0 + 0
16:39:57.154|                     |  "   |                           |             |             |                 |
16:39:57.155| INIT    -------------|  "   |                           |             |             |                 |
16:39:57.155| MGET    -------------|  "   |                           |             |             |                 |
16:39:57.161+ MPUT NE -------------+  "   +---------------------------+-------------+-------------+-----------------+-----------
16:39:57.161|                     |  "   |                           |             |             |                 |
16:39:57.162| PEND FI -------------|  "   |                           |             |             |                 |
```

13 Lastsimulation mit Workload Capture and Replay

Mit der Funktion Workload Capture & Replay kann die Kommunikation von UTM-Anwendungen mit UPIC-Clients mitgeschnitten und anschließend mit einstellbaren Lastprofilen abgespielt werden. Damit lässt sich das Verhalten der UTM-Anwendung bei hoher Last unter realen Bedingungen testen.

Workload Capture & Replay besteht aus folgenden Komponenten:

- *UPIC Capture*: schneidet die Kommunikation mit dem UPIC-Client mit.

  Zum Mitschneiden von UPIC-Sessions (Capture) wird die Trace-Funktion BTRACE (BCAM-Trace) verwendet, die auf allen Server-Plattformen vorhanden ist.

  Die Traces müssen ggf. noch zusammengemischt werden.
- *UPIC Analyzer*: dient zur Analyse der mitgeschnittenen Kommunikation.

  Zur Analyse wird das Programm *UPICAnalyzer* verwendet, das mit UPIC auf 64-Bit-Linux-Systemen ausgeliefert wird.
- *UPIC Replay*: dient zum Abspielen der mitgeschnittenen UPIC-Session mit unterschiedlichen Lastparametern (Geschwindigkeit, Client-Anzahl).

  Dazu wird das Programm *UPICReplay* verwendet, das mit UPIC auf 64-Bit-Linux-Systemen ausgeliefert wird.

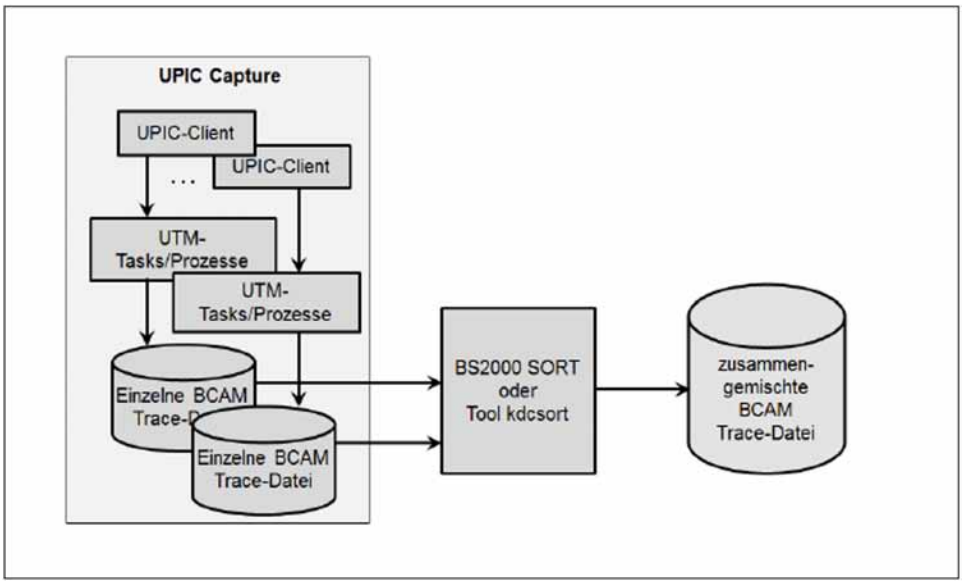
> ¡ Das *UpicAnalyzer* Programm muss zu der UTM-Version kompatibel sein, die beim Capture-Vorgang verwendet wurde. Z.B. ist „openUTM-Client V7.0 für Trägersystem UPIC“ kompatibel zu openUTM V7.0.
>
> Die Version des *UpicReplay* Programms kann nur Eingabedateien verarbeiten, die mit der gleichen Version des *UpicAnalyzer* Programms erstellt wurden.

Zusätzlich wird auf Unix-, Linux- und Windows-Systemen das Dienstprogramm *kdcsort* ausgeliefert, um die Mitschnitte zu sortieren, wenn die Kommunikation im Mehr-Prozess-Betrieb stattfand.

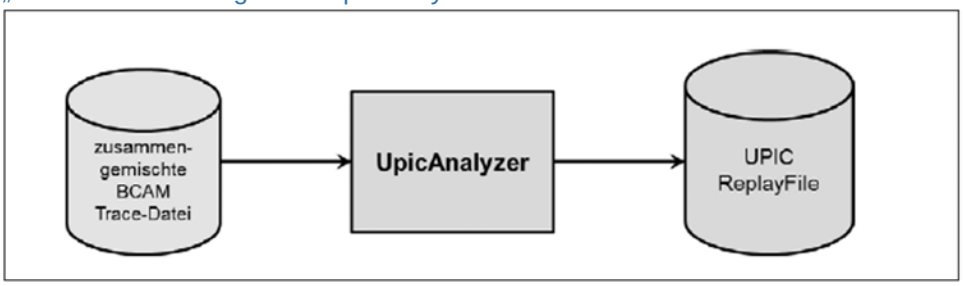
Die Funktion Workload Capture & Replay führen Sie in folgenden Schritten durch:

1. Schalten Sie den BCAM-Trace ein und starten Sie die UPIC-Kommunikation, siehe Abschnitt „UPIC-Conversation mitschneiden (UPIC Capture)“.
2. Beenden Sie den BCAM-Trace und mischen Sie die BCAM-Trace-Einträge in eine Trace-Datei zusammen (falls nötig), siehe Abschnitt „Trace-Einträge zusammenmischen".

   Beide Schritte werden in folgendem Bild veranschaulicht.

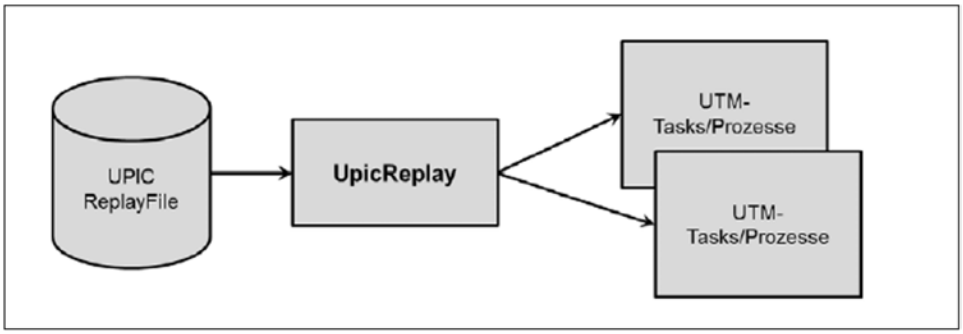


3. Übertragen Sie die Trace-Datei binär auf das 64-Bit Linux-System, auf dem Sie UPIC installiert haben. Bitte beachten Sie, dass Sie die Datei mit openFT übertragen müssen, wenn die Datei von einem BS2000-System übertragen wird.
4. Erzeugen Sie auf dem 64-Bit Linux-System mit installierten UPIC-Client eine UPIC ReplayFile. Dazu rufen Sie das Programm *UpicAnalyzer* auf mit der Trace-Datei als Eingabedatei, siehe Bild. Details siehe Abschnitt „Daten mit dem Programm UpicAnalyzer aufbereiten“.



5. Starten Sie das Programm

   UpicReplay mit UPIC ReplayFile als Eingabedatei, siehe Bild. Details siehe Abschnitt „UPIC-Session mit dem Programm UpicReplay abspielen“.



13.1 UPIC-Conversation mitschneiden (UPIC Capture)

Bei diesem Schritt kann die UTM-Anwendung auf einer beliebigen UTM-Plattform ablaufen (BS2000-, Unix-, Linux- oder Windows-System).

Die UPIC-Clients können auf jeder beliebigen UPIC-Plattform ablaufen, auch UPIC-Clients auf Basis des Produkts openUTM-JConnect werden unterstützt.

Während dieser Phase muss die Kommunikation der UTM-Anwendung mit UPIC-Clients vollständig aufgezeichnet werden, wobei die Trace-Länge größer als die maximale Nachrichtenlänge sein muss. Hierfür wird die UTM-Funktion BCAM-Trace verwendet.

> ¡ Wenn sowohl die UPIC-Clients als auch die UTM-Anwendung Verschlüsselung unterstützen, können UPIC-Conversations für Benutzer mit Passwort nicht von *UpicReplay* abgespielt werden. Dasselbe gilt für UPIC-Conversations mit Verschlüsselung von Benutzernachrichten.
>
> Außerdem ignoriert *UpicReplay* den Aufruf des UPIC-Clients zum Setzen eines neuen Passworts.
>
> Auf BS2000-Systemen unterstützen sowohl UPIC-Clients als auch UTM-Anwendungen Verschlüsselung.
>
> Auf Unix-, Linux- und Windows-Systemen unterstützen UPIC-Clients und UTM-Anwendungen nur dann Verschlüsselung wenn eine passende openSSL Bibliothek zur Verfügung gestellt wurde.

Bitte beachten Sie, dass die ausgesuchten UTM-Vorgänge auch beliebig oft wiederholbar sein müssen.

Dazu gehen Sie wie folgt vor:

1. Starten Sie den BCAM-Trace per Startparameter BTRACE=ON,*length*, siehe "Startparameter für openUTM". Es wird empfohlen, für */length* den Maximalwert anzugeben, damit Nachrichten nicht abgeschnitten werden. Sie können den BCAM-Trace auch per Administration einschalten (Kommando KDCDIAG oder über WinAdmin /WebAdmin). In diesem Fall wird aber für */length* der Standardwert (256 Bytes) angenommen, wenn Sie im Startparameter BTRACE keine andere Länge angegeben haben.
2. Führen Sie die für die Lastsimulation benötigten UPIC-Conversations zwischen dem UPIC-Client und der UTM-Anwendung aus. Dazu gehört auch der vollständige Verbindungsaufbau der UPIC-Clients. Die zugehörigen UTM-Vorgänge müssen mindestens einmal komplett durchlaufen werden.
3. Beenden Sie den BCAM-Trace per Kommando KDCDIAG oder über WinAdmin/WebAdmin.

Das Ergebnis dieses Schrittes sind binäre BTRACE-Dateien von allen UTM-Prozessen. Details zu BTRACE-Dateien finden Sie im openUTM-Handbuch „Meldungen, Test und Diagnose“.

13.2 Trace-Einträge zusammenmischen

Dieser Schritt ist nötig, falls die UTM-Anwendung beim Mitschneiden mit mehr als einem Prozess gelaufen ist, was bei produktiven UTM-Anwendungen mit mittlerer oder hoher Last in der Regel der Fall ist.

Die binären BTRACE-Dateien von allen UTM-Prozessen werden in diesem Schritt auf Basis der Zeitstempel in eine gemeinsame BTRACE-Datei einsortiert. Dieser Prozess-Schritt muss immer auf der gleichen Plattform wie Schritt 1 (UPIC Capture) ablaufen.

Auf Unix-, Linux- und Windows-Systemen müssen Sie für diesen Schritt das UTM-Dienstprogramm *kdcsort* verwenden, siehe unten.

Das Ergebnis dieses Schrittes ist eine sortierte binäre BTRACE-Datei, die alle Trace-Einträge in der zeitlich korrekten Reihenfolge enthält.

Sie können zum Sortieren die ausgelieferte Beispiel-Prozedur BTRACE verwenden.

Dienstprogramm kdcsort

Das Dienstprogramm *kdcsort* liest die Trace-Einträge aus mehreren BTRACE-Dateien ein und schreibt die Trace-Records zeitlich sortiert in eine Ausgabedatei. Es wird wie folgt gestartet:

Auf Unix- und Linux-Systemen aus der Shell mit

utmpfad`/ex/kdcsort` *btrace_out btrace-1 btrace-2 ... btrace-n*

Auf Windows-Systemen in einem Eingabeaufforderungs-Fenster mit

utmpfad`\ex\kdcsort` *btrace_out btrace-1 btrace-2 ... btrace-n*

Bedeutung der Parameter:

btrace_out

> Name der Ausgabe-Datei, in die die sortierten Trace-Records geschrieben werden sollen.

btrace-1 btrace-2 ... btrace-n

> Namen der BTRACE-Dateien, die aufgezeichnet wurden. Es müssen mindestens zwei Dateien angegeben werden.

Die Dateinamen müssen durch Leerzeichen getrennt angegeben werden.

Die Ausgabe-Datei von *kdcsort* kann anschließend entweder mit dem UTM-Dienstprogramm *kdcbtrc* als gemeinsame Liste für alle UTM-Workprozesse aufbereitet werden oder mit dem Programm *UpicAnalyzer* weiter verarbeitet werden.

13.3 Daten mit dem Programm UpicAnalyzer aufbereiten

Das Programm *UpicAnalyzer* wird mit UPIC auf Linux (64 Bit) ausgeliefert. *UpicAnalyzer* liest die Trace-Records aus einer BTRACE-Trace, filtert die UPIC-Trace-Records aus, bereitet diese auf und schreibt sie in einem bestimmten Format (UPIC ReplayFile Layout) in eine Datei. Diese Datei kann dann als Eingabedatei für das Programm *UpicReplay* verwendet werden.

UpicAnalyzer wird wie folgt aus der Linux-Shell aufgerufen:

`UpicAnalyzer` *inputfile outputfile*

Bedeutung der Parameter:

inputfile

Name der BTRACE-Datei, die Sie auf das Linux-System übertragen haben.

outputfile

Name der Ausgabedatei (UPIC ReplayFile). Diese Datei können Sie verwenden, um die UPIC-Session mit Hilfe von *UpicReplay* ablaufen zu lassen.

Das Programm *UpicAnalyzer* erkennt den Plattform-Typ, auf dem die Trace-Datei erstellt wurde, und verarbeitet den Inhalt entsprechend der plattform-spezifischen Besonderheiten.

Beispiel

Die übertragene Trace-Datei hat den Namen *btrc.sorted*. Sie soll aufbereitet und die Ausgabe in die Datei *Replayfile* geschrieben werden. Der Aufruf lautet:

```
UpicAnalyzer btrc.sorted Replayfile
```

Ausgaben:

```
Program "UpicAnalyzer": Version 7.0 build yyyy-mm-dd on Linux Intel   ,64 Bit ,
Little-Endian started

inputfile "btrc.sorted"

outputfile "Replayfile"

109 UTM BCAM trace records with 17218 bytes read.

25 UPIC replay records with 2046 bytes written.

Program "UpicAnalyzer" finished.
```

13.4 UPIC-Session mit dem Programm UpicReplay abspielen

Das Programm *UpicReplay* ist ein UPIC-Client-Programm, das mit UPIC auf Linux (64 Bit) ausgeliefert wird. Vor dem Abspielen müssen Sie ggf. die UPIC-Konfiguration und/oder die Generierung der UTM-Anwendung anpassen.

Beim Abspielen sollte die gleiche UTM-Plattform verwendet werden wie beim Mitschneiden. Ausnahmen sind möglich, siehe „Unterschiedliche Plattformen für Capture and Replay“.

13.4.1 UPIC-Konfiguration und UTM-Generierung anpassen

Für den Ablauf auf dem Linux-System wird die Side Information-Datei *upicfile* benötigt, in der mindestens ein Eintrag mit dem Namen UPREPLAY zu finden ist. Der Eintrag muss das Präfix SD oder ND haben, Ausnahme siehe „Unterschiedliche Plattformen für Capture and Replay“.

Dieser Eintrag muss ein gültiger Eintrag mit dem TAC eines Services der UTM-Anwendung sein. (z.B. "DEMO"). Dieser Eintrag wird vom Programm *UpicReplay* zur Adressierung der UTM-Anwendung verwendet. Der TAC wird ggf. vom Programm *UpicReplay* passend mit Daten aus dem Replay File gesetzt.

Beispiel für einen upicfile-Eintrag

Replay mit dem TAC DEMO. Die UTM-Anwendung UTMTEST1 läuft auf dem Rechner HOST5678.

```
SDUPREPLAY UTMTEST1.HOST5678 DEMO LISTENER-PORT=11111 T-TSEL-Format=T
```

UTMTEST1 muss entweder in MAX APPLINAME oder in einer BCAMAPPL-Anweisung generiert sein.

Hinweise zur UTM-Generierung

Die UTM-Anwendung muss beim UPIC Replay Schritt, insbesondere bei erhöhter Last, möglicherweise mehr UPIC-Verbindungen vom Programm *UpicReplay* zulassen als während des Mitschneidens ursprünglich vorhanden waren. Daher wird empfohlen, für den UPIC-Zugang einen ausreichend dimensionierten UPIC Terminal-Pool mit Multi-Connect Funktionalität zu verwenden, z.B.:

```
TPOOL LTERM=REPL,PTYPE=UPIC-R,CONNECT=MULTI,NUMBER=1000
```

In diesem Fall können sich bis zu 1000 UPIC-Clients gleichzeitig über den Terminal-Pool anmelden.

Wenn der UPIC Replay Schritt mit erhöhter Last abläuft, dann müssen lastabhängige Generierungsparameter ggf. vergrößert werden. Insbesondere müssen Sie auf Folgendes achten:

- Ausreichende Größe des UTM-Cache (MAX CACHESIZE)
- Ausreichende Größe des Page Pool (MAX PGPOOL)
- Ausreichende Anzahl der UTM-Prozesse (MAX TASKS)
- Ausreichende Anzahl der zugelassenen Concurrent User (MAX CONN-USERS)

Unterschiedliche Plattformen für Capture and Replay

Beim Abspielen werden die Daten 1:1 an die UTM-Anwendung übergeben. Wenn die Daten z.B. hardware-abhängige Binärdaten enthalten, dann führt dies beim Plattformwechsel zu Fehlern. Daher gilt Folgendes:

- Es ist nicht möglich, eine Session mit einer UTM-Anwendung auf BS2000-Systemen mitzuschneiden und später mit einer UTM-Anwendung auf Unix-, Linux-, oder Windows-Systemen abzuspielen. Der Grund: Die Daten liegen in der Trace-Datei in EBCDIC vor, eine Umcodierung nach ASCII wird in UPIC nicht unterstützt.
- Ein Wechsel zwischen einer 32-Bit- und einer 64-Bit-Plattform ist nicht möglich, auch nicht innerhalb einer Plattformfamilie.
- Es ist möglich, eine Session mit einer UTM-Anwendung auf Unix-, Linux-, oder Windows-Systemen mitzuschneiden und später mit einer UTM-Anwendung auf BS2000-Systemen abzuspielen. Voraussetzung ist, dass in der Session nur reine ASCII-Textdaten übertragen werden.

  In diesem Fall müssen Sie in der Datei *upicfile* HD als Präfix angeben, damit die Daten korrekt zwischen ASCII und EBCDIC umcodiert werden.

13.4.2 Aufruf von UpicReplay

UpicReplay spielt die aufgezeichneten UPIC-Conversations erneut ab, siehe Abschnitt „Arbeitsweise von UpicReplay“. Protokoll-Meldungen und Warnungen werden dabei nach *stdout* und Debug- oder Fehler-Meldungen nach *stderr* ausgegeben.

UpicReplay wird wie folgt aus einer Linux-Shell aufgerufen:

`UpicReplay` *InputFileName* [`-c`<*numberOfClients*>]
[`-s`<*speedPercentage*>] [`-d`[d]]

Bedeutung der Parameter

InputFileName

Name des UPIC ReplayFile, das Sie mit dem UpicAnalyzer erzeugt haben.Pflichtparameter.

c<numberOfClients>

numberOfClients gibt die Anzahl der UPIC-Clients an, für die die aufgezeichneten Conversations abgespielt werden sollen.

Standard: 1, (entspricht *-c1*) d.h. es wird nur ein Client simuliert.
Das effektive Limit hängt von den jeweiligen System-Limits ab

-s<speedPercentage>

speedPercentage gibt die Abspielgeschwindigkeit in Prozent im Vergleich zur Originalgeschwindigkeit an. Damit lassen sich lange und kurze Denkzeiten simulieren.

Standard: 100 (entspricht *-s100)* d.h. Originalgeschwindigkeit

-s200 bedeutet 200%, d.h. doppelte Geschwindigkeit, realisiert durch halbe Denkzeiten.

-d aktiviert Debug-Ausgaben auf *stderr*, d.h. Ausgabe von Debug-Meldungen bei Thread-Erzeugung sowie wenige Meldungen bei Send- und Receive-Aufrufen.

-dd aktiviert erweiterte Debug-Ausgaben auf *stderr*, d.h. Ausgabe von detaillierten Debug-Meldungen. Diese Option ist nur für die interne Diagnose von *UpicReplay* gedacht.

-dd ist nur sinnvoll bei Simulation einer kleinen Anzahl von Clients.

Standard: keine Debug-Ausgaben.

Beispiel

Die in der Datei *Replay.1239* aufgezeichneten UPIC Conversations sollen mit normaler Geschwindigkeit für 100 Clients abgespielt werden. Der Aufruf lautet:

```
UpicReplay Replay.1239 -c100
```

13.4.3 Arbeitsweise von UpicReplay

UpicReplay spielt die Kommunikation möglichst 1:1 wie beim aufgezeichneten Ablauf nach:

- Für jedes UPIC PTERM/LTERM, für das in dem UPIC ReplayFile ein Trace-Record gefunden wird, wird ein UPIC-Thread erzeugt, der die jeweilige UPIC-Conversation dieses UPIC-Clients nachspielt.
- Dieser UPIC-Thread schickt in Schleife alle Eingabe-Nachrichten in gleicher Weise an den UTM-Vorgang wie beim Mitschneiden, d.h. mit dem gleichen Daten-Inhalt und Kontrollfluss. Analoges gilt für das Holen der Ausgabe-Nachrichten von der UTM-Anwendung, wobei die Ausgabe-Nachrichten inhaltlich nicht geprüft werden.

Probleme beim Replay

In den folgenden Fällen kommt es zu Abweichungen zwischen dem Mitschnitt und der Wiederholung beim Replay:

- Unvollständiger Mitschnitt

  Eine UPIC-Conversation (d.h. ein UTM-Vorgang) wurde begonnen, bevor das Mitschneiden per BCAM-Trace eingeschaltet wurde.

  Es wird eine entsprechende Meldung ausgegeben und alle Eingabe-Nachrichten aus dieser begonnenen Conversation dieses Clients werden verworfen.

  Der UPIC-Thread sucht dann im Mitschnitt nach dem Beginn einer neuen Conversation für diesen UPIC-Client:

  - Wird eine neue Conversation in den aufgezeichneten Records für dieses PTERM/LTERM gefunden, dann wartet dieser Client zunächst noch entsprechend den aufgezeichneten Zeitstempeln und beginnt dann erst ab dieser Stelle die Last-Simulation.
  - Wird keine neue Conversation gefunden, wird dieser UPIC Replay Thread ohne Kommunikation mit der UTM-Anwendung beendet.
- Verkürzter Vorgang

  Ein UTM-Vorgang wird beim Nachspielen nach weniger Kommunikations-Schritten als beim Mitschneiden von der UTM-Anwendung beendet (Normal oder Abnormal). Dieser Fall kann auftreten:

  - wenn das Anwendungsprogramm die mitgeschnittenen Eingabe-Daten nicht korrekt verarbeiten kann, weil z. B. Zeitangaben in der Eingabe-Nachricht stehen, die vom Programm als "verspätet" abgelehnt werden. Der UTM-Vorgang wird deshalb vorzeitig beendet.
  - wenn beim Nachspielen eine nicht zugelassene UTM-Zugangsberechtigung verwendet wird, z.B. fehlende UTM-Administrationsberechtigung.

  In diesem Fall wird eine entsprechende Meldung ausgegeben und der UPIC Replay Thread verwirft weitere Nachrichten, bis er wieder einen neuen Conversation-Beginn von diesem Client im Mitschnitt findet. An diesem Conversation-Beginn setzt der UPIC-Thread nach einer entsprechenden Verzögerungszeit neu auf oder er beendet sich, wenn keine weitere Conversation für diesen UPIC-Client aufgezeichnet wurde.
- Überlanger Vorgang

  Ein UTM-Vorgang hat beim Nachspielen mehr Kommunikations-Schritte als beim Mitschneiden.

  Wegen nicht mitgeschriebener Eingabe-Daten beendet der UPIC-Thread diesen Vorgang abnormal durch Verbindungs-Abbau. Zusätzlich wird eine spezifische Warnmeldung erzeugt.

  Anschließend wird der Beginn der nächsten Conversation von diesem Client im Mitschnitt gesucht. An diesem Conversation-Beginn setzt der UPIC-Thread nach einer entsprechenden Verzögerungszeit neu auf oder er beendet sich, wenn keine weitere Conversation für diesen UPIC-Client aufgezeichnet wurde.

- Eingabe-Nachricht unvollständig

  Eine Eingabe-Nachricht konnte wegen Längenbeschränkung des Trace-Records trotz Kompressionsversuchs beim Mitschneiden nicht vollständig mitgeschrieben werden.

  Der Record wird mit Warnmeldung verworfen und es wird der Beginn der nächsten Conversation von diesem Client im Mitschnitt gesucht.

  An diesem Conversation-Beginn setzt der UPIC-Thread nach einer entsprechenden Verzögerungszeit neu auf oder er beendet sich, wenn keine weitere Conversation für diesen UPIC-Client aufgezeichnet wurde.

- Sonstiger Fehler

  Es wird ein sonstiger, unerwarteter Return Code an der UPIC-Programmschnittstelle gemeldet, der nicht in den obigen Fällen enthalten ist.

  Diese Situation kann z.B. dann eintreten, wenn die UTM-Anwendung entweder nicht erreichbar ist oder einen Verbindungsaufbau ablehnt.

  In diesen Fällen wird vom UPIC-Thread eine Fehlermeldung ausgegeben.

  Der betroffene UPIC-Thread wird beendet, ohne nach neuen Conversations für diesen Client im Mitschnitt zu suchen. Alle von dem Problem nicht direkt betroffenen anderen UPIC-Conversations laufen unbeeinflusst weiter.

14 Anhang

14.1 openUTM installieren auf Unix- und Linux-Systemen

Bevor Sie UTM-Anwendungen auf Ihrem System erzeugen können und zum Ablauf bringen können, müssen Sie openUTM selbst im System installieren.

Unter „openUTM“ versteht man die UTM-Systemfunktionen (Systemcode), C-Includes und COBOL-Copys zum Erzeugen des Anschlussprogramms, Programme für die Dialog-Terminalprozesse, die Druckprozesse, die Timerprozesse (Zeitsteuerung), die Netzprozesse sowie die Tools zum Erstellen, Betreiben und Verändern von UTM-Anwendungen.

> ℹ Wenn Sie mehrere Festplatten an Ihrem Rechner betreiben, sollten Sie aus Performancegründen die UTM-Anwendung und das Datenbanksystem auf verschiedene Platten legen.

Bei der Installation wird außerdem der C++-Anschlussmodul übersetzt.

14.1.1 UTM-Systemfunktionen auf Unix- und Linux-Systemen installieren

Wie Sie openUTM auf Ihrem System installieren müssen, ist abhängig vom Betriebssystem. Näheres zur Installation finden Sie in der Freigabemitteilung und der Lieferinformation. In der Freigabemitteilung sind auch die Versionsabhängigkeiten zu Produkten aufgeführt, mit denen openUTM zusammenarbeitet. In der Lieferinformation stehen Kommandos zum Installieren.

utmpfad

Das Verzeichnis mit den für den Ablauf von openUTM notwendigen Dateien wird in diesem Handbuch als **utmpfad** bezeichnet.

Da openUTM auf einer Plattform sowohl im 32-Bit- als auch im 64-Bit-Modus ausgeliefert wird, werden bei der Installation zwei Verzeichnisbäume eingerichtet:

utm-installationsverzeichnis`/utm70a00/32` für den 32-Bit-Modus

und

utm-installationsverzeichnis`/utm70a00/64` für den 64-Bit-Modus

utm-installationsverzeichnis ist das Verzeichnis, das bei der Installation angegeben wurde. `utm70a00` bezeichnet die aktuelle Version. Diese kann sich mit einer neuen Korrekturstufe ändern. Beachten Sie daher die Freigabemitteilung.

Um einen korrekten Ablauf von openUTM zu garantieren, müssen Sie die Umgebungsvariable UTMPATH auf den Wert von *utmpfad* setzen, siehe auch Abschnitt„Starten einer UTM-Anwendung auf Unix- und Linux-Systemen“.

Beispiel

openUTM wird unter `/opt/lib` installiert

- Sie wollen openUTM für den Ablauf im 32-Bit-Modus verwenden:
  Setzen Sie UTMPATH auf den Wert `/opt/lib/utm70a00/32`
- Sie wollen openUTM für den Ablauf im 64-Bit-Modus verwenden:
  Setzen Sie UTMPATH auf den Wert `/opt/lib/utm70a00/64`

14.1.2 Unterschiedliche Socket-Netzprozesse einsetzen (Unix- und Linux- Systeme)

Bei der Installation von openUTM werden in *utmpfad*`/ex` zwei Typen von Socket-Netzprozessen bereitgestellt.

- *utmnets* bzw. *utmnetsnnnn*: Socket-Netzprozesse zur Kommunikation ohne Verschlüsselung
- *utmnetssl*: Socket-Netzprozess mit TLS Verschlüsselung

Socket-Netzprozesse ohne Verschlüsselung

Von den Netzprozessen des ersten Typs werden mehrere Varianten bereitgestellt. Sie unterscheiden sich im wesentlichen durch die maximale Anzahl der Socket-Verbindungen, die parallel aktiv sein können. Die jeweilige maximale Anzahl ist im Dateinamen enthalten, z.B. ist *utmnets1024* der Socket-Netzprozess für bis zu 1024 Socket-Verbindungen.

Nach der Installation ist standardmäßig der Prozess für bis zu 1024 parallele Verbindungen im Einsatz. Der aktuell aktive Socket-Netzprozess hat immer den Namen *utmnets*.

Wenn Sie einen anderen Socket-Netzprozess einsetzen möchten, z.B. *utmnets2000* für bis zu 2000 Socket-Verbindungen, dann gehen Sie wie folgt vor:

1. Beenden Sie die UTM-Anwendung
2. Kopieren Sie *utmpfad*`/ex/utmnets2000` nach *utmpfad*`/ex/utmnets.`
   Dazu benötigen Sie root-Berechtigung.
3. Starten Sie die UTM-Anwendung wieder

Socket-Netzprozesse mit TLS Verschlüsselung

Vom TLS Netzprozess wird nur eine Variante bereitgestellt. Pro utmnetssl Prozess können maximal 1024 Socket-Verbindungen parallel aktiv sein.

14.1.3 openSM2-Anschluss installieren (Unix- und Linux-Systeme)

Bei der Installation von openUTM wird automatisch auch der Anschluss an openSM2 installiert. Dabei werden folgende Aktionen durchgeführt:

- Unter dem Verzeichnis *utmpfad*`/shsc` wird das Skript `utmsm2` abgelegt, das von openSM2 verwendet wird, um auf die Messdaten der UTM-Anwendungen zugreifen zu können.
- Im Anschluss daran wird das Skript `utmsm2` auf Solaris- und Linux-Systemen nach `/opt/bin` kopiert.
- Zusätzlich werden in der Datei `/opt/bin/utmsm2.dat` die beiden UTM-Pfade eingetragen, die durch diese Installation entstanden sind, also *utm-installationsverzeichnis*`/utm70a00/32` und *utm-installationsverzeichnis* `/utm70a00/64`. Wenn die Datei `/opt/bin/utmsm2.dat` noch nicht existiert, wird sie vorher angelegt.

Bei einer Deinstallation von openUTM werden diese beiden Pfade aus der Datei `/opt/bin/utmsm2.dat` wieder ausgetragen. Sollte danach kein Eintrag mehr in der Datei vorhanden sein, werden die Dateien `/opt/bin/utmsm2` und `/opt/bin/utmsm2.dat` gelöscht.

14.2 openUTM installieren auf Windows-Systemen

Bevor Sie UTM-Anwendungen an Ihrem System erzeugen können und UTM-Anwendungen ablaufen können, müssen Sie openUTM selbst im System installieren.

Unter „openUTM“ versteht man die UTM-Systemfunktionen (Systemcode), C-Includes zum Erzeugen des Anschlussprogramms, Programme für die Dialog-Terminalprozesse, die Timerprozesse (Zeitsteuerung), die Netzprozesse sowie die Tools zum Erstellen, Betreiben und Verändern von UTM-Anwendungen.

Die Hardware- und Software-Voraussetzungen entnehmen Sie bitte der Freigabemitteilung.

14.2.1 Installation des openUTM-Servers (Windows-Systeme)

Auf der openUTM-Installations-DVD befindet sich für die 64-Bit-Umgebung ein eigenes Installationsmedium.

> **¡** Hinweis zur Kommunikationskomponente PCMX:
>
> Auf der openUTM-Installations-DVD befindet sich für die 64-Bit-Umgebung ein eigenes PCMX Installationsmedium - PCMX-64. Bitte sorgen Sie dafür dass auf Ihrem System auch die Kommunikationskomponente PCMX installiert ist.

Die Installation der UTM-Systemfunktionen kann nur unter einer Kennung mit Administrator-Rechten vorgenommen werden. ES sind folgende Schritte notwendig:

1. Starten Sie das Programm `utm-64.msi` auf der openUTM-Installations-DVD:
   - durch Doppelklick im Windows-Explorer
   - oder in der Windows-Eingabeaufforderung mit dem Kommando
     `msiexec /i utm-64.msi`
2. Wählen Sie die zu installierenden Produkte aus und installieren Sie - wenn nicht schon vorhanden - PCMX. PCMX wird auch von anderen Produkten verwendet und kann deshalb bereits in einer anderen Korrekturstufe an Ihrem Rechner vorhanden und für bestimmte Netzverbindungen vorkonfiguriert sein. Es wird jedoch empfohlen, die neueste Version von PCMX zu verwenden.
3. Folgen Sie den weiteren Anweisungen des Installationsprogramms und wählen Sie die passenden Optionen.

   openUTM überprüft die Systemvoraussetzungen und die Verfügbarkeit von ausreichender Plattenspeicherkapazität. Falls die Überprüfung negativ ausfällt, wird die Installation abgelehnt.

   Falls in dem Zielverzeichnis auf dem Windows-Rechner bereits eine Version von openUTM vorhanden ist, dann werden Sie von der Installationsprozedur gefragt, ob eine vorhandene Installation deinstalliert oder überschrieben werden soll. Es wird empfohlen, die Vorgängerversion zu deinstallieren.
4. Entnehmen Sie nach der Installation die DVD aus dem Laufwerk und booten sie das System neu.

Wenn Sie auf Ihrem Windows-Rechner außerdem UTM-Teilprogramme übersetzen oder die UTM-Anwendung binden möchten, dann m üssen Sie sicherstellen, dass Microsoft Visual Studio installiert ist.

utmpfad

Das Verzeichnis mit den für den Ablauf von openUTM notwendigen Dateien wird in diesem Handbuch als *utmpfad* bezeichnet.

Bei der Installation auf Windows-Systemen wird das *utm-installationsverzeichnis* als *utmpfad* eingerichtet. *utm-installationsverzeichnis* ist das Verzeichnis, das bei der Installation angegeben wurde.

Um einen korrekten Ablauf von openUTM zu garantieren, müssen Sie die Umgebungsvariable UTMPATH auf den Wert von *utmpfad* setzen, siehe auch Abschnitt „Starten einer UTM-Anwendung auf Windows-Systemen“.

14.2.2 Benutzerumgebung (Windows-Systeme)

Für das Entwickeln von Anwendungen sind keine Administratorrechte erforderlich.

Einige Tools (z.B. KDCDEF) müssen in einem Eingabeaufforderungs-Fenster aufgerufen werden. Um diese trotzdem per Mausklick starten zu können, können Shortcuts eingerichtet werden Auf "Starten mit utmmain (Windows-Systeme)" finden Sie ein Beispiel für das Starten einer UTM-Anwendung über Shortcut; weitere Shortcut-Beispiele siehe Quick Start Kit. Mehr Informationen zu Shortcuts finden Sie in der Windows-Dokumentation.

14.3 Struktur des UTM-Installationsverzeichnisses

openUTM legt bei der Installation folgende Dateien im *utmpfad* an:

Datei	Beschreibung
Unix-, Linux- und Windows-Systeme	
applifile	Allgemeine Kontrolldatei für alle UTM-Anwendungen
msgdescription	Meldungsdefinitionsdatei
mtxtin	Eingabedatei für das Programm KDCMTXT
Unix- und Linux-Systeme	
CPIO.utmsample	Beispielanwendung
utm.log	Protokolldatei der Installation bei Unix- und Linux-Systemen
Windows-Systeme	
AddFirewallEntries.cmd	Kommando-Datei zum Eintragen der *utmnet*/*utmnets*/*utmnetss*/ Prozesse in der Firewall des Systems, wird während der Installation von UTM aufgerufen.
applifile.bak	Kopie der APPLIFILE
RemoveFirewallEntries.cmd	Kommando-Datei zum Entfernen der utmnet/utmnets/utmnetssl Prozesse aus der Firewall des Systems, wird bei der Deinstallation von UTM aufgerufen.
UnInstall.cmd	Datei zur Deinstallation bei Windows-Systemen

openUTM legt bei der Installation folgende Verzeichnisse im *utmpfad* an:

Verzeichnis	Beschreibung
Unix-, Linux- und Windows-Systeme	
copy-cobol85	Copy Elemente für COBOL (Micro Focus Compiler)
cpic	CPI-C Schnittstelle
diaglst	Datenstrukturen für die Diagnose
ex	Tools und Programme
excel	Excel-Makro für KDCEVAL
include	Header-Files
netcobol	Copy Elemente für COBOL (NetCOBOL Compiler)
nls	National Language Support, Meldungskataloge

sample	openUTM Beispieldateien
shsc	Verzeichnis mit UTM-Skripten und -Prozeduren
src	Quellcode für Änderungen bzw. Debugging
sys	Objektbibliotheken und Objektmodule
tx	TX-Schnittstelle
upicl	UPIC-L-Verzeichnis
xatmi	XATMI für UPIC-L-Clients
Unix- und Linux-Systeme	
oss	Produkt OSS
Windows-Systeme	
log	Protokoll-Dateien

14.4 Umgebungsvariablen einer UTM-Anwendung

In diesem Abschnitt werden alle Umgebungsvariablen aufgelistet, die für die Steuerung von openUTM von Bedeutung sind. Sie sind aufgeteilt in folgende Gruppen:

- Allgemeine Umgebungsvariablen für openUTM
- Umgebungsvariablen für Workprozesse
- Umgebungsvariable für die Nutzung der openSSL Bibliothek
- Umgebungsvariable für das Tool KDCDUMP
- Umgebungsvariablen für die X/Open Schnittstelle XATMI
- Zusätzliche Umgebungsvariablen für openUTM auf Unix- und Linux- Systemen
- Zusätzliche Umgebungsvariablen für openUTM auf Windows- Systemen

Zu jeder Umgebungsvariable werden Bedeutung, Wertebereich, Standardeinstellung und der UTM-Prozess angegeben, der die Umgebungsvariable auswertet.

Alle Umgebungsvariablen müssen vor dem Start der UTM-Anwendung gesetzt werden.

> **i** Nach der Deinstallation von openUTM auf Windows-Systemen müssen Sie die Umgebungsvariablen UTMPATH und PATH prüfen und eventuell bereinigen.

14.4.1 Allgemeine Umgebungsvariablen für openUTM

UTMPATH

Bedeutung
Dateiverzeichnis, in dem alle Bestandteile von openUTM sowie die Datei `applifile` stehen. Diese Umgebungsvariable ist zwingend für den Betrieb von openUTM anzugeben.

Wertebereich
Dateiverzeichnis, unter dem die für den Ablauf von openUTM notwendigen Dateien stehen (*utmpfad*, siehe "UTM-Systemfunktionen auf Unix- und Linux-Systemen installieren").

Standardwert
Unix- und Linux-Systeme: Kein Standardwert, UTMPATH muss immer gesetzt werden.
Windows-Systeme: UTMPATH wird bei der Installation gesetzt.

Prozess
Wird von jedem UTM-Prozess und beim Start eines UTM-Tools ausgewertet.

LANG

Bedeutung
Sprache, in der die UTM-Meldungen ausgegeben werden.

Wertebereich
Sprachkennzeichen z.B. De_DE.646. Für die Sprache muss ein NLS-Katalog vorhanden sein.

Standardwert
Wird LANG nicht oder falsch gesetzt (z.B. zum angegebenen Sprachkennzeichen wird kein NLS-Katalog gefunden), dann werden die Meldungen in englisch ausgegeben.

Prozess
Wird in jedem UTM-Prozess, der UTM-Meldungen ausgibt, beim Starten des Prozesses ausgewertet.

UTM_IPC_LETTER

Bedeutung
Legt die Größe des Datenbereichs im IPC-Shared Memory fest. Der Datenbereich dient zur Ablage der Nachrichten, die zwischen den Prozessen einer Anwendung ausgetauscht werden. In UTM_IPC_LETTER geben Sie die Anzahl der 4KB-Einheiten an, die der Datenbereich umfassen soll.

Wertebereich
Minimum: 5 (d.h. 20KB)

Standardwert
Abhängig von der Anzahl der generierten Semaphore.

Prozess

Wird im ersten Workprozess beim Starten einer UTM-Anwendung ausgewertet.

UTM_IPC_EXTP_LETTER

Bedeutung
Legt die Größe des Datenbereichs im IPC-Shared Memory fest, der pro Verbindung maximal zur Verfügung steht. Der in UTM_IPC_EXTP_LETTER angegebene Wert wird als Anzahl von 4KB-Bereichen interpretiert. Siehe auch "Performance-Verbesserung: Größe der Datenbereiche im IPC-Shared Memory anpassen".

Wertebereich
Minimum: 1 (d.h. 4KB)

Standardwert
16 (d.h. 64KB).

Prozess
Wird im ersten Workprozess beim Starten der Anwendung ausgewertet.

UTM_REDIRECT_FILES

Bedeutung
Legt fest, ob nach dem Start der UTM-Anwendung in die bestehenden Systemdateien *stderr* und *stdout* oder deren Ausgabeumlenkung geschrieben werden soll oder nicht. Ist UTM_REDIRECT_FILES auf "YES" gesetzt, wird nicht in *stdout* und *stderr* geschrieben. Die Dateien werden automatisch umgeschaltet und die Ausgaben in die Dateien *präfix*`.out.`*YY-MM-DD*`.`*HHMMSS* bzw. *präfix*`.err.`*YY-MM-DD*`.`*HHMMSS* geschrieben (siehe Abschnitt „Systemdateien stderr und stdout“).

Wertebereich
Folgende Werte sind möglich:

- Nicht gesetzt
- "YES" (Unix- und Linux-Systeme)

Standardwert
Nicht gesetzt. Das Verhalten ist kompatibel zu bisherigen openUTM-Versionen.

Prozess
Wird beim Starten einer UTM-Anwendung vom Prozess *utmmain* ausgewertet.

UTM_MAIN_KILL_TIME

Bedeutung
Enthält die Zeit in Sekunden, die der utmmain-Prozess durch den Aufruf von sleep() verzögert wird:

- vor dem Starten eines Workprozesses
- vor dem Löschen aller Betriebsmittel bei Anwendungsende
- vor dem Löschen der named pipe zum utmlog-Prozess
- nachdem der utmshut-Prozess gestartet wurde (Windows-Systeme)

Wertebereich
1 bis 99(sec)

Standardwert
1 (sec)
Sie können das Beenden der UTM-Anwendung bzw. das Nachstarten von Prozessen beschleunigen, wenn Sie UTM_MAIN_KILL_TIME auf einen kleineren Wert setzen.

Prozess
UTM_MAIN_KILL_TIME wird beim Starten der UTM-Anwendung im Prozess *utmmain* ausgewertet.

UTM_CORE_DUMP

Bedeutung
Verhindert, dass auf Unix- und Linux-Systemen Core-Dumps bzw. auf Windows-Systemen Mini-Dumps erzeugt werden,

- wenn es im Workprozess zu einem UTM-Dump kommt
- oder wenn externe UTM-Prozesse abnormal beendet werden.

Wertebereich
Falls die Umgebungsvariable gesetzt ist und den Wert "NO" enthält, wird kein core-Dump bzw. Mini-Dump erzeugt.

Standardwert
Keiner. Falls nicht gesetzt oder nicht mit "NO" belegt, wird in den oben beschriebenen Situationen ein core-Dump bzw. Mini-Dump erzeugt.

Prozess
Wird in jedem UTM-Prozess beim Anfordern eines core-Dumps bzw. Mini-Dumps ausgewertet.

UTM_MSG_DATE

Bedeutung
Verhindert, dass bei den Ausgaben auf STDOUT und STDERR das Datum und die Uhrzeit vorangestellt werden.

Wertebereich
Falls die Umgebungsvariable gesetzt ist und den Wert "NO" enthält, werden bei Meldungsausgabe nach STDOUT und STDERR weder Datum noch Uhrzeit vorangestellt.

Standardwert
Keiner. Falls die Umgebungsvariable nicht gesetzt ist oder nicht den Wert "NO" enthält, werden allen UTM-Meldungen Datum und Uhrzeit zur besseren Diagnose vorangestellt. Eine Ausnahme sind Meldungen von UTM-Tools. Bei diesen werden nie das Datum und Uhrzeit vorangestellt.

Prozess
UTM_MSG_DATE wird in jedem UTM-Prozess beim Starten des Prozesses ausgewertet.

UTM_MSG_PID

Bedeutung
Verhindert, dass bei den Ausgaben auf STDOUT und STDERR die PID des erzeugenden Prozesses vorangestellt wird.

Wertebereich
Falls die Umgebungsvariable gesetzt ist und den Wert "NO" enthält, wird bei Meldungsausgabe nach STDOUT und STDERR die PID des erzeugenden Prozesses nicht vorangestellt.

Standardwert
Keiner. Falls die Umgebungsvariable nicht gesetzt ist oder nicht den Wert "NO" enthält, wird allen UTM-Meldungen die PID des erzeugenden Prozesses zur besseren Diagnose vorangestellt.

Prozess
Wird beim Starten der UTM-Anwendung im Prozess *utmmain* einmalig ausgewertet.

UTMTRAC

Bedeutung
Schaltet optional den dynamischen Trace ein.

Wertebereich
Auswahl der zu protokollierenden UTM-Programme und der Trace-Units. Syntax siehe Handbuch „Meldungen, Test und Diagnose“.

Standardwert
Keiner. Falls nicht gesetzt, wird kein dynamischer Trace erzeugt.

Prozess
Wird in jedem UTM-Prozess beim Starten des Prozesses ausgewertet.

14.4.2 Umgebungsvariablen für Workprozesse

KDCS_C_DEBUG

Bedeutung
Ist KDCS_C_DEBUG gesetzt, dann wird jeder Aufruf eines C/C++- oder COBOL-Teilprogramms und jeder KDCS-Aufrufe in einem C/C++-Teilprogramm auf *stdout* protokolliert.

Wertebereich
Falls gesetzt, wird die Protokollierung aktiviert.

Standardwert
Keine Protokollierung.

Prozess
Wird in jedem Workprozess einer UTM-Anwendung beim ersten Aufruf eines C/C++- oder COBOL-Teilprogramms ausgewertet.

UTM_ABORT_WITH_EXCEPTION

Bedeutung
Legt fest, ob bei einem Anwendungsabbruch auf Unix- und Linux-Systemen ein core erzeugt bzw. auf Windows-Systemen der Debugger aktiviert wird.
UTM_ABORT_WITH_EXCEPTION sollte nur zusammen mit dem Startparameter STXIT=OFF verwendet werden.

Wertebereich
Falls gesetzt, wird bei Anwendungsabruch ein core erzeugt bzw. der Debugger aktiviert.

Standardwert
Kein Erzeugen eines cores bzw. kein Aktivieren des Debuggers.

Prozess
Wird in jedem Workprozess beim Anwendungsabbruch ausgewertet.

PATH

Bedeutung
Die Variable gilt nur für Unix- und Linux-Systeme.
Sie legt die Pfade fest, unter denen die Shellskripte *admlp* und *utmlp* gesucht werden. Das *admlp*-Skript wird beim Drucken von EAM-Dateien verwendet, das *utmlp*-Skript beim Drucken im Printerprozess *utmprint*.

Wertebereich
Anzahl von Dateiverzeichnissen (in der Form *datei-verzeichnis1*:*datei-verzeichnis2*:...)

Standardwert
Wird unter den in PATH angegebenen Dateiverzeichnissen das *admlp*- bzw. *utmlp* Skript nicht gefunden, dann wird das *admlp*- bzw. *utmlp*-Skript unter $UTMPATH/`shsc` verwendet.

Prozess
Wird in jedem Workprozess beim Drucken von EAM-Dateien und in jedem Druckerprozess beim Starten des Prozesses ausgewertet.

14.4.3 Umgebungsvariable für die Nutzung der openSSL Bibliothek

UTM_SSL_LIBRARY

Bedeutung

Gibt den vollqualifizierten Namen der openSSL Bibliothek an, die für die Verschlüsselungsfunktionalität und die TLS Verbindungen benötigt wird.

Wertebereich

Folgende Werte sind möglich:

- Nicht gesetzt
- Angabe eines Dateinamens

Der Dateiname enthält die komplette Pfadangabe für die openSSL Bibliothek.

Maximale Länge des Dateinamens: 300

Standardwert

- Variable nicht gesetzt
  - Unix- und Linux-Systeme: libssl.so
  - Windows-Systeme: libeay32.dll

  Es wird versucht die openSSL Bibliothek unter diesem Standard Namen zu laden.
- Variable mit Angabe eines Dateinamens:
  Es wird versucht die openSSL Bibliothek unter dem angegebenen Namen zu laden.

Prozess

Die Umgebungsvariable wird beim Starten des Dienstprogramms KDCDEF, eines *utmwork* und eines *utmnetss/* Prozesses ausgewertet.

Für den Einsatz der TLS Kommunikation auf Windows-Systemen muss eine zusätzliche Umgebungsvariable gesetzt werden:

UTM_SSL_LIBRARY2

Bedeutung

Gibt den vollqualifizierten Namen einer weiteren openSSL Bibliothek an, die für TLS Verbindungen benötigt wird.

Wertebereich

Folgende Werte sind möglich:

- Nicht gesetzt

- Angabe eines Dateinamens
  Der Dateiname enthält die komplette Pfadangabe für die openSSL Bibliothek
  Maximale Länge des Dateinamens: 300

Standardwert

- Variable nicht gesetzt
  ssleay32.dll

  Es wird versucht die openSSL Bibliothek unter dem Standard Namen zu laden.
- Variable mit Angabe eines Dateinamens:
  Es wird versucht die openSSL Bibliothek unter dem angegebenen Namen zu laden.

Prozess

Die Umgebungsvariable wird beim Starten eines *utmnetss/* Prozesses ausgewertet.

14.4.4 Umgebungsvariable für das Tool KDCDUMP

EDITOR

Bedeutung
Enthält den Editor, der vom Tool KDCDUMP beim Kommando `EDT` aufgerufen wird. Siehe openUTM-Handbuch „Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen“.

Wertebereich
Editorprogramm; z.B. vi, PFE32, notepad

Standardwert
Unix- und Linux-Systeme: vi
Windows-Systeme: wordpad

Prozess
EDITOR wird von KDCDUMP beim Kommando `EDT` ausgewertet.

14.4.5 Umgebungsvariablen für die X/Open Schnittstelle XATMI

Im Folgenden sind die Umgebungsvariablen aufgelistet, mit denen Sie XATMI-Anwendungen steuern können. Eine detaillierte Beschreibung zu diesen Umgebungsvariablen finden Sie im openUTM-Handbuch „Anwendungen erstellen mit X/Open-Schnittstellen“.

> **i** Die Umgebungsvariablen für das Steuern von Traces für die X/Open-Schnittstellen (ein-/ausschalten, Trace-Pfadnamen setzen) werden nicht mehr benötigt, da die Traces per Startparameter eingeschaltet werden können, siehe Abschnitt „Startparameterdatei der Anwendung“.

XTLCF

Bedeutung
Enthält den Namen der verwendeten Local Configuration File (LCF)

Wertebereich
Dateiname bzw. Pfadname, der den Konventionen des Betriebssystems entspricht. Wenn XTPALCF verwendet wird, darf XTLCF nur einen Dateinamen enthalten.

Standardwert
`xatmilcf` im Dateiverzeichnis, in dem die Anwendung gestartet wurde.

Prozess
XTLCF wird in jedem Workprozess beim Starten des Prozesses ausgewertet.

XTPALCF

Bedeutung
Legt die Dateiverzeichnisse fest, unter denen openUTM zusätzliche Beschreibungen von typisierten Puffern sucht. Siehe auch openUTM-Handbuch „Anwendungen erstellen mit X/Open-Schnittstellen“.

Wertebereich
Dateiverzeichnisse, angegeben in folgender Form:
Unix- und Linux-Systeme: *verzeichnis1*:*verzeichnis2*:...(getrennt durch Doppelpunkt)

Windows-Systeme: *verzeichnis1*;*verzeichnis2*;...(getrennt durch Semikolon)

Standardwert
Es wird nur die LCF in XTLCF durchsucht (bzw. die Datei `xatmilcf`, wenn XTLCF nicht gesetzt ist).

Prozess
XTPALCF wird in jedem Workprozess beim Starten des Prozesses ausgewertet.

14.4.6 Zusätzliche Umgebungsvariablen für openUTM auf Unix- und Linux- Systemen

UTM_NO_GCORE_DUMP

Bedeutung
Die Umgebungsvariablen UTM_NO_GCORE_DUMP gilt nur für Unix- und Linux-Systeme.
Mit ihr wird das Erzeugen eines gcore im Skript *utmgcore* geregelt.

Wertebereich
Folgende Werte sind möglich:

- nicht gesetzt
- "YES"

Standardwert

Wenn die Umgebungsvariable nicht gesetzt ist oder nicht den Wert „YES" enthält, fordert das Skript *utmgcore* einen gcore an.

Prozess
Die Umgebungsvariable UTM_NO_GCORE_DUMP wird im Skript *utmgcore* vom *utmwork*-Prozess ausgewertet.

14.4.7 Zusätzliche Umgebungsvariablen für openUTM auf Windows- Systemen

USERNAME

Bedeutung
Die Variable gilt nur für Windows-Systeme.
Sie enthält die UTM-Benutzerkennung für den Dialog-Terminalprozess *utmdtp*.

Wertebereich
UTM-Benutzerkennung

Standardwert
Windows-Benutzerkennung, unter der der *utmdtp* gestartet wird.

Prozess
USERNAME wird in jedem *utmdtp*-Prozess beim Starten ausgewertet.

UTM_PIPE_TIME

Bedeutung
Die Variable gilt nur für Windows-Systeme.
Sie enthält die Wartezeit in Sekunden bzgl. des named pipe handlings auf Windows-Systemen.

Wertebereich
1 - 99 (sec).

Standardwert
3 (sec).

Prozess
UTM_PIPE_TIME wird in jedem UTM-Prozess, der named pipes verwendet, beim Starten des Prozesses ausgewertet.

UTM_BREAK_BEFORE_DUMP

Bedeutung
Die Variable gilt nur für Windows-Systeme.
Sie legt fest, ob vor jedem UTM-Dump ein Breakpoint angefordert wird.

Wertebereich
Umgebungsvariable definiert oder nicht definiert. Es ist kein Wert notwendig.
Ist die Umgebungsvariable definiert, dann wird vor jedem UTM-Dump ein Breakpoint angefordert. In diesem Fall kann der Anwendungslauf unter der Kontrolle des Microsoft Visual Studio Debuggers fortgesetzt werden, um den Fehler zu analysieren. Dabei wird der UTM-Dump erzeugt (wie ohne Breakpoint).

Standardwert
Umgebungsvariable nicht definiert, d.h. kein Breakpoint.

Prozess

Die Umgebungsvariable wird im Workprozess vor jedem UTM-Dump ausgewertet.

UTM_BREAK_BEFORE_KCSTRMA

Bedeutung
Die Variable gilt nur für Windows-Systeme.
Sie legt fest, ob vor jedem Anwendungsabbruch ein Breakpoint angefordert wird.

Wertebereich
Umgebungsvariable definiert oder nicht definiert. Es ist kein Wert notwendig.
Ist die Umgebungsvariable definiert, dann wird vor jedem Anwendungsabbruch ein Breakpoint angefordert. In diesem Fall kann der Anwendungslauf unter der Kontrolle des Microsoft Visual Studio Debuggers fortgesetzt werden, um den Fehler zu analysieren. Dabei wird der UTM-Dump erzeugt (wie ohne Breakpoint).

Standardwert
Umgebungsvariable nicht definiert, d.h. kein Breakpoint.

Prozess
Die Umgebungsvariable wird im Workprozess vor jedem Anwendungsabbruch ausgewertet.

14.5 Aufbau der Accounting-Sätze von openUTM

Die Accounting-Sätze von openUTM werden in die Abrechnungsdateien im Datenverzeichnis ACCNT geschrieben.

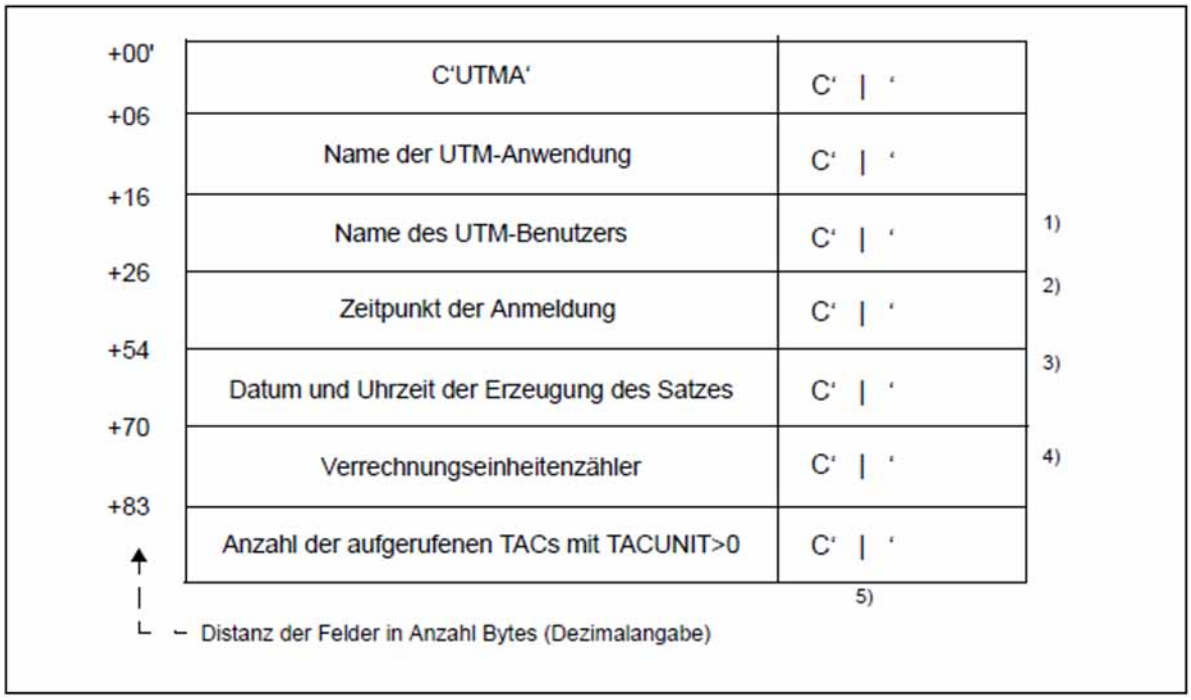
Es gibt folgende zwei Satztypen:

- Abrechnungssätze (Satztyp UTMA)
- Kalkulationssätze (Satztyp UTMK)

In diesem Abschnitt sind die Datenfelder der Sätze beschrieben, die UTM-spezifische Informationen enthalten.

Die Bedeutung dieser Sätze ist im Kapitel „Accounting“ beschrieben.

14.5.1 Aufbau des Abrechnungssatzes



Anmerkungen

1) Name des Benutzers. In einer UTM-Anwendung ohne generierte Benutzer trägt openUTM den Namen des LTERM-Partners ein.

2) Zeitpunkt der Anmeldung dieses Benutzers (USER) an diesem LTERM in der Form:

```
FRI  SEPT  15   00:00:00   2000
```

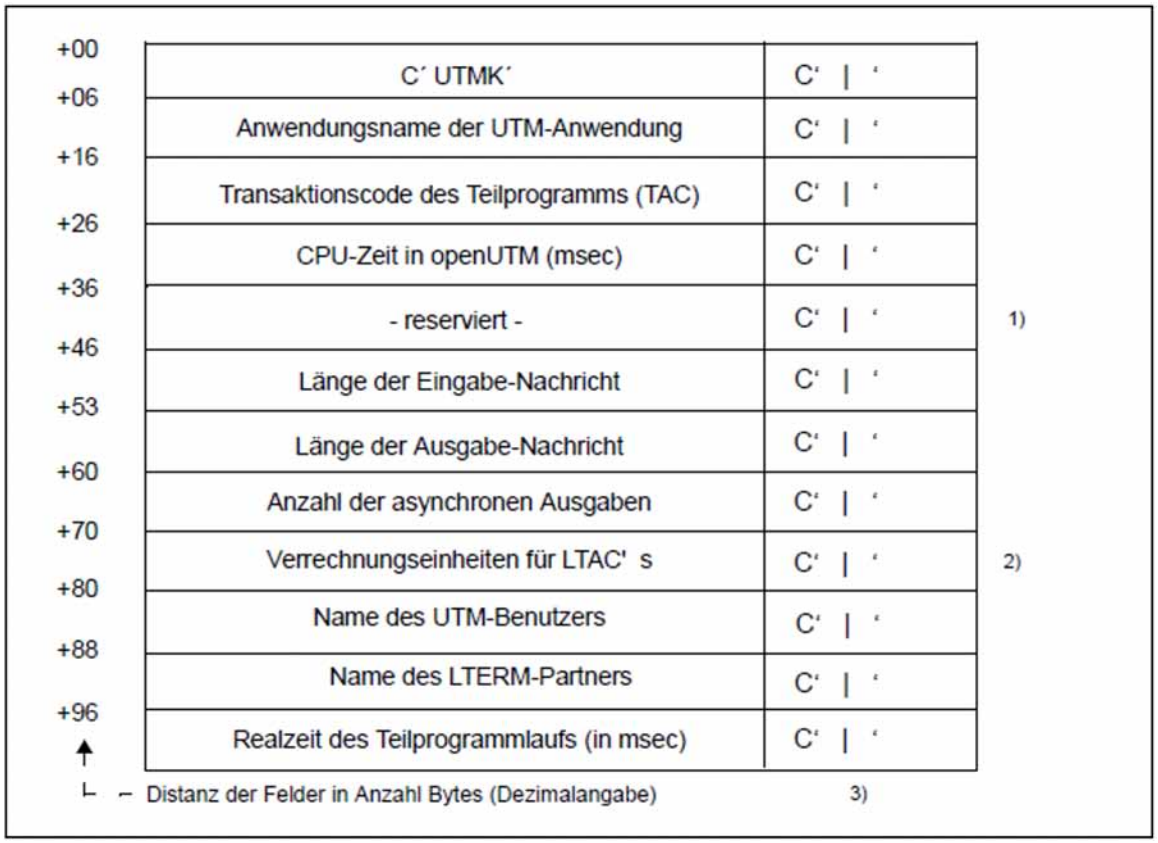
Wenn im aktuellen Lauf der UTM-Anwendung für diesen USER nur Asynchron-TACs aufgerufen wurden, ist der Inhalt dieses Feldes ------ .

3) Format: *yyyymmddhhmmss* (Jahr/Monat/Tag/Stunde/Minute/Sekunde)

4) Summe der Verrechnungseinheiten für diesen Benutzer, seit der letzte Verrechnungssatz geschrieben wurde bzw. seit dem Zeitpunkt der Anmeldung.

5) Die einzelnen Felder sind durch „ | “ voneinander getrennt.

14.5.2 Aufbau des Kalkulationssatzes



+00	C´ UTMK´	C' | '	
+06	Anwendungsname der UTM-Anwendung	C' | '	
+16	Transaktionscode des Teilprogramms (TAC)	C' | '	
+26	CPU-Zeit in openUTM (msec)	C' | '	
+36	- reserviert -	C' | '	1)
+46	Länge der Eingabe-Nachricht	C' | '	
+53	Länge der Ausgabe-Nachricht	C' | '	
+60	Anzahl der asynchronen Ausgaben	C' | '	
+70	Verrechnungseinheiten für LTAC' s	C' | '	2)
+80	Name des UTM-Benutzers	C' | '	
+88	Name des LTERM-Partners	C' | '	
+96	Realzeit des Teilprogrammlaufs (in msec)	C' | '	
↑			
L –	Distanz der Felder in Anzahl Bytes (Dezimalangabe)	3)	

Anmerkungen

1) Es wird immer 0 eingetragen.

2) siehe KDCDEF-Anweisung LTAC...,LTACUNIT=

3) Die einzelnen Felder sind durch „ | “ voneinander getrennt.

Beispiel: Abrechnungsphase mit Verteilter Verarbeitung über LU6.1 mit LTACs

Die Anwendung VTV10S im Rechner FILIALE kommuniziert mit Anwendung VTV10S im Rechner ZENTRALE, indem sie LTAC-Aufrufe an die Anwendung VTV10S in ZENTRALE sendet.

Rechner FILIALE

```
UTMK |VTV10S   |CVARL1   |       0 |         |   24 |   73 |       0 |       0 |apu
|BRCV0004 |      50 |
UTMK |VTV10S   |CVARL1   |       0 |         |   51 |   73 |       0 |       0 |apu
|BRCV0004 |      50 |
UTMK |VTV10S   |CVARL1   |       0 |         |    6 |  722 |       0 |       0 |apu
|BRCV0004 |      10 |
UTMK |VTV10S   |CVAR1    |      10 |         |  722 | 2205 |       0 |      50 |apu
|BRCV0004 |     100 |
UTMK |VTV10S   |CVAR2    |      20 |         | 2209 |   89 |       0 |       0 |apu
|BRCV0004 |      40 |
UTMA |VTV10S   |apu      |Fri Sep 15 11:34:50 2000  | 20000915113453 |    1133 |       5 |
UTMK |VTV10S   |KDCBADTC |      20 |         |    0 |  240 |       0 |       0 |apu
|BRCV0004 |      80 |
UTMA |VTV10S   |apu      |Fri Sep 15 11:34:50 2000  | 20000915113453 |     655 |       0 |
```

Rechner ZENTRALE

Diese Anwendung empfängt die von der Anwendung VTV10S an der FILIALE gesendeten LTACs. Die hier eingetragenen USER- und LTERM-Namen entsprechen dem Session- bzw. dem LPAP-Namen.

```
UTMK |VTV10S   |PERE     |      40 |         |   10 |   80 |       0 |       0 |VTV11S
|LPAP1    |     490 |
UTMK |VTV10S   |RT1      |      30 |         |   10 |   12 |       0 |       0 |VTV12S
|LPAP1    |     200 |
UTMK |VTV10S   |MPF      |      20 |         |   10 |   80 |       0 |       0 |VTV12S
|LPAP1    |     240 |
UTMK |VTV10S   |MPF      |      40 |         |   10 |   80 |       0 |       0 |VTV12S
|LPAP1    |     250 |
UTMK |VTV10S   |RT1      |      30 |         | 2205 | 2209 |       0 |       0 |VTV12S
|LPAP1    |     300 |
UTMK |VTV10S   |RT1      |      50 |         | 4400 | 4404 |       0 |       0 |VTV12S
|LPAP1    |     320 |
UTMK |VTV10S   |RT1      |      30 |         |   10 |   14 |       0 |       0 |VTV12S
|LPAP1    |     220 |
UTMK |VTV10S   |ATAC     |       0 |         |    5 |    0 |       0 |       0 |VTV12S
|LPAP1    |     170 |
...
...
UTMK |VTV10S   |VPASS    |    7940 |         |    8 |    4 |       0 |       0 |ap
|LTP00001 |   84460 |
```

14.6 Druckausgaben ohne Druckersteuerung abwickeln (Unix- und Linux-Systeme)

Die Druckausgabe bei einer UTM-Anwendung auf Unix- und Linux-Systemen wird im Automatikmodus über das Drucker-Shellskript *utmlp* gesteuert. Ist ein Drucker mit einer Anwendung verbunden, so existiert für ihn ein eigener Printerprozess. Erhält ein Drucker und damit ein Printerprozess einen Druckauftrag, dann startet der Printerprozess das Drucker-Shellskript *utmlp*. Die Daten werden in einer Pipe übergeben.

utmlp können Sie bei Bedarf selbst erstellen. Der Printerprozess wertet die Shell-Variable PATH aus und sucht darüber *utmlp*. Findet der Printerprozess darüber kein *utmlp*, dann startet er das Druckerskript *utmpfad*`/shsc /utmlp`, das mit openUTM ausgeliefert wird. Sie können dieses Shellskript Anwendungs-spezifisch abändern.

Nähere Informationen zum Skript *utmlp* finden Sie im openUTM-Handbuch „Anwendungen generieren“.

Tritt bei der Verarbeitung der Daten in *utmlp* ein Fehler auf, dann beendet sich das Druckerskript mit einem Exit-Code ungleich Null. Der Printerprozess erzeugt eine negative Abdruckquittung. Daraufhin baut openUTM die Verbindung zu diesem Printerprozess ab und erzeugt die Meldung K046. Der Printerprozess beendet sich.

Alle Ausgabe-Aufträge für einen Drucker werden von openUTM in der Message Queue des zugehörigen LTERM-Partners zwischengespeichert. Die zu druckenden Nachrichten gehen also im Fall einer negativen Abdruckquittung nicht verloren. Sie werden beim nächsten Verbindungsaufbau an diesen Drucker gesendet.

14.7 Beispielprogramme und Beispielanwendungen

Für openUTM werden standardmäßig Beispielprogramme, Beispielprozeduren sowie ablauffähige Beispielanwendungen ausgeliefert. Diese Beispiele können Sie als Vorlage für die eigene Anwendungsentwicklung nehmen und entsprechend anpassen. Die Beschreibung der Beispielprogramme für die Administration finden Sie im openUTM-Handbuch „Anwendungen administrieren“.

14.7.1 Beispielprogramme für Publish / Subscribe Server

Mit diesen Beispielprogrammen soll gezeigt werden, wie ein einfacher Publish- und Subsribe-Dienst in einer UTM-Anwendung realisiert werden kann.

Funktion

Ein Benutzer kann sich beim Dienst anmelden (subsrcibe). Er bekommt dann alle ab diesem Zeitpunkt veröffentlichten Nachrichten (publish) in seiner USER-Queue zugestellt.Die möglichen Kommandos an den Dienst sind:

- help: Hilfetext holen
- subscribe: Nachrichten abonnieren
- unsubscribe: Nachrichten abbestellen
- who: Namen der Abonnenten ausgeben
- publish <*message*>: Nachricht veröffentlichen

Der Dienst wird von einem Asynchron-Vorgang mit dem TAC PUBSUBA erbracht, der ständig an der TAC-Queue PUBSUBMQ auf Aufträge wartet. Die Benutzer kommunizieren mit dem Dienst über den Dialog-Vorgang PUPSUBD. Auftragsbestätigungen werden an die USER-Queue des Benutzers gesendet und können z.B. mit dem Dialog-Programm UPDGET gelesen werden (siehe Beispielprogramme zur Asynchron-Verarbeitung für UPIC-Client). Außerdem kann in jedem Teilprogramm beim INIT PU abgefragt werden, ob Nachrichten in der Queue des Benutzers vorliegen.

Der Dienst muss nur einmal durch Aufruf des TAC PUBSUBA gestartet werden. Der offene Asynchron-Vorgang bleibt dann auch über den Anwendungslauf hinweg erhalten. Nach Neugenerierung wird er durch KDCUPD in die neue Anwendung übertragen.

Sollte sich durch einen Fehler der Asynchron-Vorgang abnormal beenden, so wird der zuletzt bearbeitete Auftrag in die Dead Letter Queue gestellt.

Auslieferung

Die Programme sind Bestandteil der Beispielanwendung und werden als *pubsubd.c* und *pubsuba.c* ausgelietert.

UTM-Generierung

Die Anweisungen für die Teilprogramme im KDCDEF-Lauf sind in den einzelnen Sourcen als Kommentar angegeben. Ebenso die Anweisung für die TAC-Queue „PUBSUBMQ“.

Es muss mindestens ein GSSB generiert werden (MAX GSSBS), da der Service zur Verwaltung der Abonnenten den GSSB „PUBSUBGB“ verwendet.

Soll nach Abbruch des Service der zuletzt bearbeitete Auftrag in die Dead Letter Queue gestellt werden, so muss MAX REDELIVERY = (...,0) generiert werden. Ansonsten bleibt er in der Auftragsqueue PUBSUBMQ.

14.7.2 Beispielprogramm für selektives Verschieben aus der Dead Letter Queue

Funktion

Das Dialog-Programm verschiebt alle Nachrichten der Dead Letter Queue mit vorgegebenem ursprünglichen Ziel an ein vorgegebenes neues Ziel. Als Eingabe werden daher insgesamt 16 Zeichen erwartet, d.h. je nach Typ des ursprünglichen Ziels entweder zwei TACs oder zwei LPAP-Partner oder zwei OSI-LPAP-Partner. Das Programm gibt zur Bestätigung die Anzahl der verschobenen Nachrichten aus.

Auslieferung

Das C-Programm ist Bestandteil der Beispielanwendung und wird als *dadmmvsc.c* ausgeliefert.

UTM-Generierung

Die Anweisungen für die Teilprogramme im KDCDEF-Lauf sind in den einzelnen Sourcen als Kommentar angegeben.

14.7.3 Beispielprogramme für HTTP-Clients

Es stehen vier Beispielprogramme für HTTP-Clients zur Verfügung.

- HTTPECHO
- HTTPEXH
- HTTPEXT
- HTTPINF

Funktion

Details zur Funktionalität sind in den einzelnen Sourcen als Kommentar angegeben.

Auslieferung

Die C-Programme sind Bestandteil der Beispielanwendung und werden als *httpecho.c*, *httpexh.c*, *httpext.c* sowie *httpinf.c* ausgeliefert.

UTM-Generierung

Die Anweisungen für die Teilprogramme im KDCDEF-Lauf sind in den einzelnen Sourcen als Kommentar angegeben.

14.7.4 CPI-C-Beispielprogramme

Unix- und Linux-Systeme

CPI-C-Beispielprogramme finden Sie auf Unix- und Linux-Systemen in *utmpfad*`/cpic/sample`.

Name	Bedeutung
../src/kcpsam1.c	C-Source (asynchroner Teil)
../src/kcpsam2.c	C-Source (synchroner Teil)
../sys/libcpsam.a	Bibliothek mit Objekten von asynchronem Teil und synchronem Teil

Windows-Systeme

Auf Windows-Systemen sind diese Programme im Quick Start Kit enthalten, siehe "openUTM Quick Start Kit für Windows-Systeme".

14.7.5 Beispielprozeduren für Unix- und Linux-Systeme

Auf Unix- und Linux-Systemen sind die unten angeführten Prozeduren Produktbestandteile von openUTM und werden beispielsweise beim Installieren oder Erstellen der Anwendung benötigt. Sie können die Prozeduren nach Ihren Wünschen modifizieren und erweitern. Die Prozeduren enthalten englische Kommentare.

Im Dateiverzeichnis *utmpfad*`/shsc` werden folgende Beispielprozeduren ausgeliefert:

Prozedur	**Funktion**
admlp	Administrationskommandoausgaben mit dem `lp`-Kommando ausdrucken
CCmainutm	erzeugt ein Objekt für den C++-Anschluss
dumpstart	dient zur symbolischen Auswertung eines UTM-Dumps
stat2dyn	erzeugt aus einer statischen Bibliothek eine Shared Objects-Bibliothek
utmlp	zur Verarbeitung von Druckausgaben
utm-c.emergency	Emergency-Skript für UTM-Cluster-Anwendungen
utm-c.failure	Failure-Skript für UTM-Cluster-Anwendungen

14.7.6 Beispielprozeduren für Windows-Systeme

Auf Windows-Systemen sind die unten angeführten Prozeduren Produktbestandteile von openUTM und werden für den Betrieb von UTM-Cluster-Anwendungen benötigt. Sie können die Prozeduren nach Ihren Wünschen modifizieren und erweitern. Die Prozeduren enthalten englische Kommentare.

Im Dateiverzeichnis *utmpfad*\shsc werden folgende Beispielprozeduren ausgeliefert:

Prozedur	**Funktion**
utm-c.emergency	Emergency-Skript für UTM-Cluster-Anwendungen
utm-c.failure	Failure-Skript für UTM-Cluster-Anwendungen

14.7.7 Beispielanwendung für Unix- und Linux-Systeme

Zum Lieferumfang von openUTM gehört eine Beispielanwendung *utmsample*. Die Datei `CPIO.utmsample` der Beispielanwendung wird zusammen mit openUTM installiert und befinden sich in *utmpfad*.

Sie installieren die Beispielanwendung inklusive Beschreibung unter Ihrer Benutzerkennung, in dem Sie die Prozedur *utmpfad*`/shsc/install.sample` aufrufen. Aus den Dateien der Beispielanwendung können Sie durch Aufruf einer Shell-Prozedur sofort eine einfache UTM-Anwendung erzeugen.

Mit der Beispielanwendung können Sie sich die UTM-Generierung und die Inbetriebnahme Ihrer UTM-Anwendung vereinfachen. Erzeugen Sie dazu aus der Beispielanwendung eine Anwendung, die die Komponenten und Schnittstellen verwendet, die Sie für Ihre Anwendung benötigen (Datenbankanschluss z.B. mit Oracle, verteilte Verarbeitung über OSI TP bzw. LU6.1 etc.). Die Input-Datei für das Generierungstool KDCDEF und die Makefile dieser Anwendung können Sie dann als Vorlage für Ihre Anwendung verwenden.

Zu allen Programmen der Beispielanwendung werden die Quellcodes ausgeliefert, um Ihnen die Programmierung eigener UTM-Programme zu erleichtern.

14.7.8 openUTM Quick Start Kit für Windows-Systeme

Mit openUTM auf Windows wird die UTM-Beispielanwendung inklusive Prozeduren in Form eines Quick Start Kit ausgeliefert. Das Quick Start Kit setzt ein bereits installiertes openUTM voraus.

Das Quick Start Kit enthält ablauffähige Beispielprogramme, angefangen von einem CPI-C-Client bis zur UTM-Datenbank-Anwendung. Für diese Programme werden bei der Installation Icons eingerichtet, so dass Sie die Programme einfach per Mausklick starten können. Damit lernen Sie die Funktionalität von openUTM auf einfache Art und Weise kennen.

Im Quick Start Kit sind auch die zugehörigen Programm-Sourcen, Kommandodateien und Konfigurationsdateien enthalten. Diese Sourcen sollen Ihnen die Programmierung eigener Anwendungen erleichtern und können z.B. als Gerüst für eigene Anwendungen hergenommen werden.

Das Quick Start Kit enthält auch eine Dokumentation. Darin sind die ausgelieferten Programme und Source-Dateien mit ihrer Funktion kurz beschrieben.

15 Fachwörter

Fachwörter, die an anderer Stelle erklärt werden, sind mit *kursiver* Schrift ausgezeichnet.

Ablaufinvariantes Programm
reentrant program

siehe *reentrant-fähiges Programm*.

Abnormale Beendigung einer UTM-Anwendung
abnormal termination of a UTM application

Beendigung einer *UTM-Anwendung*, bei der die *KDCFILE* nicht mehr aktualisiert wird. Eine abnormale Beendigung wird ausgelöst durch einen schwerwiegenden Fehler, z.B. Rechnerausfall, Fehler in der Systemsoftware. Wird die Anwendung erneut gestartet, führt openUTM einen *Warmstart* durch.

Abstrakte Syntax (OSI)
abstract syntax

Eine abstrakte Syntax ist die Menge der formal beschriebenen Datentypen, die zwischen Anwendungen über *OSI TP* ausgetauscht werden sollen. Eine abstrakte Syntax ist unabhängig von der eingesetzten Hardware und der jeweiligen Programmiersprache.

Access-List
access list

Eine Access-List definiert die Berechtigung für den Zugriff auf einen bestimmten *Service*, auf eine bestimmte *TAC-Queue* oder auf eine bestimmte *USER-Queue*. Eine Access-List ist als *Keyset* definiert und enthält einen oder mehrere *Keycodes*, die jeweils eine Rolle in der Anwendung repräsentieren. Benutzer, LTERMs oder (OSI-)LPAPs dürfen nur dann auf den Service oder die *TAC-Queue*/*USER-Queue* zugreifen, wenn ihnen die entsprechenden Rollen zugeteilt wurden, d.h. wenn ihr *Keyset* und die Access-List mindestens einen gemeinsamen *Keycode* enthalten.

Access Point (OSI)

siehe *Dienstzugriffspunkt.*

ACID-Eigenschaften
ACID properties

Abkürzende Bezeichnung für die grundlegenden Eigenschaften von *Transaktionen*: Atomicity, Consistency, Isolation und Durability.

Administration
administration

Verwaltung und Steuerung einer *UTM-Anwendung* durch einen *Administrator* oder ein *Administrationsprogramm*.

Administrations-Journal
administration journal

siehe *Cluster-Administrations-Journal*.

Administrationskommando
administration command

Kommandos, mit denen der *Administrator* einer *UTM-Anwendung* Administrationsfunktionen für diese Anwendung durchführt. Die Administrationskommandos sind als *Transaktionscodes* realisiert.

Administrationsprogramm
administration program

Teilprogramm, das Aufrufe der *Programmschnittstelle für die Administration* enthält. Dies kann das Standard-Administrationsprogramm *KDCADM* sein, das mit openUTM ausgeliefert wird, oder ein vom Anwender selbst erstelltes Programm.

Administrator
administrator

Benutzer mit Administrationsberechtigung.

AES

AES (Advanced Encryption Standard) ist der aktuelle symmetrische Verschlüsselungsstandard, festgelegt vom NIST (National Institute of Standards and Technology), basierend auf dem an der Universität Leuven (B) entwickelten Rijndael-Algorithmus. Wird das AES-Verfahren verwendet, dann erzeugt der UPIC-Client für jede Sitzung einen AES-Schlüssel.

Akzeptor (CPI-C)
acceptor

Die Kommunikationspartner einer *Conversation* werden *Initiator* und Akzeptor genannt. Der Akzeptor nimmt die vom Initiator eingeleitete Conversation mit Accept_Conversation entgegen.

Anmelde-Vorgang (KDCS)
sign-on service

Spezieller *Dialog-Vorgang*, bei dem die Anmeldung eines Benutzers an eine UTM-Anwendung durch *Teilprogramme* gesteuert wird.

Anschlussprogramm
linkage program

siehe *KDCROOT*.

Anwendungsinformation
application information

Sie stellt die Gesamtmenge der von der *UTM-Anwendung* benutzten Daten dar. Dabei handelt es sich um Speicherbereiche und Nachrichten der UTM-Anwendung, einschließlich der aktuell auf dem Bildschirm angezeigten Daten. Arbeitet die UTM-Anwendung koordiniert mit einem Datenbanksystem, so gehören die in der Datenbank gespeicherten Daten ebenfalls zur Anwendungsinformation.

Anwendungs-Kaltstart
application cold start

siehe *Kaltstart*.

Anwendungsprogramm
application program

Ein Anwendungsprogramm bildet den Hauptbestandteil einer *UTM-Anwendung*. Es besteht aus der Main Routine *KDCROOT* und den *Teilprogrammen*. Es bearbeitet alle Aufträge, die an eine *UTM-Anwendung* gerichtet werden.

Anwendungs-Warmstart
application warm start

siehe *Warmstart*.

Apache Axis

Apache Axis (Apache eXtensible Interaction System) ist eine SOAP-Engine zur Konstruktion von darauf basierenden Web Services und Client-Anwendungen. Es existiert eine Implementierung in C++ und Java.

Apache Tomcat

Apache Tomcat stellt eine Umgebung zur Ausführung von Java-Code auf Web-Servern bereit, die im Rahmen des Jakarta-Projekts der Apache Software Foundation entwickelt wird. Es handelt sich um einen in Java geschriebenen Servlet-Container, der mithilfe des JSP-Compilers Jasper auch JavaServer Pages in Servlets übersetzen und ausführen kann. Dazu kommt ein kompletter HTTP-Server.

Application Context (OSI)
application context

Der Application Context ist die Menge der Regeln, die für die Kommunikation zwischen zwei Anwendungen gelten sollen. Dazu gehören z.B. die *abstrakten Syntaxen* und die zugeordneten *Transfer-Syntaxen*.

Application Entity (OSI)
application entity

Eine Application Entity (AE) repräsentiert alle für die Kommunikation relevanten Aspekte einer realen Anwendung. Eine Application Entity wird durch einen global (d.h. weltweit) eindeutigen Namen identifiziert, den *Application Entity Title* (AET). Jede Application Entity repräsentiert genau einen *Application Process*. Ein Application Process kann mehrere Application Entities umfassen.

Application Entity Qualifier (OSI)
application entity qualifier

Bestandteil des *Application Entity Titles*. Der Application Entity Qualifier identifiziert einen *Dienstzugriffspunkt* innerhalb der Anwendung. Ein Application Entity Qualifier kann unterschiedlich aufgebaut sein. openUTM unterstützt den Typ "Zahl".

Application Entity Title (OSI)
application entity title

Ein Application Entity Title ist ein global (d.h. weltweit) eindeutiger Name für eine *Application Entity*. Er setzt sich zusammen aus dem *Application Process Title* des jeweiligen *Application Process* und dem *Application Entity Qualifier.*

Application Process (OSI)
application process

Der Application Process repräsentiert im *OSI-Referenzmodell* eine Anwendung. Er wird durch den *Application Process Title* global (d.h. weltweit) eindeutig identifiziert.

Application Process Title (OSI)
application process title

Gemäß der OSI-Norm dient der Application Process Title (APT) zur global (d.h. weltweit) eindeutigen Identifizierung von Anwendungen. Er kann unterschiedlich aufgebaut sein. openUTM unterstützt den Typ *Object Identifier*.

Application Service Element (OSI)
application service element

Ein Application Service Element (ASE) repräsentiert eine Funktionsgruppe der Anwendungsschicht (Schicht 7) des *OSI-Referenzmodells*.

Association (OSI)
association

Eine Association ist eine Kommunikationsbeziehung zwischen zwei *Application Entities*. Dem Begriff Association entspricht der *LU6.1*-Begriff *Session*.

Asynchron-Auftrag
queued job

Auftrag, der vom Auftraggeber zeitlich entkoppelt durchgeführt wird. Zur Bearbeitung von Asynchron-Aufträgen sind in openUTM *Message Queuing* Funktionen integriert, vgl. *UTM-gesteuerte Queue* und *Service-gesteuerte Queue*. Ein Asynchron-Auftrag wird durch die *Asynchron-Nachricht*, den Empfänger und ggf. den gewünschten Ausführungszeitpunkt beschrieben.
Ist der Empfänger ein Terminal, ein Drucker oder eine Transportsystem-Anwendung, so ist der Asynchron-Auftrag ein *Ausgabe-Auftrag*; ist der Empfänger ein Asynchron-Vorgang derselben oder einer fernen Anwendung, so handelt es sich um einen *Hintergrund-Auftrag*.
Asynchron-Aufträge können *zeitgesteuerte Aufträge* sein oder auch in einen *Auftrags-Komplex* integriert sein.

Asynchron-Conversation
asynchronous conversation

CPI-C-Conversation, bei der nur der *Initiator* senden darf. Für den *Akzeptor* muss in der *UTM-Anwendung* ein asynchroner Transaktionscode generiert sein.

Asynchron-Nachricht
asynchronous message

Asynchron-Nachrichten sind Nachrichten, die an eine *Message Queue* gerichtet sind. Sie werden von der lokalen *UTM-Anwendung* zunächst zwischengespeichert und dann unabhängig vom Auftraggeber weiter verarbeitet. Je nach Empfänger unterscheidet man folgende Typen von Asynchron-Nachrichten:

- Bei Asynchron-Nachrichten an eine *UTM-gesteuerte Queue* wird die Weiterverarbeitung komplett durch openUTM gesteuert. Zu diesem Typ gehören Nachrichten, die einen lokalen oder fernen *Asynchron-Vorgang* starten (vgl. auch *Hintergrund-Auftrag*) und Nachrichten, die zur Ausgabe an ein Terminal, einen Drucker oder eine Transportsystem-Anwendung geschickt werden (vgl. auch *Ausgabe-Auftrag*).
- Bei Asynchron-Nachrichten an eine *Service-gesteuerte Queue* wird die Weiterverarbeitung durch einen *Service* der Anwendung gesteuert. Zu diesem Typ gehören Nachrichten an eine *TAC-Queue*, Nachrichten an eine *USER-Queue* und Nachrichten an eine *Temporäre Queue*. Die User-Queue und die Temporäre Queue müssen dabei zur lokalen Anwendung gehören, die TAC-Queue kann sowohl in der lokalen als auch in einer fernen Anwendung liegen.

Asynchron-Programm
asynchronous program

Teilprogramm, das von einem *Hintergrund-Auftrag* gestartet wird.

Asynchron-Vorgang (KDCS)
asynchronous service

Vorgang , der einen *Hintergrund-Auftrag* bearbeitet. Die Verarbeitung erfolgt entkoppelt vom Auftraggeber. Ein Asynchron-Vorgang kann aus einem oder mehreren Teilprogrammen /Transaktionen bestehen. Er wird über einen asynchronen *Transaktionscode* gestartet.

Auftrag
job

Anforderung eines *Services* , der von einer *UTM-Anwendung* zur Verfügung gestellt wird, durch Angabe eines *Transaktionscodes* . Siehe auch: *Ausgabe-Auftrag* , *Dialog-Auftrag* , *Hintergrund-Auftrag* , *Auftrags-Komplex*.

Auftraggeber-Vorgang
job-submitting service

Ein Auftraggeber-Vorgang ist ein *Vorgang*, der zur Bearbeitung eines Auftrags einen Service von einer anderen Server-Anwendung (*Auftragnehmer-Vorgang*) anfordert.

Auftragnehmer-Vorgang
job-receiving service

Ein Auftragnehmer-Vorgang ist ein *Vorgang*, der von einem *Auftraggeber-Vorgang* einer anderen Server-Anwendung gestartet wird.

Auftrags-Komplex
job complex

Auftrags-Komplexe dienen dazu, *Asynchron-Aufträgen Quittungsaufträge* zuzuordnen. Ein Asynchron-Auftrag innerhalb eines Auftrags-Komplexes wird *Basis-Auftrag* genannt.

Ausgabe-Auftrag
queued output job

Ausgabeaufträge sind *Asynchron-Aufträge*, die die Aufgabe haben, eine Nachricht, z.B. ein Dokument, an einen Drucker, ein Terminal oder eine Transportsystem-Anwendung auszugeben. Ausgabeaufträge werden ausschließlich von UTM-Systemfunktionen bearbeitet, d.h. für die Bearbeitung müssen keine Teilprogramme erstellt werden.

Authentisierung
authentication

siehe *Zugangskontrolle*.

Autorisierung
authorization

siehe *Zugriffskontrolle*.

Axis

siehe *Apache Axis*.

Basis-Auftrag
basic job

Asynchron-Auftrag in einem *Auftrags-Komplex*.

Basisformat
basic format

Format, in das der Terminal-Benutzer alle Angaben eintragen kann, die notwendig sind, um einen Vorgang zu starten.

Basisname
filebase

Basisname der UTM-Anwendung.
Auf BS2000-Systemen ist Basisname das Präfix für die *KDCFILE*, die *Benutzerprotokoll-Datei* USLOG und die *System-Protokolldatei* SYSLOG.
Auf Unix-, Linux- und Windows-Systemen ist Basisname der Name des Verzeichnisses, unter dem die KDCFILE, die Benutzerprotokoll-Datei USLOG, die System-Protokolldatei SYSLOG und weitere Dateien der UTM-Anwendung abgelegt sind.

Basisname der Knoten-Anwendung
node filebase

Dateinamens-Präfix bzw. Verzeichnisname für die *KDCFILE*, *Benutzerprotokoll-Datei* und *Systemprotokoll-Datei* der *Knoten-Anwendung*.

Basisname der UTM-Cluster-Anwendung
cluster filebase

Dateinamens-Präfix bzw. Verzeichnisname für die *UTM-Cluster-Dateien*.

Benutzerausgang
user exit

Begriff ersetzt durch *Event-Exit.*

Benutzerkennung
user ID

Bezeichner für einen Benutzer, der in der *Konfiguration* der *UTM-Anwendung* festgelegt ist (optional mit Passwort zur *Zugangskontrolle*) und dem spezielle Zugriffsrechte (*Zugriffskontrolle*) zugeordnet sind. Ein Terminal-Benutzer muss bei der Anmeldung an die UTM-Anwendung diesen Bezeichner (und ggf. das zugeordnete Passwort) angeben. Auf BS2000-Systemen ist außerdem eine Zugangskontrolle über *Kerberos* möglich.
Für andere Clients ist die Angabe der Benutzerkennung optional, siehe auch *Verbindungs-Benutzerkennung*.
UTM-Anwendungen können auch ohne Benutzerkennungen generiert werden.

Benutzer-Protokolldatei
user log file

Datei oder Dateigeneration, in die der Benutzer mit dem KDCS-Aufruf LPUT Sätze variabler Länge schreibt. Jedem Satz werden die Daten aus dem KB-Kopf des *KDCS-Kommunikationsbereichs* vorangestellt. Die Benutzerprotokolldatei unterliegt der Transaktionssicherung von openUTM.

Berechtigungsprüfung
sign-on check

siehe *Zugangskontrolle*.

Beweissicherung (BS2000-Systeme)
audit

Im Betrieb einer *UTM-Anwendung* können zur Beweissicherung sicherheitsrelevante UTM-Ereignisse von *SAT* protokolliert werden.

Bildschirm-Wiederanlauf
screen restart

Wird ein *Dialog-Vorgang* unterbrochen, gibt openUTM beim *Vorgangswiederanlauf* die *Dialog-Nachricht* der letzten abgeschlossenen *Transaktion* erneut auf dem Bildschirm aus, sofern die letzte Transaktion eine Nachricht auf den Bildschirm ausgegeben hat.

Browsen von Asynchron-Nachrichten
browsing asynchronous messages

Ein *Vorgang* liest nacheinander die *Asynchron-Nachrichten*, die sich in einer *Service-gesteuerten Queue* befinden. Die Nachrichten werden während des Lesens nicht gesperrt und verbleiben nach dem Lesen in der Queue. Dadurch ist gleichzeitiges Lesen durch unterschiedliche Vorgänge möglich.

Bypass-Betrieb (BS2000-Systeme)
bypass mode

Betriebsart eines Druckers, der lokal an ein Terminal angeschlossen ist. Im Bypass-Betrieb wird eine an den Drucker gerichtete *Asynchron-Nachricht* an das Terminal gesendet und von diesem auf den Drucker umgeleitet, ohne auf dem Bildschirm angezeigt zu werden.

Cache-Speicher
cache

Pufferbereich zur Zwischenspeicherung von Anwenderdaten für alle Prozesse einer *UTM-Anwendung*. Der Cache-Speicher dient zur Optimierung der Zugriffe auf den *Pagepool* und für UTM-Cluster-Anwendungen zusätzlich auf den *Cluster-Pagepool*.

CCR (Commitment, Concurrency and Recovery)

CCR ist ein von OSI definiertes Application Service Element (ASE) für die OSI-TP-Kommunkation, welches die Protokollelemente (Services) zum Beginn und Abschluss (Commit oder Rollback) einer *Transaktion* enthält. CCR unterstützt das Zwei-Phasen-Commitment.

CCS-Name (BS2000-Systeme)
CCS name

siehe *Coded-Character-Set-Name* .

Client
client

Clients einer *UTM-Anwendung* können sein:

- Terminals
- UPIC-Client-Programme
- Transportsystem-Anwendungen (z.B. DCAM-, PDN-, CMX-, Socket-Anwendungen oder UTM-Anwendungen, die als *Transportsystem-Anwendung* generiert sind)

Clients werden über LTERM-Partner an die UTM-Anwendung angeschlossen.
Hinweis: UTM-Clients mit Trägersystem OpenCPIC werden wie *OSI TP-Partner* behandelt.

Client-Seite einer Conversation
client side of a conversation

Begriff ersetzt durch *Initiator*.

Cluster

Eine Anzahl von Rechnern, die über ein schnelles Netzwerk verbunden sind und die von außen in vielen Fällen als ein Rechner gesehen werden können. Das Ziel des "Clustering" ist meist die Erhöhung der Rechenkapazität oder der Verfügbarkeit gegenüber einem einzelnen Rechner.

Cluster-Administrations-Journal
cluster administration journal

Das Cluster-Administrations-Journal besteht aus:

- zwei Protokolldateien mit Endungen JRN1 und JRN2 für globale Administrationsaktionen,
- der JKAA-Datei, die eine Kopie der KDCS Application Area (KAA) enthält. Aus dieser Kopie werden administrative Änderungen übernommen, die nicht mehr in den beiden Protokolldateien enthalten sind.

Die Administrations-Journal-Dateien dienen dazu, administrative Aktionen, die in einer UTM-Cluster-Anwendung Cluster-weit auf alle Knoten-Anwendungen wirken sollen, an die anderen Knoten-Anwendungen weiterzugeben.

Cluster-GSSB-Datei
cluster GSSB file

Datei zur Verwaltung von GSSBs in einer *UTM-Cluster-Anwendung*. Die Cluster-GSSB-Datei wird mit dem UTM-Generierungstool *KDCDEF* erstellt.

Cluster-Konfigurationsdatei
cluster configuration file

Datei, die die zentralen Konfigurationsdaten einer *UTM-Cluster-Anwendung* enthält. Die Cluster-Konfigurationsdatei wird mit dem UTM-Generierungstool *KDCDEF* erstellt.

Cluster-Lock-Datei
cluster lock file

Datei einer *UTM-Cluster-Anwendung*, die dazu dient, Knoten-übergreifende Sperren auf Anwenderdatenbereiche zu verwalten.

Cluster-Pagepool
cluster pagepool

Der Cluster-Pagepool besteht aus einer Verwaltungsdatei und bis zu 10 Dateien, in denen die Cluster-weit verfügbaren Anwenderdaten (Vorgangsdaten inklusive LSSB, GSSB und ULS) einer *UTM-Cluster-Anwendung* gespeichert werden. Der Cluster-Pagepool wird mit dem UTM-Generierungstool *KDCDEF* erstellt.

Cluster-Startserialisierungs-Datei
cluster start serialization file

Lock-Datei, mit der die Starts einzelner Knoten-Anwendungen serialisiert werden (nur auf Unix-, Linux- und Windows-Systemen).

Cluster-ULS-Datei
cluster ULS file

Datei zur Verwaltung von ULS-Bereichen einer *UTM-Cluster-Anwendung*. Die Cluster-ULS-Datei wird mit dem UTM-Generierungstool *KDCDEF* erstellt.

Cluster-User-Datei
cluster user file

Datei, die die Verwaltungsdaten der Benutzer einer *UTM-Cluster-Anwendung* enthält. Die Cluster-User-Datei wird mit dem UTM-Generierungstool *KDCDEF* erstellt.

Coded-Character-Set-Name (BS2000-Systeme)
coded character set name

Bei Verwendung des Produkts *XHCS* (e**X**tended **H**ost **C**ode **S**upport) wird jeder verwendete Zeichensatz durch einen Coded-Character-Set-Namen (abgekürzt: "CCS-Name" oder "CCSN") eindeutig identifiziert.

Communication Resource Manager
communication resource manager

Communication Resource Manager (CRMs) kontrollieren in verteilten Systemen die Kommunikation zwischen den Anwendungsprogrammen. openUTM stellt CRMs für den internationalen Standard OSI TP, für den Industrie-Standard *LU6.1* und für das openUTM-eigene Protokoll UPIC zur Verfügung.

Contention Loser
contention loser

Jede Verbindung zwischen zwei Partnern wird von einem der Partner verwaltet. Der Partner, der die Verbindung verwaltet, heißt *Contention Winner*. Der andere Partner ist der Contention Loser.

Contention Winner
contention winner

Der Contention Winner einer Verbindung übernimmt die Verwaltung der Verbindung. Aufträge können sowohl vom Contention Winner als auch vom *Contention Loser* gestartet werden. Im Konfliktfall, wenn beide Kommunikationspartner gleichzeitig einen Auftrag starten wollen, wird die Verbindung vom Auftrag des Contention Winner belegt.

Conversation
conversation

Bei CPI-C nennt man die Kommunikation zwischen zwei CPI-C-Anwendungsprogrammen Conversation. Die Kommunikationspartner einer Conversation werden *Initiator* und *Akzeptor* genannt.

Conversation-ID
conversation ID

Jeder *Conversation* wird von CPI-C lokal eine Conversation-ID zugeordnet, d.h. *Initiator* und *Akzeptor* haben jeweils eine eigene Conversation-ID. Mit der Conversation-ID wird jeder CPI-C-Aufruf innerhalb eines Programms eindeutig einer Conversation zugeordnet.

CPI-C

CPI-C (**C**ommon **P**rogramming **I**nterface for **C**ommunication) ist eine von X/Open und dem CIW (**C**PI-C **I**mplementor's **W**orkshop) normierte Programmschnittstelle für die Programm-Programm-Kommunikation in offenen Netzen. Das in openUTM implementierte CPI-C genügt der CPI-C V2.0 CAE Specification von X/Open. Die Schnittstelle steht in COBOL und C zur Verfügung.
CPI-C in openUTM kann über die Protokolle OSI TP, LU6.1, UPIC und mit openUTM-LU6.2 kommunizieren.

Cross Coupled System / XCS

Verbund von BS2000-Rechnern mit *Highly Integrated System Complex* Multiple System Control Facility (HIPLEX® MSCF).

Datenraum (BS2000-Systeme)
data space

Virtueller Adressraum des BS2000, der in seiner gesamten Größe vom Anwender genutzt werden kann.
In einem Datenraum können nur Daten und als Daten abgelegte Programme adressiert werden, es kann kein Programmcode zum Ablauf gebracht werden.

Dead Letter Queue
dead letter queue

Die Dead Letter Queue ist eine *TAC-Queue* mit dem festen Namen KDCDLETQ. Sie steht immer zur Verfügung, um Asynchron-Nachrichten an *Transaktionscodes*, TAC-Queues, LPAP- oder OSI-LPAP-Partner zu sichern, die nicht verarbeitet werden konnten.
Die Sicherung von Asynchron-Nachrichten in der Dead Letter Queue kann durch den Parameter DEAD-LETTER-Q der TAC-, LPAP- oder OSI-LPAP-Anweisung für jedes Nachrichtenziel einzeln ein- und ausgeschaltet werden.

DES

DES (Data Encryption Standard) ist eine internationale Norm zur Verschlüsselung von Daten. Bei diesem Verfahren wird ein Schlüssel zum Ver- und Entschlüsseln verwendet. Wird das DES-Verfahren verwendet, dann erzeugt der UPIC-Client für jede Sitzung einen DES-Schlüssel.

Dialog-Auftrag
dialog job, interactive job

Auftrag, der einen *Dialog-Vorgang* startet. Der Auftrag kann von einem *Client* oder - bei *Server-Server-Kommunikation* - von einer anderen Anwendung erteilt werden.

Dialog-Conversation
dialog conversation

CPI-C-Conversation, bei der sowohl der *Initiator* als auch der *Akzeptor* senden darf. Für den *Akzeptor* muss in der *UTM-Anwendung* ein Dialog-Transaktionscode generiert sein.

Dialog-Nachricht
dialog message

Nachricht, die eine Antwort erfordert oder selbst eine Antwort auf eine Anfrage ist. Dabei bilden Anfrage und Antwort einen *Dialog-Schritt*.

Dialog-Programm
dialog program

Teilprogramm, das einen *Dialog-Schritt* teilweise oder vollständig bearbeitet.

Dialog-Schritt
dialog step

Ein Dialog-Schritt beginnt mit dem Empfang einer *Dialog-Nachricht* durch die *UTM-Anwendung*. Er endet mit der Antwort der UTM-Anwendung.

Dialog-Terminalprozess (Unix-, Linux- und Windows-Systeme)
dialog terminal process

Ein Dialog-Terminalprozess verbindet ein Unix-, Linux- oder Windows-Terminal mit den *Workprozessen* der *UTM-Anwendung*. Dialog-Terminalprozesse werden entweder vom Benutzer durch Eingabe von utmdtp oder über die LOGIN-Shell gestartet. Für jedes Terminal, das an eine UTM-Anwendung angeschlossen werden soll, ist ein eigener Dialog-Terminalprozess erforderlich.

Dialog-Vorgang
dialog service

Vorgang , der einen *Auftrag* im Dialog (zeitlich gekoppelt) mit dem Auftraggeber (*Client* oder eine andere Server-Anwendung) bearbeitet. Ein Dialog-Vorgang verarbeitet *Dialog-Nachrichten* vom Auftraggeber und erzeugt Dialog-Nachrichten für diesen. Ein Dialog-Vorgang besteht aus mindestens einer *Transaktion.* Ein Dialog-Vorgang umfasst in der Regel mindestens einen *Dialog-Schritt*. Ausnahme: Bei *Vorgangskettung* können auch mehrere Vorgänge einen Dialog-Schritt bilden.

Dienst
service

Programm auf Windows-Systemen, das im Hintergrund unabhängig von angemeldeten Benutzern oder Fenstern abläuft.

Dienstzugriffspunkt
service access point

Im *OSI-Referenzmodell* stehen einer Schicht am Dienstzugriffspunkt die Leistungen der darunterliegenden Schicht zur Verfügung. Der Dienstzugriffspunkt wird im lokalen System durch einen *Selektor* identifiziert. Bei der Kommunikation bindet sich die *UTM-Anwendung* an einen Dienstzugriffspunkt. Eine Verbindung wird zwischen zwei Dienstzugriffspunkten aufgebaut.

Distributed Transaction Processing

X/Open-Architekturmodell für die transaktionsorientierte *verteilte Verarbeitung*.

Druckadministration
print administration

Funktionen zur *Drucksteuerung* und Administration von *Ausgabeaufträgen*, die an einen Drucker gerichtet sind.

Druckerbündel
printer pool

Mehrere Drucker, die demselben *LTERM-Partner* zugeordnet sind.

Druckergruppe (Unix- und Linux-Systeme)
printer group

Die Unix- oder Linux-Plattform richtet für jeden Drucker standardmäßig eine Druckergruppe ein, die genau diesen Drucker enthält. Darüber hinaus lassen sich mehrere Drucker einer Druckergruppe, aber auch ein Drucker mehreren Druckergruppen zuordnen.

Druckerprozess (Unix- und Linux-Systeme)
printer process

Prozess, der vom *Mainprozess* zur Ausgabe von *Asynchron-Nachrichten* an eine *Druckergruppe* eingerichtet wird. Er existiert, solange die Druckergruppe an die *UTM-Anwendung* angeschlossen ist. Pro angeschlossener Druckergruppe gibt es einen Druckerprozess.

Druckersteuerstation
printer control terminal

Begriff wurde ersetzt durch *Druckersteuer-LTERM*.

Druckersteuer-LTERM
printer control LTERM

Über ein Druckersteuer-LTERM kann sich ein *Client* oder ein Terminal-Benutzer an eine *UTM-Anwendung* anschließen. Von dem Client-Programm oder Terminal aus kann dann die *Administration* der Drucker erfolgen, die dem Druckersteuer-LTERM zugeordnet sind. Hierfür ist keine Administrationsberechtigung notwendig.

Drucksteuerung
print control

openUTM-Funktionen zur Steuerung von Druckausgaben.

Dynamische Konfiguration
dynamic configuration

Änderung der *Konfiguration* durch die Administration. Im laufenden Betrieb der Anwendung können UTM-Objekte wie z.B. *Teilprogramme*, *Transaktionscodes*, *Clients*, *LU6.1-Verbindungen*, Drucker oder *Benutzerkennungen* in die Konfiguration aufgenommen, modifiziert oder teilweise auch gelöscht werden. Hierzu können die Administrationsprogramme WinAdmin oder WebAdmin verwendet werden, oder es müssen eigene *Administrationsprogramme* erstellt werden, die die Funktionen der *Programmschnittstelle der Administration* nutzen.

Einschritt-Transaktion
single-step transaction

Transaktion, die genau einen *Dialog-Schritt* umfasst.

Einschritt-Vorgang
single-step service

Dialog-Vorgang, der genau einen *Dialog-Schritt* umfasst.

Ereignisgesteuerter Vorgang
event-driven service

Begriff ersetzt durch *Event-Service*.

Event-Exit
event exit

Routine des *Anwendungsprogramms*, das bei bestimmten Ereignissen (z.B. Start eines Prozesses, Ende eines Vorgangs) automatisch gestartet wird. Diese darf - im Gegensatz zu den *Event-Services* - keine KDCS-, CPI-C- und XATMI-Aufrufe enthalten.

Event-Funktion
event function

Oberbegriff für *Event-Exits* und *Event-Services*.

Event-Service
event service

Vorgang, der beim Auftreten bestimmter Ereignisse gestartet wird, z.B. bei bestimmten UTM-Meldungen. Die *Teilprogramme* ereignisgesteuerter Vorgänge müssen KDCS-Aufrufe enthalten.

Funktionseinheit, Functional Unit (FU)
functional unit

Teilmenge des *OSI-TP*-Protokolls, die eine bestimmte Funktionalität beinhaltet. Das OSI-TP-Protokoll ist in folgende Funktionseinheiten aufgeteilt:

- Dialogue
- Shared Control
- Polarized Control
- Handshake
- Commit
- Chained Transactions
- Unchained Transactions
- Recovery

Ein Hersteller, der OSI-TP implementiert, muss nicht alle Funktionseinheiten realisieren, sondern kann sich auf eine Teilmenge beschränken. Eine Kommunikation zwischen Anwendungen zweier unterschiedlicher OSI-TP-Implementierungen ist nur dann möglich, wenn die realisierten Funktionseinheiten zueinander passen.

Generierung
generation

siehe *UTM-Generierung*.

Globaler Sekundärer Speicherbereich/GSSB
global secondary storage area

siehe *Sekundärspeicherbereich.*

Hardcopy-Betrieb
hardcopy mode

Betriebsart eines Druckers, der lokal an ein Terminal angeschlossen ist. Dabei wird eine Nachricht, die auf dem Bildschirm angezeigt wird, zusätzlich auf dem Drucker abgedruckt.

Heterogene Kopplung
heterogeneous link

Bei *Server-Server-Kommunikation:* Kopplung einer *UTM-Anwendung* mit einer Nicht-UTM-Anwendung, z.B. einer CICS- oder TUXEDO-Anwendung.

Highly Integrated System Complex / HIPLEX ®

Produktfamilie zur Realisierung eines Bedien-, Last- und Verfügbarkeitsverbunds mit mehreren BS2000-Servern.

Hintergrund-Auftrag
background job

Hintergrund-Aufträge sind *Asynchron-Aufträge*, die an einen *Asynchron-Vorgang* der eigenen oder einer fernen Anwendung gerichtet sind. Hintergrund-Aufträge eignen sich besonders für zeitintensive oder zeitunkritische Verarbeitungen, deren Ergebnis keinen direkten Einfluss auf den aktuellen Dialog hat.

HIPLEX ® MSCF

(MSCF = **M**ultiple **S**ystem **C**ontrol **F**acility)
stellt bei HIPLEX® die Infrastruktur sowie Basisfunktionen für verteilte Anwendungen bereit.

Homogene Kopplung
homogeneous link

Bei *Server-Server-Kommunikation*: Kopplung von *UTM-Anwendungen.* Dabei spielt es keine Rolle, ob die Anwendungen auf der gleichen oder auf unterschiedlichen Betriebssystem-Plattformen ablaufen.

Inbound-Conversation (CPI-C)
inbound conversation

siehe *Incoming-Conversation.*

Incoming-Conversation (CPI-C)
incoming conversation

Eine *Conversation*, bei der das lokale CPI-C-Programm *Akzeptor* ist, heißt Incoming-Conversation. In der X/Open-Specification wird für Incoming-Conversation auch das Synonym Inbound-Conversation verwendet.

Initiale KDCFILE
initial KDCFILE

In einer *UTM-Cluster-Anwendung* die *KDCFILE*, die von *KDCDEF* erzeugt wurde und vor dem Start der Knoten-Anwendungen für jeden Knoten kopiert werden muss.

Initiator (CPI-C)
initiator

Die Kommunikationspartner einer *Conversation* werden Initiator und *Akzeptor* genannt. Der Initiator baut die Conversation mit den CPI-C-Aufrufen Initialize_Conversation und Allocate auf.

Insert
insert

Feld in einem Meldungstext, in das openUTM aktuelle Werte einträgt.

Inverser KDCDEF
inverse KDCDEF

Funktion, die aus den Konfigurationsdaten der *KDCFILE*, die im laufenden Betrieb dynamisch angepasst wurde, Steueranweisungen für einen *KDCDEF*-Lauf erzeugt. Der inverse KDCDEF kann "offline" unter KDCDEF oder "online" über die *Programmschnittstelle zur Administration* gestartet werden.

IUTMDB
IUTMDB

Schnittstelle für die koordinierte Zusammenarbeit mit externen Resource Managern auf BS2000-Systemen. Dazu gehören Datenhaltungssysteme (LEASY) und Datenbanksysteme (SESAM/SQL, UDS/SQL).

JConnect-Client
JConnect client

Bezeichung für Clients auf Basis des Produkts openUTM-JConnect. Die Kommunikation mit der UTM-Anwendung erfolgt über das *UPIC-Protokoll*.

JDK

Java Development Kit
Standard-Entwicklungsumgebung von Oracle Corporation für die Entwicklung von Java-Anwendungen.

Kaltstart
cold start

Starten einer *UTM-Anwendung* nach einer *normalen Beendigung* der Anwendung oder nach einer Neugenerierung (vgl. auch *Warmstart*).

KDCADM

Standard-Administrationsprogramm, das zusammen mit openUTM ausgeliefert wird. KDCADM stellt Administrationsfunktionen zur Verfügung, die über Transaktionscodes (*Administrationskommandos*) aufgerufen werden.

KDCDEF

UTM-Tool für die *Generierung* von *UTM-Anwendungen*. KDCDEF erstellt anhand der Konfigurationsinformationen in den KDCDEF-Steueranweisungen die UTM-Objekte *KDCFILE* und die ROOT-Tabellen-Source für die Main Routine *KDCROOT*.
In UTM-Cluster-Anwendungen erstellt KDCDEF zusätzlich die *Cluster-Konfigurationsdatei,* die *Cluster-User-Datei*, den *Cluster-Pagepool*, die *Cluster-GSSB-Datei* und die *Cluster-ULS-Datei*.

KDCFILE

Eine oder mehrere Dateien, die für den Ablauf einer *UTM-Anwendung* notwendige Daten enthalten. Die KDCFILE wird mit dem UTM-Generierungstool *KDCDEF* erstellt. Die KDCFILE enthält unter anderem die *Konfiguration* der Anwendung.

KDCROOT

Main Routine eines *Anwendungsprogramms*, die das Bindeglied zwischen *Teilprogrammen* und UTM-Systemcode bildet. KDCROOT wird zusammen mit den *Teilprogrammen* zum *Anwendungsprogramm* gebunden.

KDCS-Parameterbereich
KDCS parameter area

siehe *Parameterbereich.*

KDCS-Programmschnittstelle
KDCS program interface

Universelle UTM-Programmschnittstelle, die den nationalen Standard DIN 66 265 erfüllt und Erweiterungen enthält. Mit KDCS (Kompatible Datenkommunikationsschnittstelle) lassen sich z.B. Dialog-Services erstellen und *Message Queuing* Funktionen nutzen. Außerdem stellt KDCS Aufrufe zur *verteilten Verarbeitung* zur Verfügung.

Kerberos

Kerberos ist ein standardisiertes Netzwerk-Authentisierungsprotokoll (RFC1510), das auf kryptographischen Verschlüsselungsverfahren basiert, wobei keine Passwörter im Klartext über das Netzwerk gesendet werden.

Kerberos-Principal
Kerberos principal

Eigentümer eines Schlüssels.
Kerberos arbeitet mit symmetrischer Verschlüsselung, d.h. alle Schlüssel liegen an zwei Stellen vor, beim Eigentümer eines Schlüssels (Principal) und beim KDC (Key Distribution Center).

Keycode
key code

Code, der in einer Anwendung eine bestimmte Zugriffsberechtigung oder eine bestimmte Rolle repräsentiert. Mehrere Keycodes werden zu einem *Keyset* zusammengefasst.

Keyset
key set

Zusammenfassung von einem oder mehrerer *Keycodes* unter einem bestimmten Namen. Ein Keyset definiert Berechtigungen im Rahmen des verwendeten Berechtigungskonzepts (Lock-/Keycode-Konzept oder *Access-List* -Konzept).
Ein Keyset kann einer *Benutzerkennung* , einem *LTERM-Partner* , einem *(OSI-) LPAP-Partner* , einem *Service* oder einer *TAC-Queue* zugeordnet werden.

Knoten
node

Einzelner Rechner eines *Clusters*.

Knoten-Anwendung
node application

UTM-Anwendung, die als Teil einer *UTM-Cluster-Anwendung* auf einem einzelnen *Knoten* zum Ablauf kommt.

Knoten-Recovery
node recovery

Wenn für eine abnormal beendete Knoten-Anwendung zeitnah kein Warmstart auf ihrem eigenen *Knoten-Rechner* möglich ist, kann man für diesen Knoten auf einem anderen Knoten des UTM-Clusters eine Knoten-Recovery (Wiederherstellung) durchführen. Dadurch können Sperren, die von der ausgefallenen Knoten-Anwendung gehalten werden, freigegeben werden, um die laufende *UTM-Cluster-Anwendung* nicht unnötig zu beeinträchtigen.

Knotengebundener Vorgang
node bound service

Ein knotengebundener Vorgang eines Benutzers kann nur an der Knoten-Anwendung fortgesetzt werden, an der der Benutzer zuletzt angemeldet war. Folgende Vorgänge sind immer knotengebunden:

- Vorgänge, die eine Kommunikation mit einem Auftragnehmer über LU6.1 oder OSI TP begonnen haben und bei denen der Auftragnehmervorgang noch nicht beendet wurde
- eingeschobene Vorgänge einer Vorgangskellerung
- Vorgänge, die eine SESAM-Transaktion abgeschlossen haben

Außerdem ist der Vorgang eines Benutzers knotengebunden, solange der Benutzer an einer Knoten-Anwendung angemeldet ist.

Kommunikationsbereich/KB (KDCS)
communication area

Transaktionsgesicherter KDCS-*Primärspeicherbereich*, der Vorgangs-spezifische Daten enthält. Der Kommunikationsbereich besteht aus 3 Teilen:

- dem KB-Kopf mit allgemeinen Vorgangsdaten
- dem KB-Rückgabebereich für Rückgaben nach KDCS-Aufrufen
- dem KB-Programmbereich zur Datenübergabe zwischen UTM-Teilprogrammen innerhalb eines *Vorgangs*.

Kommunikationsendpunkt
communication end point

siehe *Transportsystem-Endpunkt*

Konfiguration
configuration

Summe aller Eigenschaften einer *UTM-Anwendung*. Die Konfiguration beschreibt:

- Anwendungs- und Betriebsparameter
- die Objekte der Anwendung und die Eigenschaften dieser Objekte. Objekte sind z.B. *Teilprogramme* und *Transaktionscodes*, Kommunikationspartner, Drucker, *Benutzerkennungen*
- definierte Zugriffsschutz- und Zugangsschutzmaßnahmen

Die Konfiguration einer UTM-Anwendung wird bei der UTM-Generierung festgelegt (*statische Konfiguration*) und kann per *Administration* dynamisch (während des Anwendungslaufs) geändert werden (*dynamische Konfiguration*). Die Konfiguration ist in der *KDCFILE* abgelegt.

Logging-Prozess
logging process

Prozess auf Unix-, Linux- und Windows-Systemen, der die Protokollierung von Abrechnungssätzen oder Messdaten steuert.

Logische Verbindung
virtual connection

Zuordnung zweier Kommunikationspartner.

Log4j

Log4j ist ein Teil des Apache Jakarta Projekts. Log4j bietet Schnittstellen zum Protokollieren von Informationen (Ablauf-Informationen, Trace-Records,...) und zum Konfigurieren der Protokoll-Ausgabe. *WS4UTM* verwendet das Softwareprodukt Log4j für die Trace- und Logging-Funktionalität.

Lockcode

Code, um einen LTERM-Partner oder einen Transaktionscode vor unberechtigtem Zugriff zu schützen. Damit ist ein Zugriff nur möglich, wenn das *Keyset* des Zugreifenden den passenden *Keycode* enthält (Lock-/Keycode-Konzept).

Lokaler Sekundärer Speicherbereich/LSSB
local secondary storage area

siehe *Sekundärspeicherbereich.*

LPAP-Bündel
LPAP bundle

LPAP-Bündel ermöglichen die Verteilung von Nachrichten an LPAP-Partner auf mehrere Partner-Anwendungen. Soll eine UTM-Anwendung sehr viele Nachrichten mit einer Partner-Anwendung austauschen, kann es für die Lastverteilung sinnvoll sein, mehrere Instanzen der Partner-Anwendung zu starten und die Nachrichten auf die einzelnen Instanzen zu verteilen. In einem LPAP-Bündel übernimmt openUTM die Verteilung der Nachrichten an die Instanzen der Partner-Anwendung. Ein LPAP-Bündel besteht aus einem Master-LPAP und mehreren Slave-LPAPs. Die Slave-LPAPs werden dem Master-LPAP bei der UTM-Generierung zugeordnet. LPAP-Bündel gibt es sowohl für das OSI TP-Protokoll als auch für das LU6.1-Protokoll.

LPAP-Partner
LPAP partner

Für die *verteilte Verarbeitung* über das *LU6.1*-Protokoll muss in der lokalen Anwendung für jede Partner-Anwendung ein LPAP-Partner konfiguriert werden. Der LPAP-Partner spiegelt in der lokalen Anwendung die Partner-Anwendung wider. Bei der Kommunikation wird die Partner-Anwendung nicht über ihren Anwendungsnamen oder ihre Adresse, sondern über den Namen des zugeordneten LPAP-Partners angesprochen.

LTERM-Bündel
LTERM bundle

Ein LTERM-Bündel (Verbindungsbündel) besteht aus einem Master-LTERM und mehreren Slave-LTERMs. Mit einem LTERM-Bündel (Verbindungsbündel) verteilen Sie asynchrone Nachrichten an eine logische Partner-Anwendung gleichmäßig auf mehrere parallele Verbindungen.

LTERM-Gruppe
LTERM group

Eine LTERM-Gruppe besteht aus einem oder mehreren Alias-LTERMs, den Gruppen-LTERMs, und einem Primary-LTERM. In einer LTERM-Gruppe ordnen Sie mehrere LTERMs einer Verbindung zu.

LTERM-Partner
LTERM partner

Um *Clients* oder Drucker an eine *UTM-Anwendung* anschließen zu können, müssen in der Anwendung LTERM-Partner konfiguriert werden. Ein Client oder Drucker kann nur angeschlossen werden, wenn ihm ein LTERM-Partner mit entsprechenden Eigenschaften zugeordnet ist. Diese Zuordnung wird i.A. in der *Konfiguration* festgelegt, sie kann aber auch dynamisch über Terminal-Pools erfolgen.

LTERM-Pool
LTERM pool

Statt für jeden *Client* eine LTERM- und eine PTERM-Anweisung anzugeben, kann mit der Anweisung TPOOL ein Pool von LTERM-Partnern definiert werden. Schließt sich ein Client über einen LTERM-Pool an, wird ihm dynamisch ein LTERM-Partner aus dem Pool zugeordnet.

LU6.1

Geräteunabhängiges Datenaustauschprotokoll (Industrie-Standard) für die transaktionsgesicherte *Server-Server-Kommunikation*.

LU6.1-LPAP-Bündel
LU6.1-LPAP bundle

LPAP-Bündel für *LU6.1*-Partner-Anwendungen.

LU6.1-Partner
LU6.1 partner

Partner der *UTM-Anwendung*, der mit der UTM-Anwendung über das Protokoll *LU6.1* kommuniziert. Beispiele für solche Partner sind:

- eine UTM-Anwendung, die über LU6.1 kommuniziert
- eine Anwendung im IBM-Umfeld (z.B. CICS, IMS oder TXSeries), die über LU6.1 kommuniziert

Mainprozess (Unix-, Linux- und Windows-Systeme)
main process

Prozess, der die *UTM-Anwendung* startet. Er startet die *Workprozesse*, die *UTM-System-Prozesse*, *Druckerprozesse, Netzprozesse, Logging-Prozess* und den *Timerprozess* und überwacht die *UTM-Anwendung*.

Main Routine KDCROOT
main routine KDCROOT

siehe *KDCROOT*.

Management Unit
management unit

Komponente des *SE Servers*; ermöglicht mit Hilfe des *SE Managers* ein zentrales, web-basiertes Management aller Units eines SE Servers.

Meldung / UTM-Meldung
UTM message

Meldungen werden vom Transaktionsmonitor openUTM oder von UTM-Tools (wie z.B. *KDCDEF*) an *Meldungsziele* ausgegeben. Eine Meldung besteht aus einer Meldungsnummer und dem Meldungstext, der ggf. *Inserts* mit aktuellen Werten enthält. Je nach Meldungsziel werden entweder die gesamte Meldung oder nur Teile der Meldung (z.B. nur die Inserts) ausgegeben.

Meldungsdefinitionsdatei
message definition file

Die Meldungsdefinitionsdatei wird mit openUTM ausgeliefert und enthält standardmäßig die UTM-Meldungstexte in deutscher und englischer Sprache und die Definitionen der Meldungseigenschaften. Aufbauend auf diese Datei kann der Anwender auch eigene, individuelle Meldungsmodule erzeugen.

Meldungsziel
message destination

Ausgabemedium für eine *Meldung*. Mögliche Meldungsziele von Meldungen des Transaktionsmonitors openUTM sind z.B. Terminals, *TS-Anwendungen*, der *Event-Service* MSGTAC, die *System-Protokolldatei* SYSLOG oder *TAC-Queues, Asynchron-TACs, USER-Queues*, SYSOUT/SYSLST bzw. stderr/stdout. Meldungsziele von Meldungen der UTM-Tools sind SYSOUT /SYSLST bzw. stderr/stdout.

Mehrschritt-Transaktion
multi-step transaction

Transaktion, die aus mehr als einem *Verarbeitungsschritt* besteht.

Mehrschritt-Vorgang (KDCS)
multi-step service

Vorgang, der in mehreren *Dialog-Schritten* ausgeführt wird.

Message Queuing
message queuing

Message Queuing (MQ) ist eine Form der Kommunikation, bei der die Nachrichten (Messages) nicht unmittelbar, sondern über zwischengeschaltete *Message Queues* ausgetauscht werden. Sender und Empfänger können zeitlich und räumlich entkoppelt ablaufen. Die Übermittlung der Nachricht hängt nicht davon ab, ob gerade eine Netzverbindung besteht oder nicht. Bei openUTM gibt es *UTM-gesteuerte Queues* und *Service-gesteuerte Queues*.

Message Queue
message queue

Warteschlange, in der bestimmte Nachrichten transaktionsgesichert bis zur Weiterverarbeitung eingereiht werden. Je nachdem, wer die Weiterverarbeitung kontrolliert, unterscheidet man *Service-gesteuerte Queues* und *UTM-gesteuerte Queues*.

MSGTAC
MSGTAC

Spezieller Event-Service, der Meldungen mit dem Meldungsziel MSGTAC per Programm verarbeitet. MSGTAC ist ein Asynchron-Vorgang und wird vom Betreiber der Anwendung erstellt.

Multiplex-Verbindung (BS2000-Systeme)
multiplex connection

Spezielle Möglichkeit, die *OMNIS* bietet, um Terminals an eine *UTM-Anwendung* anzuschließen. Eine Multiplex-Verbindung ermöglicht es, dass sich mehrere Terminals eine *Transportverbindung* teilen.

Nachrichten-Bereich/NB (KDCS)
KDCS message area

Bei KDCS-Aufrufen: Puffer-Bereich, in dem Nachrichten oder Daten für openUTM oder für das *Teilprogramm* bereitgestellt werden.

Network File System/Service / NFS

Ermöglicht den Zugriff von Unix- und Linux-Rechnern auf Dateisysteme über das Netzwerk.

Netzprozess (Unix-, Linux- und Windows-Systeme)
net process

Prozess einer *UTM-Anwendung* zur Netzanbindung.

Netzwerk-Selektor
network selector

Der Netzwerk-Selektor identifiziert im lokalen System einen *Dienstzugriffspunkt* zur Vermittlungsschicht des *OSI-Referenzmodells*.

Normale Beendigung einer UTM-Anwendung
normal termination of a UTM application

Kontrollierte Beendigung einer *UTM-Anwendung*; das bedeutet u.a., dass die Verwaltungsdaten auf der *KDCFILE* aktualisiert werden. Eine normale Beendigung veranlasst der *Administrator* (z.B. mit KDCSHUT N). Den Start nach einer normalen Beendigung führt openUTM als *Kaltstart* durch.

Object Identifier
object identifier

Ein Object Identifier ist ein weltweit eindeutiger Bezeichner für Objekte im OSI-Umfeld. Ein Object Identifier besteht aus einer Folge von ganzen Zahlen, die einen Pfad in einer Baumstruktur repräsentiert.

Offener Terminalpool
open terminal pool

Terminalpool, der nicht auf *Clients* eines Rechners oder eines bestimmten Typs beschränkt ist. An diesen Terminalpool können sich alle Clients anschließen, für die kein Rechner- oder Typ-spezifischer Terminalpool generiert ist.

OMNIS (BS2000-Systeme)
OMNIS

OMNIS ist ein „Session-Manager“ auf einem BS2000-System, der die gleichzeitige Verbindungsaufnahme von einem Terminal zu mehreren Partnern in einem Netzwerk ermöglicht. OMNIS ermöglicht es außerdem, mit *Multiplex-Verbindungen* zu arbeiten.

Online-Import
online import

Als Online-Import wird in einer *UTM-Cluster-Anwendung* das Importieren von Anwendungsdaten aus einer normal beendeten Knoten-Anwendung in eine laufende Knoten-Anwendung bezeichnet.

Online-Update
online update

Als Online-Update wird in einer *UTM-Cluster-Anwendung* die Änderung der Konfiguration der Anwendung oder des Anwendungsprogramms oder der Einsatz einer neuen UTM-Korrekturstufe bei laufender *UTM-Cluster-Anwendung* bezeichnet.

OpenCPIC

Trägersystem für UTM-Clients, die das *OSI TP* Protokoll verwenden.

OpenCPIC-Client
OpenCPIC client

OSI TP Partner-Anwendungen mit Trägersystem *OpenCPIC*.

openSM2

Die Produktlinie openSM2 ist eine einheitliche Lösung für das unternehmensweite Performance Management von Server- und Speichersystemen. openSM2 bietet eine Messdatenerfassung, Online-Überwachung und Offline-Auswertung.

openUTM-Cluster
openUTM cluster

aus der Sicht von UPIC-Clients, **nicht** aus Server-Sicht:
Zusammenfassung mehrerer Knoten-Anwendungen einer UTM-Cluster-Anwendung zu einer logischen Anwendung, die über einen gemeinsamen Symbolic Destination Name adressiert wird.

openUTM-D

openUTM-D (openUTM-Distributed) ist eine openUTM-Komponente, die *verteilte Verarbeitung* ermöglicht. openUTM-D ist integraler Bestandteil von openUTM.

OSI-LPAP-Bündel
OSI-LPAP bundle

LPAP-Bündel für *OSI TP*-Partner-Anwendungen.

OSI-LPAP-Partner
OSI-LPAP partner

OSI-LPAP-Partner sind die bei openUTM generierten Adressen der *OSI TP-Partner*. Für die *verteilte Verarbeitung* über das Protokoll *OSI TP* muss in der lokalen Anwendung für jede Partner-Anwendung ein OSI-LPAP-Partner konfiguriert werden. Der OSI-LPAP-Partner spiegelt in der lokalen Anwendung die Partner-Anwendung wider. Bei der Kommunikation wird die Partner-Anwendung nicht über ihren Anwendungsnamen oder ihre Adresse, sondern über den Namen des zugeordneten OSI-LPAP-Partners angesprochen.

OSI-Referenzmodell
OSI reference model

Das OSI-Referenzmodell stellt einen Rahmen für die Standardisierung der Kommunikation von offenen Systemen dar. ISO, die Internationale Organisation für Standardisierung, hat dieses Modell im internationalen Standard
ISO IS7498 beschrieben. Das OSI-Referenzmodell unterteilt die für die Kommunikation von Systemen notwendigen Funktionen in sieben logische Schichten. Diese Schichten haben jeweils klar definierte Schnittstellen zu den benachbarten Schichten.

OSI TP

Von der ISO definiertes Kommunikationsprotokoll für die verteilte Transaktionsverarbeitung. OSI TP steht für Open System Interconnection Transaction Processing.

OSI TP-Partner
OSI TP partner

Partner der UTM-Anwendung, der mit der UTM-Anwendung über das OSI TP-Protokoll kommuniziert.
Beispiele für solche Partner sind:

- eine UTM-Anwendung, die über OSI TP kommuniziert
- eine Anwendung im IBM-Umfeld (z.B. CICS), die über openUTM-LU62 angeschlossen ist
- ein *OpenCPIC-Client*
- Anwendungen anderer TP-Monitore, die OSI TP unterstützen

Outbound-Conversation (CPI-C)
outbound conversation

siehe *Outgoing-Conversation*.

Outgoing-Conversation (CPI-C)
outgoing conversation

Eine Conversation, bei der das lokale CPI-C-Programm der *Initiator* ist, heißt Outgoing-Conversation. In der X/Open-Specification wird für Outgoing-Conversation auch das Synonym Outbound-Conversation verwendet.

Pagepool
page pool

Teil der *KDCFILE*, in dem Anwenderdaten gespeichert werden.
In einer *stand-alone Anwendung* sind dies z.B. *Dialog-Nachrichten*, Nachrichten an *Message Queues*, *Sekundärspeicherbereiche*.
In einer *UTM-Cluster-Anwendung* sind dies z.B. Nachrichten an *Message Queues, TLS*.

Parameterbereich
parameter area

Datenstruktur, in der ein *Teilprogramm* bei einem UTM-Aufruf die für diesen Aufruf notwendigen Operanden an openUTM übergibt.

Partner-Anwendung
partner application

Partner einer UTM-Anwendung bei *verteilter Verarbeitung*. Für die verteilte Verarbeitung werden höhere Kommunikationsprotokolle verwendet (*LU6.1*, *OSI TP* oder LU6.2 über das Gateway openUTM-LU62).

Postselection (BS2000-Systeme)
postselection

Auswahl der protokollierten UTM-Ereignisse aus der SAT-Protokolldatei, die ausgewertet werden sollen. Die Auswahl erfolgt mit Hilfe des Tools SATUT.

Programmraum (BS2000-Systeme)
program space

In Speicherklassen aufgeteilter virtueller Adressraum des BS2000, in dem sowohl ablauffähige Programme als auch reine Daten adressiert werden.

Prepare to commit (PTC)
prepare to commit

Bestimmter Zustand einer verteilten Transaktion:
Das Transaktionsende der verteilten Transaktion wurde eingeleitet, es wird jedoch noch auf die Bestätigung des Transaktionsendes durch den Partner gewartet.

Preselection (BS2000-Systeme)
preselection

Festlegung der für die *SAT-Beweissicherung* zu protokollierenden UTM-Ereignisse. Die Preselection erfolgt durch die UTM-SAT-Administration. Man unterscheidet Ereignis-spezifische, Benutzer-spezifische und Auftrags-(TAC-)spezifische Preselection.

Presentation-Selektor
presentation selector

Der Presentation-Selektor identifiziert im lokalen System einen *Dienstzugriffspunkt* zur Darstellungsschicht des *OSI-Referenzmodells*.

Primärspeicherbereich
primary storage area

Bereich im Arbeitsspeicher, auf den das *KDCS-Teilprogramm* direkt zugreifen kann, z.B. *Standard Primärer Arbeitsbereich*, *Kommunikationsbereich*.

Printerprozess (Unix- und Linux-Systeme)
printer process

siehe *Druckerprozess*.

Programmschnittstelle zur Administration
program interface for administration

UTM-Programmschnittstelle, mit deren Hilfe der Anwender eigene *Administrationsprogramme* erstellen kann. Die Programmschnittstelle zur Administration bietet u.a. Funktionen zur *dynamischen Konfiguration*, zur Modifikation von Eigenschaften und Anwendungsparametern und zur Abfrage von Informationen zur *Konfiguration* und zur aktuellen Auslastung der Anwendung.

Prozess
prozess

In den openUTM-Handbüchern wird der Begriff "Prozess" als Oberbegriff für Prozess (Unix-, Linux- und Windows-Systeme) und Task (BS2000-Systeme) verwendet.

Queue
queue

siehe *Message Queue*

Quick Start Kit

Beispielanwendung, die mit openUTM (Windows-Systeme) ausgeliefert wird.

Quittungs-Auftrag
confirmation job

Bestandteil eines *Auftrags-Komplexes*, worin der Quittungs-Auftrag dem *Basis-Auftrag* zugeordnet ist. Es gibt positive und negative Quittungsaufträge. Bei positivem Ergebnis des *Basis-Auftrags* wird der positive Quittungs-Auftrag wirksam, sonst der negative.

Redelivery
redelivery

Erneutes Zustellen einer *Asynchron-Nachricht*, nachdem diese nicht ordnungsgemäß verarbeitet werden konnte, z.B. weil die *Transaktion* zurückgesetzt oder der *Asynchron-Vorgang* abnormal beendet wurde. Die Nachricht wird wieder in die Message Queue eingereiht und lässt sich damit erneut lesen und/oder verarbeiten.

Reentrant-fähiges Programm
reentrant program

Programm, dessen Code durch die Ausführung nicht verändert wird.
Auf BS2000-Systemen ist dies Voraussetzung dafür, *Shared Code* zu nutzen.

Request
request

Anforderung einer *Service-Funktion* durch einen *Client* oder einen anderen Server.

Requestor
requestor

In XATMI steht der Begriff Requestor für eine Anwendung, die einen Service aufruft.

Resource Manager
resource manager

Resource Manager (RMs) verwalten Datenressourcen. Ein Beispiel für RMs sind Datenbank-Systeme. openUTM stellt aber auch selbst Resource Manager zur Verfügung, z.B. für den Zugriff auf *Message Queues*, lokale Speicherbereiche und Logging-Dateien. Anwendungsprogramme greifen auf RMs über RM-spezifische Schnittstellen zu. Für Datenbank-Systeme ist dies meist SQL, für die openUTM-RMs die Schnittstelle KDCS.

RFC1006

Von IETF (Internet Engineering Task Force) definiertes Protokoll der TCP/IP-Familie zur Realisierung der ISO-Transportdienste (Transportklasse 0) auf TCP/IP-Basis.

RSA

Abkürzung für die Erfinder des RSA-Verschlüsselungsverfahrens Rivest, Shamir und Adleman. Bei diesem Verfahren wird ein Schlüsselpaar verwendet, das aus einem öffentlichen und einem privaten Schlüssel besteht. Eine Nachricht wird mit dem öffentlichen Schlüssel verschlüsselt und kann nur mit dem privaten Schlüssel entschlüsselt werden. Das RSA-Schlüsselpaar wird von der UTM-Anwendung erzeugt.

SAT-Beweissicherung (BS2000-Systeme)
SAT audit

Beweissicherung durch die Komponente SAT (Security Audit Trail) des BS2000-Softwareproduktes SECOS.

SE Manager
SE manager

Web-basierte Benutzeroberfläche (GUI) für Business Server der SE Serie. Der SE Manager läuft auf der *Management Unit* und ermöglicht die zentrale Bedienung und Verwaltung von Server Units (mit /390-Architektur und/oder x86-Architektur), Application Units (x86-Architektur), Net Unit und der Peripherie.

SE Server
SE server

Ein Business Server der SE Serie von Fujitsu.

Sekundärspeicherbereich
secondary storage area

Transaktionsgesicherter Speicherbereich, auf den das KDCS-*Teilprogramm* mit speziellen Aufrufen zugreifen kann. Lokale Sekundärspeicherbereiche (LSSB) sind einem *Vorgang* zugeordnet, auf globale Sekundärspeicherbereiche (GSSB) kann von allen Vorgängen einer *UTM-Anwendung* zugegriffen werden. Weitere Sekundärspeicherbereiche sind der *Terminal-spezifische Langzeitspeicher (TLS)* und der *User-spezifische Langzeitspeicher (ULS)* .

Selektor
selector

Ein Selektor identifiziert im lokalen System einen *Zugriffspunkt* auf die Dienste einer Schicht des *OSI-Referenzmodells*. Jeder Selektor ist Bestandteil der Adresse des Zugriffspunktes.

Semaphor (Unix-, Linux- und Windows-Systeme)
semaphore

Betriebsmittel auf Unix-, Linux- und Windows-Systemen, das zur Steuerung und Synchronisation von Prozessen dient.

Server
server

Ein Server ist eine *Anwendung*, die *Services* zur Verfügung stellt. Oft bezeichnet man auch den Rechner, auf dem Anwendungen laufen, als Server.

Server-Seite einer Conversation (CPI-C)
server side of a conversation

Begriff ersetzt durch *Akzeptor*.

Server-Server-Kommunikation
server-server communication

siehe *verteilte Verarbeitung*.

Service Access Point

siehe *Dienstzugriffspunkt.*

Service
service

Services bearbeiten die Aufträge, die an eine Server-Anwendung geschickt werden. Ein Service in einer UTM-Anwendung wird auch Vorgang genannt und setzt sich aus einer oder mehreren Transaktionen zusammen. Ein Service wird über den Vorgangs-TAC aufgerufen. Services können von Clients oder anderen Services angefordert werden.

Service-gesteuerte Queue
service controlled queue

Message Queue, bei der der Abruf und die Weiterverarbeitung der Nachrichten durch Services gesteuert werden. Ein Service muss zum Lesen der Nachricht explizit einen KDCS-Aufruf (DGET) absetzen.
Service-gesteuerte Queues gibt es bei openUTM in den Varianten USER-Queue, TAC-Queue und Temporäre Queue.

Service Routine
service routine

siehe Teilprogramm.

Session
session

Kommunikationsbeziehung zweier adressierbarer Einheiten im Netz über das SNA-Protokoll LU6.1.

Session-Selektor
session selector

Der Session-Selektor identifiziert im lokalen System einen Zugriffspunkt zu den Diensten der Kommunikationssteuerschicht (Session-Layer) des OSI-Referenzmodells.

Shared Code (BS2000-Systeme)
shared code

Code, der von mehreren Prozessen gemeinsam benutzt werden kann.

Shared Memory
shared memory

Virtueller Speicherbereich, auf den mehrere Prozesse gleichzeitig zugreifen können.

Shared Objects (Unix-, Linux- und Windows-Systeme)
shared objects

Teile des Anwendungsprogramms können als Shared Objects erzeugt werden. Diese werden dynamisch zur Anwendung dazugebunden und können im laufenden Betrieb ausgetauscht werden. Shared Objects werden mit der KDCDEF-Anweisung SHARED-OBJECT definiert.

Sicherungspunkt
synchronization point, consistency point

Ende einer Transaktion. Zu diesem Zeitpunkt werden alle in der Transaktion vorgenommenen Änderungen der Anwendungsinformation gegen Systemausfall gesichert und für andere sichtbar gemacht. Während der Transaktion gesetzte Sperren werden wieder aufgehoben.

Single System Image

Unter single system image versteht man die Eigenschaft eines Clusters, nach außen hin als ein einziges, in sich geschlossenes System zu erscheinen. Die heterogene Natur des Clusters und die interne Verteilung der Ressourcen im Cluster ist für die Benutzer des Clusters und die Anwendungen, die mit dem Cluster kommunizieren, nicht sichtbar.

SOA

SOA (Service-oriented architecture).
SOA ist ein Konzept für eine Systemarchitektur, in dem Funktionen in Form von wieder verwendbaren, technisch voneinander unabhängigen und fachlich lose gekoppelten Services implementiert werden. Services können unabhängig von zugrunde liegenden Implementierungen über Schnittstellen aufgerufen werden, deren Spezifikationen öffentlich und damit vertrauenswürdig sein können. Service-Interaktion findet über eine dafür vorgesehene Kommunikationsinfrastruktur statt.

SOAP

SOAP (Simple Object Access Protocol) ist ein Protokoll, mit dessen Hilfe Daten zwischen Systemen ausgetauscht und Remote Procedure Calls durchgeführt werden können. SOAP stützt sich auf die Dienste anderer Standards, XML zur Repräsentation der Daten und Internet-Protokolle der Transport- und Anwendungsschicht zur Übertragung der Nachrichten.

Socket-Verbindung
socket connection

Transportsystem-Verbindung, die die Socket-Schnittstelle verwendet. Die Socket-Schnittstelle ist eine Standard-Programmschnittstelle für die Kommunikation über TCP/IP.

Stand-alone Anwendung
stand-alone application

siehe stand-alone UTM-Anwendung.

Stand-alone UTM-Anwendung
stand-alone UTM application

Herkömmliche UTM-Anwendung, die nicht Bestandteil einer UTM-Cluster-Anwendung ist.

Standard Primärer Arbeitsbereich/SPAB (KDCS)
standard primary working area

Bereich im Arbeitsspeicher, der jedem KDCS-Teilprogramm zur Verfügung steht. Sein Inhalt ist zu Beginn des Teilprogrammlaufs undefiniert oder mit einem Füllzeichen vorbelegt.

Startformat
start format

Format, das openUTM am Terminal ausgibt, wenn sich ein Benutzer erfolgreich bei der UTM-Anwendung angemeldet hat (ausgenommen nach Vorgangs-Wiederanlauf und beim Anmelden über Anmelde-Vorgang).

Statische Konfiguration
static configuration

Festlegen der Konfiguration bei der UTM-Generierung mit Hilfe des UTM-Tools KDCDEF.

SYSLOG-Datei
SYSLOG file

siehe System-Protokolldatei.

System-Protokolldatei
system log file

Datei oder Dateigeneration, in die openUTM während des Laufs einer UTM-Anwendung alle UTM-Meldungen protokolliert, für die das Meldungsziel SYSLOG definiert ist.

TAC
TAC

siehe Transaktionscode.

TAC-Queue
TAC queue

Message Queue, die explizit per KDCDEF-Anweisung generiert wird. Eine TAC-Queue ist eine Service-gesteuerte Queue und kann unter dem generierten Namen von jedem Service aus angesprochen werden.

Teilprogramm
program unit

UTM-Services werden durch ein oder mehrere Teilprogramme realisiert. Die Teilprogramme sind Bestandteile des Anwendungsprogramms. Abhängig vom verwendeten API müssen sie KDCS-, XATMI- oder CPIC-Aufrufe enthalten. Sie sind über Transaktionscodes ansprechbar. Einem Teilprogramm können mehrere Transaktionscodes zugeordnet werden.

Temporäre Queue
temporary queue

Message Queue, die dynamisch per Programm erzeugt wird und auch wieder per Programm gelöscht werden kann, vgl. Service-gesteuerte Queue.

Terminal-spezifischer Langzeitspeicher/TLS (KDCS)
terminal-specific long-term storage

Sekundärspeicher, der einem LTERM-, LPAP- oder OSI-LPAP-Partner zugeordnet ist und über das Anwendungsende hinaus erhalten bleibt.

Timerprozess (Unix-, Linux- und Windows-Systeme)
timer process

Prozess, der Aufträge zur Zeitüberwachung von Workprozessen entgegennimmt, sie in ein Auftragsbuch einordnet und nach einer im Auftragsbuch festgelegten Zeit den Workprozessen zur Bearbeitung wieder zustellt.

TLS Termination Proxy
TLS termination proxy

Ein TLS-Terminierungsproxy ist ein Proxy-Server, der verwendet wird, um eingehende TLS-Verbindungen zu verarbeiten, die Daten zu entschlüsseln und die unverschlüsselte Anforderung an andere Server weiterzugeben.

TNS (Unix-, Linux- und Windows-Systeme)

Abkürzung für den Transport Name Service, der einem Anwendungsnamen einen Transport-Selektor und das Transportsystem zuordnet, über das die Anwendung erreichbar ist.

Tomcat

siehe Apache Tomcat

Transaktion
transaction

Verarbeitungsabschnitt innerhalb eines Services, für den die Einhaltung der ACID-Eigenschaften garantiert wird. Von den in einer Transaktion beabsichtigten Änderungen der Anwendungsinformation werden entweder alle konsistent durchgeführt oder es wird keine durchgeführt (Alles-oder-Nichts Regel). Das Transaktionsende bildet einen Sicherungspunkt.

Transaktionscode/TAC
transaction code

Name, über den ein Teilprogramm aufgerufen werden kann. Der Transaktionscode wird dem Teilprogramm bei der statischen oder dynamischen Konfiguration zugeordnet. Einem Teilprogramm können auch mehrere Transaktionscodes zugeordnet werden.

Transaktionsrate
transaction rate

Anzahl der erfolgreich beendeten Transaktionen pro Zeiteinheit.

Transfer-Syntax
transfer syntax

Bei OSI TP werden die Daten zur Übertragung zwischen zwei Rechnersystemen von der lokalen Darstellung in die Transfer-Syntax umgewandelt. Die Transfer-Syntax beschreibt die Daten in einem neutralen Format, das von allen beteiligten Partnern verstanden wird. Jeder Transfer-Syntax muss ein Object Identifier zugeordnet sein.

Transport Layer Security
transport layer security

Der Transport Layer Security, ist ein hybrides Verschlüsselungsprotokoll zur sicheren Datenübertragung im Internet.

Transport-Selektor
transport selector

Der Transport-Selektor identifiziert im lokalen System einen Dienstzugriffspunkt zur Transportschicht des OSI-Referenzmodells.

Transportsystem-Anwendung
transport system application

Anwendung, die direkt auf einer Transportsystem-Schnittstelle wie z.B. CMX, DCAM oder Socket aufsetzt. Für den Anschluss von Transportsystem-Anwendungen muss bei der Konfiguration als Partnertyp APPLI oder SOCKET angegeben werden. Eine Transportsystem-Anwendung kann nicht in eine Verteilte Transaktion eingebunden werden.

Transportsystem-Endpunkt
transport system end point

Bei der Client-/Server- oder Server-/Server-Kommunikation wird eine Verbindung zwischen zwei Transportsystem-Endpunkten aufgebaut. Ein Transportsystem-Endpunkt wird auch als lokaler Anwendungsname bezeichnet und wird mit der Anweisung BCAMAPPL oder mit MAX APPLINAME definiert.

Transportsystem-Zugriffspunkt
transport system access point

siehe Transportsystem-Endpunkt.

Transportverbindung
transport connection

Im OSI-Referenzmodell eine Verbindung zwischen zwei Instanzen der Schicht 4 (Transportschicht).

TS-Anwendung
TS application

siehe Transportsystem-Anwendung.

Typisierter Puffer (XATMI)
typed buffer

Puffer für den Austausch von typisierten und strukturierten Daten zwischen Kommunikationspartnern. Durch diese typisierten Puffer ist die Struktur der ausgetauschten Daten den Partnern implizit bekannt.

UPIC

Trägersystem für UTM-Clients. UPIC steht für Universal Programming Interface for Communication. Die Kommunikation mit der UTM-Anwendung erfolgt über das UPIC-Protokoll.

UPIC-Client

Bezeichnung für UTM-Clients mit Trägersystem UPIC und JConnect-Clients.

UPIC-Protokoll
Upic protocol

Protokoll für die Client-Server-Kommunikation mit UTM-Anwendungen. Das UPIC-Protokoll wird von UPIC-Clients und von JConnect-Clients verwendet.

UPIC Analyzer

Komponente zur Analyse der mit UPIC Capture mitgeschnittenen UPIC-Kommunikation. Dieser Schritt dient dazu, den Mitschnitt für das Abspielen mit UPIC Replay aufzubereiten.

UPIC Capture

Mitschneiden der Kommunikation zwischen UPIC-Clients und UTM-Anwendungen, um sie zu einem späteren Zeitpunkt abspielen zu können
(UPIC Replay).

UPIC Replay

Komponente zum Abspielen der mit UPIC Capture mitgeschnittenen und mit UPIC Analyzer aufbereiteten UPIC-Kommunikation.

USER-Queue
USER queue

Message Queue, die openUTM jeder Benutzerkennung zur Verfügung stellt. Eine USER-Queue zählt zu den Service-gesteuerten Queues und ist immer der jeweiligen Benutzerkennung zugeordnet. Der Zugriff von fremden UTM-Benutzern auf die eigene USER-Queue kann eingeschränkt werden.

User-spezifischer Langzeitspeicher/ULS
user-specific long-term storage

Sekundärspeicher, der einer Benutzerkennung, einer Session oder einer Association zugeordnet ist und über das Anwendungsende hinaus erhalten bleibt.

USLOG-Datei
USLOG file

siehe Benutzer-Protokolldatei.

UTM-Anwendung
UTM application

Eine UTM-Anwendung stellt Services zur Verfügung, die Aufträge von Clients oder anderen Anwendungen bearbeiten. openUTM übernimmt dabei u.a. die Transaktionssicherung und das Management der Kommunikations- und Systemressourcen. Technisch gesehen ist eine UTM-Anwendung eine Prozessgruppe, die zur Laufzeit eine logische Server-Einheit bildet.

UTM-Client
UTM client

siehe Client.

UTM-Cluster-Anwendung
UTM cluster application

UTM-Anwendung, die für den Einsatz in einem Cluster generiert ist und die man logisch als **eine** Anwendung betrachten kann.
Physikalisch gesehen besteht eine UTM-Cluster-Anwendung aus mehreren, identisch generierten UTM-Anwendungen, die auf den einzelnen Knoten laufen.

UTM-Cluster-Dateien
UTM cluster files

Oberbegriff für alle Dateien, die für den Ablauf einer UTM-Cluster-Anwendung auf Unix-, Linux- und Windows-Systemen benötigt werden. Dazu gehören folgende Dateien:

- Cluster-Konfigurationsdatei
- Cluster-User-Datei
- Dateien des Cluster-Pagepool
- Cluster-GSSB-Datei
- Cluster-ULS-Datei
- Dateien des Cluster-Administrations-Journals*
- Cluster-Lock-Datei*
- Lock-Datei zur Start-Serialisierung*

Die mit * gekennzeichneten Dateien werden beim Start der ersten Knoten-Anwendung angelegt, alle anderen Dateien werden bei der Generierung mit KDCDEF erzeugt.

UTM-D

siehe openUTM-D.

UTM-Datenstation
UTM terminal

Begriff ersetzt durch LTERM-Partner.

UTM-F

UTM-Anwendungen können als UTM-F-Anwendungen (UTM-Fast) generiert werden. Bei UTM-F wird zugunsten der Performance auf Platteneingaben/-ausgaben verzichtet, mit denen bei UTM-S die Sicherung von Benutzer- und Transaktionsdaten durchgeführt wird. Gesichert werden lediglich Änderungen der Verwaltungsdaten.
In UTM-Cluster-Anwendungen, die als UTM-F-Anwendung generiert sind (APPLIMODE=FAST), werden Cluster-weit gültige Anwenderdaten auch gesichert. Dabei werden GSSB- und ULS-Daten genauso behandelt wie in UTM-Cluster-Anwendungen, die mit UTM-S generiert sind. Vorgangs-Daten von Benutzern mit RESTART=YES werden jedoch nur beim Abmelden des Benutzers anstatt bei jedem Transaktionsende geschrieben.

UTM-Generierung
UTM generation

Statische Konfiguration einer UTM-Anwendung mit dem UTM-Tool KDCDEF und Erzeugen des Anwendungsprogramms.

UTM-gesteuerte Queues
UTM controlled queue

Message Queues, bei denen der Abruf und die Weiterverarbeitung der Nachrichten vollständig durch openUTM gesteuert werden. Siehe auch Asynchron-Auftrag, Hintergrund-Auftrag und Asynchron-Nachricht.

UTM-S

Bei UTM-S-Anwendungen sichert openUTM neben den Verwaltungsdaten auch alle Benutzerdaten über ein Anwendungsende und einen Systemausfall hinaus. Außerdem garantiert UTM-S bei allen Störungen die Sicherheit und Konsistenz der Anwendungsdaten. Im Standardfall werden UTM-Anwendungen als UTM-S-Anwendungen (UTM-Secure) generiert.

UTM-SAT-Administration (BS2000-Systeme)
UTM SAT administration

Durch die UTM-SAT-Administration wird gesteuert, welche sicherheitsrelevanten UTM-Ereignisse, die im Betrieb der UTM-Anwendung auftreten, von SAT protokolliert werden sollen. Für die UTM-SAT-Administration wird eine besondere Berechtigung benötigt.

UTM-Seite
UTM page

Ist eine Speichereinheit, die entweder 2K, 4K oder 8K umfasst. In stand-alone UTM-Anwendungen kann die Größe einer UTM-Seite bei der Generierung der UTM-Anwendung auf 2K, 4K oder 8K gesetzt werden. In einer UTM-Cluster-Anwendung ist die Größe einer UTM-Seite immer 4K oder 8K. Pagepool und Wiederanlauf-Bereich der KDCFILE sowie UTM-Cluster-Dateien werden in Einheiten der Größe einer UTM-Seite unterteilt.

UTM Socket Protokoll (USP)
UTM socket protocol

Proprietäres Protokoll von openUTM oberhalb von TCP/IP zur Umsetzung der über die Socket-Schnittstelle empfangenen Bytestreams in Nachrichten.

UTM-System-Prozess
UTM system process

UTM-Prozess, der zusätzlich zu den per Startparameter angegebenen Prozessen gestartet wird und nur ausgewählte Aufträge bearbeitet. UTM-System-Prozesse dienen dazu, eine UTM-Anwendung auch bei sehr hoher Last reaktionsfähig zu halten.

UTM-Tool
UTM tool

Programm, das zusammen mit openUTM zur Verfügung gestellt und für bestimmte UTM-spezifische Aufgaben benötigt wird (z.B. zum Konfigurieren).

utmpfad (Unix-, Linux- und Windows-Systeme)
utmpath

Das Dateiverzeichnis unter dem die Komponenten von openUTM installiert sind, wird in diesem Handbuch als utmpfad bezeichnet.
Um einen korrekten Ablauf von openUTM zu garantieren, muss die Umgebungsvariable UTMPATH auf den Wert von utmpfad gesetzt werden. Auf Unix- und Linux-Systemen müssen Sie UTMPATH vor dem Starten einer UTM-Anwendung setzen. Auf Windows-Systemen wird UTMPATH passend zu der zuletzt installierten UTM-Version gesetzt.

Verarbeitungsschritt
processing step

Ein Verarbeitungsschritt beginnt mit dem Empfangen einer Dialog-Nachricht, die von einem Client oder einer anderen Server-Anwendung an die UTM-Anwendung gesendet wird. Der Verarbeitungsschritt endet entweder mit dem Senden einer Antwort und beendet damit auch den Dialog-Schritt oder er endet mit dem Senden einer Dialog-Nachricht an einen Dritten.

Verbindungs-Benutzerkennung
connection user ID

Benutzerkennung, unter der eine TS-Anwendung oder ein UPIC-Client direkt nach dem Verbindungsaufbau bei der UTM-Anwendung angemeldet wird. Abhängig von der Generierung des Clients (= LTERM-Partner) gilt:

- Die Verbindungs-Benutzerkennung ist gleich dem USER der LTERM-Anweisung (explizite Verbindungs-Benutzerkennung). Eine explizite Verbindungs-Benutzerkennung muss mit einer USER-Anweisung generiert sein und kann nicht als "echte" Benutzerkennung verwendet werden.
- Die Verbindungs-Benutzerkennung ist gleich dem LTERM-Partner (implizite Verbindungs-Benutzerkennung), wenn bei der LTERM-Anweisung kein USER angegeben wurde oder wenn ein LTERM-Pool generiert wurde.

In einer UTM-Cluster-Anwendung ist der Vorgang einer Verbindungs-Benutzerkennung (RESTART=YES bei LTERM oder USER) an die Verbindung gebunden und damit Knoten-lokal. Eine Verbindungs-Benutzerkennung, die mit RESTART=YES generiert ist, kann in jeder Knoten-Anwendung einen eigenen Vorgang haben.

Verbindungsbündel
connection bundle

siehe LTERM-Bündel.

Verschlüsselungsstufe
encryption level

Die Verschlüsselungsstufe legt fest, ob und inwieweit ein Client Nachrichten und Passwort verschlüsseln muss.

Verteilte Transaktion
distributed transaction

Transaktion, die sich über mehr als eine Anwendung erstreckt und in mehreren (Teil-)Transaktionen in verteilten Systemen ausgeführt wird.

Verteilte Transaktionsverarbeitung
Distributed Transaction Processing

Verteilte Verarbeitung mit verteilten Transaktionen.

Verteilte Verarbeitung
distributed processing

Bearbeitung von Dialog-Aufträgen durch mehrere Anwendungen oder Übermittlung von Hintergrundaufträgen an eine andere Anwendung. Für die verteilte Verarbeitung werden die höheren Kommunikationsprotokolle LU6.1 und OSI TP verwendet. Über openUTM-LU62 ist verteilte Verarbeitung auch mit LU6.2 Partnern möglich. Man unterscheidet verteilte Verarbeitung mit verteilten Transaktionen (Anwendungs-übergreifende Transaktionssicherung) und verteilte Verarbeitung ohne verteilte Transaktionen (nur lokale Transaktionssicherung). Die verteilte Verarbeitung wird auch Server-Server-Kommunikation genannt.

Vorgang (KDCS)
service

Ein Vorgang dient zur Bearbeitung eines Auftrags in einer UTM-Anwendung. Er setzt sich aus einer oder mehreren Transaktionen zusammen. Die erste Transaktion wird über den Vorgangs-TAC aufgerufen. Es gibt Dialog-Vorgänge und Asynchron-Vorgänge. openUTM stellt den Teilprogrammen eines Vorgangs gemeinsame Datenbereiche zur Verfügung. Anstelle des Begriffs Vorgang wird häufig auch der allgemeinere Begriff Service gebraucht.

Vorgangs-Kellerung (KDCS)
service stacking

Ein Terminal-Benutzer kann einen laufenden Dialog-Vorgang unterbrechen und einen neuen Dialog-Vorgang einschieben. Nach Beendigung des eingeschobenen Vorgangs wird der unterbrochene Vorgang fortgesetzt.

Vorgangs-Kettung (KDCS)
service chaining

Bei Vorgangs-Kettung wird nach Beendigung eines Dialog-Vorgangs ohne Angabe einer Dialog-Nachricht ein Folgevorgang gestartet.

Vorgangs-TAC (KDCS)
service TAC

Transaktionscode, mit dem ein Vorgang gestartet wird.

Vorgangs-Wiederanlauf (KDCS)
service restart

Wird ein Vorgang unterbrochen, z.B. infolge Abmeldens des Terminal-Benutzers oder Beendigung der UTM-Anwendung, führt openUTM einen Vorgangs-Wiederanlauf durch. Ein Asynchron-Vorgang wird neu gestartet oder beim zuletzt erreichten Sicherungspunkt fortgesetzt, ein Dialog-Vorgang wird beim zuletzt erreichten Sicherungspunkt fortgesetzt. Für den Terminal-Benutzer wird der Vorgangs-Wiederanlauf eines Dialog-Vorgangs als Bildschirm-Wiederanlauf sichtbar, sofern am letzten Sicherungspunkt eine Dialog-Nachricht an den Terminal-Benutzer gesendet wurde.

Warmstart
warm start

Start einer UTM-S-Anwendung nach einer vorhergehenden abnormalen Beendigung. Dabei wird die Anwendungsinformation auf den zuletzt erreichten konsistenten Zustand gesetzt. Unterbrochene Dialog-Vorgänge werden dabei auf den zuletzt erreichten Sicherungspunkt zurückgesetzt, so dass die Verarbeitung an dieser Stelle wieder konsistent aufgenommen werden kann (Vorgangs-Wiederanlauf). Unterbrochene Asynchron-Vorgänge werden zurückgesetzt und neu gestartet oder beim zuletzt erreichten Sicherungspunkt fortgesetzt.
Bei UTM-F-Anwendungen werden beim Start nach einer vorhergehenden abnormalen Beendigung lediglich die dynamisch geänderten Konfigurationsdaten auf den zuletzt erreichten konsistenten Zustand gesetzt.
In UTM-Cluster-Anwendungen werden die globalen Sperren auf GSSB und ULS, die bei der abnormalen Beendigung von dieser Knoten-Anwendung gehalten wurden, aufgehoben. Außerdem werden Benutzer, die zum Zeitpunkt der abnormalen Beendigung an dieser Knoten-Anwendung angemeldet waren, abgemeldet.

Web Service
web service

Anwendung, die auf einem Web-Server läuft und über eine standardisierte und programmatische Schnittstelle (öffentlich) verfügbar ist. Die Web Services-Technologie ermöglicht es, UTM-Teilprogramme für moderne Web-Client-Anwendungen verfügbar zu machen, unabhängig davon, in welcher Programmiersprache sie entwickelt wurden.

WebAdmin
WebAdmin

Web-basiertes Tool zur Administration von openUTM-Anwendungen über Web-Browser. WebAdmin enthält neben dem kompletten Funktionsumfang der Programmschnittstelle zur Administration noch zusätzliche Funktionen.

Wiederanlauf
restart

siehe Bildschirm-Wiederanlauf,
siehe Vorgangs-Wiederanlauf.

WinAdmin
WinAdmin

Java-basiertes Tool zur Administration von openUTM-Anwendungen über eine grafische Oberfläche. WinAdmin enthält neben dem kompletten Funktionsumfang der Programmschnittstelle zur Administration noch zusätzliche Funktionen.

Workload Capture & Replay
workload capture & replay

Programmfamilie zur Simulation von Lastsituationen, bestehend aus den Haupt-Komponenten UPIC Capture, UPIC Analyzer und Upic Replay und auf Unix-, Linux- und Windows-Systemen dem Dienstprogramm kdcsort. Mit Workload Capture & Replay lassen sich UPIC-Sessions mit UTM-Anwendungen aufzeichnen, analysieren und mit veränderten Lastparametern wieder abspielen.

Workprozess (Unix-, Linux- und Windows-Systeme)
work process

Prozess, in dem die Services der UTM-Anwendung ablaufen.

WS4UTM

WS4UTM (**W**eb**S**ervices for open**UTM**) ermöglicht es Ihnen, auf komfortable Weise einen Service einer UTM-Anwendung als Web Service zur Verfügung zu stellen.

XATMI

XATMI (X/Open Application Transaction Manager Interface) ist eine von X/Open standardisierte Programmschnittstelle für die Programm-Programm-Kommunikation in offenen Netzen.
Das in openUTM implementierte XATMI genügt der XATMI CAE Specification von X/Open. Die Schnittstelle steht in COBOL und C zur Verfügung. XATMI in openUTM kann über die Protokolle OSI TP, LU6.1 und UPIC kommunizieren.

XHCS (BS2000-Systeme)

XHCS (Extended Host Code Support) ist ein BS2000-Softwareprodukt für die Unterstützung internationaler Zeichensätze.

XML

XML (eXtensible Markup Language) ist eine vom W3C (WWW-Konsortium) genormte Metasprache, in der Austauschformate für Daten und zugehörige Informationen definiert werden können.

Zeitgesteuerter Auftrag
time-driven job

Auftrag, der von openUTM bis zu einem definierten Zeitpunkt in einer Message Queue zwischengespeichert und dann an den Empfänger weitergeleitet wird. Empfänger kann sein: ein Asynchron-Vorgang der selben Anwendung, eine TAC-Queue, eine Partner-Anwendung, ein Terminal oder ein Drucker. Zeitgesteuerte Aufträge können nur von KDCS-Teilprogrammen erteilt werden.

Zugangskontrolle
system access control

Prüfung durch openUTM, ob eine bestimmte Benutzerkennung berechtigt ist, mit der UTM-Anwendung zu arbeiten. Die Berechtigungsprüfung entfällt, wenn die UTM-Anwendung ohne Benutzerkennungen generiert wurde.

Zugriffskontrolle
data access control

Prüfung durch openUTM, ob der Kommunikationspartner berechtigt ist, auf ein bestimmtes Objekt der Anwendung zuzugreifen. Die Zugriffsrechte werden als Bestandteil der Konfiguration festgelegt.

Zugriffspunkt
access point

siehe Dienstzugriffspunkt.

16 Abkürzungen

ACSE	Association Control Service Element
AEQ	Application Entity Qualifier
AES	Advanced Encryption Standard
AET	Application Entity Title
APT	Application Process Title
ASCII	American Standard Code for Information Interchange
ASE	Application Service Element
Axis	Apache eXtensible Interaction System
BCAM	Basic Communication Access Method
BER	Basic Encoding Rules
BLS	Binder-Lader-Starter (BS2000-Systeme)
CCP	Communication Control Program
CCR	Commitment, Concurrency and Recovery
CCS	Codierter Zeichensatz (Coded Character Set)
CCSN	Name des codierten Zeichensatzes (Coded Character Set Name)
CICS	Customer Information Control System (IBM)
CID	Control Identification
CMX	Communication Manager in Unix-, Linux- und Windows-Systemen
COM	Component Object Model
CPI-C	Common Programming Interface for Communication
CRM	Communication Resource Manager
CRTE	Common Runtime Environment (BS2000-Systeme)
DB	Database
DBH	Database Handler
DC	Data Communication
DCAM	Data Communication Access Method
DES	Data Encryption Standard

DLS	Distributed Lock Manager (BS2000-Systeme)
DMS	Data Management System
DNS	Domain Name Service
DSS	Datensichtstation (=Terminal)
DTD	Document Type Definition
DTP	Distributed Transaction Processing
DVS	Datenverwaltungssystem
EBCDIC	Extended Binary-Coded Decimal Interchange Code
EJB	Enterprise JavaBeans™
FGG	File Generation Group
FHS	Format Handling System
FT	File Transfer
GCM	Galois/Counter Mode
GSSB	Globaler Sekundärer Speicherbereich
HIPLEX®	Highly Integrated System Complex (BS2000-Systeme)
HLL	High-Level Language
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IFG	Interaktiver Format-Generator
ILCS	Inter Language Communication Services (BS2000-Systeme)
IMS	Information Management System (IBM)
IPC	Inter-Process-Communication
IRV	Internationale Referenzversion
ISO	International Organization for Standardization
Java EE	Java Platform, Enterprise Edition
JCA	Java EE Connector Architecture
JDK	Java Development Kit
KAA	KDCS Application Area
KB	Kommunikationsbereich

KBPROG	KB-Programmbereich
KDCADMI	KDC Administration Interface
KDCS	Kompatible Datenkommunikationsschnittstelle
KTA	KDCS Task Area
LAN	Local Area Network
LCF	Local Configuration File
LLM	Link and Load Module (BS2000-Systeme)
LSSB	Lokaler Sekundärer Speicherbereich
LU	Logical Unit
MQ	Message Queuing
MSCF	Multiple System Control Facility (BS2000-Systeme)
NB	Nachrichtenbereich
NEA	Netzwerkarchitektur bei BS2000-Systemen
NFS	Network File System/Service
NLS	Unterstützung der Landessprache (Native Language Support)
OLTP	Online Transaction Processing
OML	Object Modul Library
OSI	Open System Interconnection
OSI TP	Open System Interconnection Transaction Processing
OSS	OSI Session Service
PCMX	Portable Communication Manager
PID	Prozess-Identifikation
PIN	Persönliche Identifikationsnummer
PLU	Primary Logical Unit
PTC	Prepare to commit
RAV	Rechenzentrums-Abrechnungs-Verfahren
RDF	Resource Definition File
RM	Resource Manager
RSA	Encryption-Algorithmus nach Rivest, Shamir, Adleman

RSO	Remote SPOOL Output (BS2000-Systeme)
RTS	Runtime System (Laufzeitsystem)
SAT	Security Audit Trail (BS2000-Systeme)
SECOS	Security Control System
SEM	SE Manager
SGML	Standard Generalized Markup Language
SLU	Secondary Logical Unit
SM2	Software Monitor 2
SNA	Systems Network Architecture
SOA	Service-oriented Architecture
SOAP	Simple Object Access Protocol
SPAB	Standard Primärer Arbeitsbereich
SQL	Structured Query Language
SSB	Sekundärer Speicherbereich
SSL	Secure Socket Layer
SSO	Single-Sign-On
TAC	Transaktionscode
TCEP	Transport Connection End Point
TCP/IP	Transport Control Protocol / Internet Protocol
TIAM	Terminal Interactive Access Method
TLS	Terminal-spezifischer Langzeitspeicher
TLS	Transport Layer Security
TM	Transaction Manager
TNS	Transport Name Service
TP	Transaction Processing (Transaktions-Betrieb)
TPR	Task privileged (privilegierter Funktionszustand des BS2000-Systems)
TPSU	Transaction Protocol Service User
TSAP	Transport Service Access Point
TSN	Task Sequence Number
TU	Task user (nicht privilegierter Funktionszustand des BS2000-Systems)

TX	Transaction Demarcation (X/Open)
UDDI	Universal Description, Discovery and Integration
UDS	Universelles Datenbanksystem
UDT	Unstructured Data Transfer
ULS	User-spezifischer Langzeitspeicher
UPIC	Universal Programming Interface for Communication
USP	UTM-Socket-Protokoll
UTM	Universeller Transaktionsmonitor
UTM-D	UTM-Funktionen für verteilte Verarbeitung („Distributed“)
UTM-F	Schnelle UTM-Variante („Fast“)
UTM-S	UTM-Sicherheitsvariante
UTM-XML	XML-Schnittstelle von openUTM
VGID	Vorgangs-Identifikation
VTSU	Virtual Terminal Support
VTV	Verteilte Transaktionsverarbeitung
VV	Verteilte Verarbeitung
WAN	Wide Area Network
WS4UTM	WebServices for openUTM
WSDD	Web Service Deployment Descriptor
WSDL	Web Services Description Language
XA	X/Open Access Interface (Schnittstelle von X/Open zum Zugriff auf Resource Manager)
XAP	X/OPEN ACSE/Presentation programming interface
XAP-TP	X/OPEN ACSE/Presentation programming interface Transaction Processing extension
XATMI	X/Open Application Transaction Manager Interface
XCS	Cross Coupled System
XHCS	eXtended Host Code Support
XML	eXtensible Markup Language

17 Literatur

Die Handbücher finden Sie im Internet unter https://bs2manuals.ts.fujitsu.com.

Dokumentation zu openUTM

openUTM
Konzepte und Funktionen
Benutzerhandbuch

openUTM
Anwendungen programmieren mit KDCS für COBOL, C und C++
Basishandbuch

openUTM
Anwendungen generieren
Benutzerhandbuch

openUTM
Einsatz von UTM-Anwendungen auf BS2000-Systemen
Benutzerhandbuch

openUTM
Einsatz von UTM-Anwendungen auf Unix-, Linux- und Windows-Systemen
Benutzerhandbuch

openUTM
Anwendungen administrieren
Benutzerhandbuch

openUTM
Meldungen, Test und Diagnose auf BS2000-Systemen
Benutzerhandbuch

openUTM
Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen
Benutzerhandbuch

openUTM
Anwendungen erstellen mit X/Open-Schnittstellen
Benutzerhandbuch

openUTM
XML für openUTM

openUTM-Client (Unix-Systeme) **für Trägersystem OpenCPIC**
Client-Server-Kommunikation mit openUTM
Benutzerhandbuch

openUTM-Client für Trägersystem UPIC
Client-Server-Kommunikation mit openUTM
Benutzerhandbuch

openUTM WinAdmin
Grafischer Administrationsarbeitsplatz für openUTM
Beschreibung und Online-Hilfe

openUTM WebAdmin
Web-Oberfläche zur Administration von openUTM
Beschreibung und Online-Hilfe

openUTM, openUTM-LU62
Verteilte Transaktionsverarbeitung
zwischen openUTM und CICS-, IMS- und LU6.2-Anwendungen
Benutzerhandbuch

openUTM (BS2000)
Anwendungen programmieren mit KDCS für Assembler
Ergänzung zum Basishandbuch

openUTM (BS2000)
Anwendungen programmieren mit KDCS für Fortran
Ergänzung zum Basishandbuch

openUTM (BS2000)
Anwendungen programmieren mit KDCS für Pascal-XT
Ergänzung zum Basishandbuch

openUTM (BS2000)
Anwendungen programmieren mit KDCS für PL/I
Ergänzung zum Basishandbuch

WS4UTM (Unix- und Windows-Systeme**)**
Web-Services für openUTM

Dokumentation zum openSEAS-Produktumfeld

BeanConnect
Benutzerhandbuch

openUTM-JConnect
Verbindung von Java-Clients zu openUTM
Benutzerdokumentation und Java-Docs

WebTransactions
Konzepte und Funktionen

WebTransactions
Template-Sprache

WebTransactions
Anschluss an openUTM-Anwendungen über UPIC

WebTransactions
Anschluss an MVS-Anwendungen

WebTransactions
Anschluss an OSD-Anwendungen

Dokumentation zum BS2000-Umfeld

AID Advanced Interactive Debugger
Basishandbuch
Benutzerhandbuch

AID Advanced Interactive Debugger
Testen von COBOL-Programmen
Benutzerhandbuch

AID Advanced Interactive Debugger
Testen von C/C++-Programmen
Benutzerhandbuch

BCAM
BCAM Band 1/2
Benutzerhandbuch

BINDER
Benutzerhandbuch

BS2000 OSD/BC
Kommandos Band 1-7
Benutzerhandbuch

BS2000 OSD/BC
Makroaufrufe an den Ablaufteil
Benutzerhandbuch

BS2IDE
Eclipse-based Integrated Development Environment for BS2000
User Guide and Installation Guide
Webseite: *https://bs2000.ts.fujitsu.com/bs2ide/*

BLSSERV
Bindelader-Starter in BS2000/OSD
Benutzerhandbuch

DCAM
COBOL-Aufrufe
Benutzerhandbuch

DCAM
Makroaufrufe
Benutzerhandbuch

DCAM
Programmschnittstellen
Beschreibung

FHS
Formatierungssystem für openUTM, TIAM, DCAM
Benutzerhandbuch

IFG für FHS
Benutzerhandbuch

HIPLEX AF
Hochverfügbarkeit von Anwendungen in BS2000/OSD
Produkthandbuch

HIPLEX MSCF
BS2000-Rechner im Verbund
Benutzerhandbuch

IMON
Installationsmonitor
Benutzerhandbuch

LMS
SDF-Format
Benutzerhandbuch

MT9750 (MS Windows)
9750-Emulation unter Windows
Produkthandbuch

OMNIS/OMNIS-MENU
Funktionen und Kommandos
Benutzerhandbuch

OMNIS/OMNIS-MENU
Administration und Programmierung
Benutzerhandbuch

OSS (BS2000)
OSI Session Service
User Guide

openSM2
Software Monitor
Benutzerhandbuch

RSO
Remote SPOOL Output
Benutzerhandbuch

SECOS
Security Control System
Benutzerhandbuch

SECOS
Security Control System
Tabellenheft

SESAM/SQL
Datenbankbetrieb
Benutzerhandbuch

TIAM
Benutzerhandbuch

UDS/SQL
Datenbankbetrieb
Benutzerhandbuch

Unicode im BS2000/OSD
Übersichtshandbuch

VTSU
Virtual Terminal Support
Benutzerhandbuch

XHCS
8-bit-Code- und Unicode-Unterstützung im BS2000/OSD
Benutzerhandbuch

Dokumentation zum Umfeld von Unix-, Linux- und Windows-Systemen

CMX V6.0 (Unix-Systeme)
Betrieb und Administration
Benutzerhandbuch

CMX V6.0
CMX-Anwendungen programmieren
Programmierhandbuch

OSS (UNIX)
OSI Session Service
User Guide

PRIMECLUSTER™
Konzept (Solaris, Linux)
Benutzerhandbuch

openSM2
Die Dokumentation zu openSM2 wird in Form von ausführlichen Online-Hilfen bereitgestellt, die mit dem Produkt ausgeliefert werden.

Sonstige Literatur

CPI-C
X/Open CAE Specification
Distributed Transaction Processing:
The CPI-C Specification, Version 2
ISBN 1 85912 135 7

Reference Model
X/Open Guide
Distributed Transaction Processing:
Reference Model, Version 2
ISBN 1 85912 019 9

REST
Architectural Styles and the Design of Network-based Software Architectures
Dissertation Roy Fielding

TX
X/Open CAE Specification
Distributed Transaction Processing:
The TX (Transaction Demarcation) Specification
ISBN 1 85912 094 6

XATMI
X/Open CAE Secification
Distributed Transaction Processing
The XATMI Specification
ISBN 1 85912 130 6

XML
Spezifikation des W3C (www – Konsortium)
Webseite: *http://www.w3.org/XML*