openUTM V5.3, openUTM-LU62 V5.1

Verteilte Transaktionsverarbeitung zwischen openUTM- und CICS-, IMS- und LU6.2-Anwendungen

Ausgabe Juli 2008

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 manuals@ts.fujitsu.com senden.

Zertifizierte Dokumentation nach DIN EN ISO 9001:2000

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 9001:2000 erfüllt.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright und Handelsmarken

Inhalt

1 Einleitung

Moderne unternehmensweite IT-Umgebungen unterliegen zahlreichen Herausforderungen von zunehmender Brisanz. Dies wird verursacht durch

- heterogene Systemlandschaften
- unterschiedliche HW-Plattformen
- unterschiedliche Netze und Netzzugriffe (TCP/IP, SNA, HTTP)
- Verflechtung der Anwendungen mit den Unternehmen

Dadurch entwickeln sich Problemfelder, sei es bei Fusionen, durch Kooperationen oder auch nur durch Rationalisierungsmaßnahmen. Die Unternehmen fordern flexible und skalierbare Anwendungen, gleichzeitig soll die Transaktionssicherheit für Prozesse und Daten gewährleistet bleiben, obwohl die Geschäftsprozesse immer komplexer werden. Die wachsende Globalisierung geht selbstverständlich davon aus, dass Anwendungen im 7x24-Stunden-Betrieb laufen und hochverfügbar sind, um beispielsweise Internetzugriffe auf bestehende Anwendungen über Zeitzonen hinweg zu ermöglichen.

Der klassische Anwendungsserver openUTM bietet eine Ablaufumgebung, die allen diesen Anforderungen moderner unternehmenskritischer Anwendungen gewachsen ist, denn openUTM verbindet alle Standards und Vorteile von Transaktionsmonitor-Middleware und Message Queuing Systemen:

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

openUTM ist Teil des umfassenden Angebots von **openSEAS**. Im Rahmen des Produktangebots **openSEAS** (Open Suite for Enterprise Application Servers) nutzen innovative Produkte die ausgereifte Technologie von openUTM:

- BeanConnect ist eine Familie von Adaptern gemäß der Java Connector Architecture (JCA) von Sun und bietet den standardisierten Anschluss von UTM-Anwendungen an J2EE Application Server, insbesondere an Oracle AS 10g. Oracle AS 10g ist der Application Server, der als Bestandteil von openSEAS als Entwicklungs- und Ablaufplattform aller Geschäftsanwendungen, Portale und Web-Seiten im Grid Computing-Maßstab zur Verfügung steht.
- Web*Transactions* ermöglicht u.a. zusammen mit openUTM moderne eBusiness-Applikationen zu erstellen. Bestehende UTM-Anwendungen können unverändert mit Web*Transactions* in das World Wide Web übernommen und in Portale integriert werden. Web*Transactions* enthält mit dem Business Object Builder Biz*Transactions* das openSEAS-Werkzeug zum Erzeugen wiederverwendbarer Business-Objekte. Die Business-Objekte werden aus bestehenden Anwendungen erzeugt (z.B. BS2000/OSD- und OS/390-Anwendungen, Transaktionsanwendungen mit openUTM und SAP R/3) und können je nach Bedarf als JavaBeans, .Net-Komponenten oder Web Services verfügbar gemacht werden.

1.1 Konzept und Zielgruppen dieses Handbuchs

Das vorliegende Handbuch „Verteilte Transaktionsverarbeitung zwischen openUTM und CICS-, IMS oder LU6.2-Anwendungen“ beschreibt, wie man bei verteilter Transaktionsverarbeitung UTM-Anwendungen in BS2000/OSD, UNIX- und Windows-Systemen mit CICS-, IMS- und anderen LU6.2-Anwendungen in IBM-Systemen koppeln kann. Es richtet sich an Systemverwalter, Netzverwalter und Anwendungsprogrammierer.

Das Kapitel 2 enthält allgemeine Informationen zu Kopplungen zwischen openUTM und IBM-Systemen.

Die Kapitel 3 bis 5 beschreiben die LU6.2-Kopplung zwischen openUTM und IBM-Systemen. Kapitel 3 beschreibt das Gateway openUTM-LU62 einschließlich der Administration von openUTM-LU62. Kapitel 4 und 5 gehen näher auf die Kopplung mit CICS und IMS ein.

Die Kapitel 6 bis 8 beschreiben die LU6.1-Kopplung zwischen openUTM und CICS bzw. IMS. Dabei beschreibt Kapitel 6 die Kopplung zu CICS, Kapitel 7 die LU6.1-Kopplung zu IMS. In Kapitel 8 werden spezielle Erweiterungen an der openUTM-Programmschnittstelle für die Kopplung openUTM - IMS beschrieben.

In Kapitel 9 erhalten Sie Hinweise für eine Fehlerdiagnose, im Kapitel 10 werden alle Meldungen von openUTM-LU62 aufgelistet.

Die ausführlichen Verzeichnisse am Ende des Handbuches - Fachwörter, Abkürzungen, Literatur, Stichwörter - sollen Ihnen den Umgang mit diesem Handbuch erleichtern.

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, einer Online-Hilfe für den grafischen Administrationsarbeitsplatz openUTM WinAdmin sowie einer Freigabemitteilung für jede Plattform, auf der openUTM freigegeben wird.

Es gibt Handbücher, die für alle Plattformen gültig sind, sowie Handbücher, die jeweils für BS2000/OSD bzw. für UNIX-Systeme und Windows-Systeme gelten.

Sämtliche Handbücher sind als PDF-Datei im Internet verfügbar unter der Adresse

`http://manuals.ts.fujitsu.com`

Die meisten Handbücher können außerdem auch als gebundene Papierexemplare bestellt werden. Zusätzlich werden die pdf-Dateien auf der openUTM-Produkt-DVD ausgeliefert.

Die folgenden Abschnitte geben einen Aufgaben-bezogenen Überblick über die Dokumentation zu openUTM V5.3. Eine vollständige Liste der Dokumentation zu openUTM finden Sie im Literaturverzeichnis auf Seite 377.

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/OSD-, UNIX- 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 beschrieben ist. Diese Schnittstelle umfasst sowohl die Basisfunktionen des universellen Transaktionsmonitors als auch die Aufrufe für verteilte Verarbeitung. Es wird auch die Zusammenarbeit mit Datenbanken beschrieben.
- Wollen Sie die X/Open-Schnittstellen nutzen, benötigen Sie das Handbuch **Anwendungen erstellen mit X/Open-Schnittstellen**. Es enthält die UTM-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-Specification für die jeweilige X/Open-Schnittstelle.
- Wenn Sie Daten auf Basis von XML austauschen wollen, benötigen Sie das Dokument **Data Marshalling mit XML für openUTM**. Darin werden die C- und COBOL-Aufrufe beschrieben, die zum Bearbeiten von XML-Dokumenten benötigt werden. Diese Beschreibung ist nur als PDF-Dokument erhältlich (online und auf CD).
- Für BS2000/OSD gibt es Ergänzungsbände für die Programmiersprachen Assembler, Fortran, Pascal-XT und PL/1. Diese sind nur als PDF-Datei verfügbar (online und auf der WinAdmin-CD).

Konfigurieren

Zur Definition von Konfigurationen steht Ihnen das Handbuch **Anwendungen generieren** zur Verfügung. Darin ist beschrieben, wie Sie für eine UTM-Anwendung mit Hilfe des UTM-Tools KDCDEF die Konfiguration definieren und die KDCFILE 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.

Binden, Starten und Einsetzen

Um UTM-Anwendungen einsetzen zu können, benötigen Sie für das betreffende Betriebssystem (BS2000/OSD bzw. UNIX-/Windows-Systeme) das Handbuch **Einsatz von openUTM-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.

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 und über die Möglichkeiten der zentralen Administration mehrerer Anwendungen. 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** einsetzen, dann steht Ihnen folgende Dokumentation zur Verfügung:
  - Die **WinAdmin-Beschreibung** bietet einen umfassenden Überblick über den Funktionsumfang von WinAdmin und das Handling von WinAdmin. Dieses Dokument wird mit der Software ausgeliefert und ist zusätzlich auch online als PDF-Datei verfügbar.
  - Das **Online-Hilfesystem** beschreibt kontextsensitiv alle Dialogfelder und die zugehörigen Parameter, die die grafische Oberfläche bietet. Außerdem wird dargestellt, wie man WinAdmin konfiguriert, um UTM-Anwendungen administrieren und generieren zu können.

Testen und Fehler diagnostizieren

Für die o.g. Aufgaben benötigen Sie außerdem die Handbücher **Meldungen, Test und Diagnose** (jeweils ein Handbuch für UNIX-/Windows-Systeme und für BS2000/OSD). Sie beschreiben das Testen einer UTM-Anwendung, den Inhalt und die Auswertung eines UTM-Dumps, das Verhalten im Fehlerfall, 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. Neben der Beschreibung der Schnittstellen CPI-C und XATMI erhalten Sie Informationen, wie Sie die C++-Klassen oder ActiveX für die schnelle und einfache Programmerstellung nutzen können.
- 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 die mit **BeanConnect** ausgelieferten **JUpic-Java-Klassen** wird die Dokumentation mit der Software ausgeliefert. Diese Dokumentation besteht aus Word- und PDF-Dateien, die die Einführung und die Installation beschreiben, sowie aus einer Java-Dokumentation mit der Beschreibung der Java-Klassen.
- Wenn Sie UTM-Services auf einfache Weise ins Web stellen möchten, benötigen Sie das Handbuch **Web-Services für openUTM**. Das Handbuch beschreibt, wie Sie mit dem Software-Produkt WS4UTM (WebServices for openUTM) Services von UTM-Anwendungen als Web Services verfügbar machen. Diese Beschreibung ist nur als PDF-Dokument erhältlich (online und auf der openUTM-Produkt-DVD). Die Arbeit mit der grafischen Bedienoberfläche ist in der zugehörigen **Online-Hilfe** beschrieben.

Kopplung mit der IBM-Welt

Das vorliegende Handbuch **Verteilte Transaktionsverarbeitung zwischen openUTM und CICS-, IMS- und LU6.2-Anwendungen** beschreibt, wie Sie aus Ihrer UTM-Anwendung mit Transaktionssystemen von IBM kommunizieren. 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.

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 J2EE Application Servern und UTM-Anwendungen

Die Adapter-Familie BeanConnect gehört zur Produkt-Suite openSEAS (open Suite for Enterprise Application Servers). Die BeanConnect-Adapter realisieren die Verknüpfung zwischen klassischen Transaktionsmonitoren und modernen Application Servern und ermöglichen damit die effiziente Integration von Legacy-Anwendungen in moderne Java-Anwendungen.

- Das Handbuch **BeanConnect for openUTM** beschreibt das Produkt BeanConnect for openUTM. BeanConnect for openUTM bietet einen JCA 1.5-konformen Adapter, der UTM-Anwendungen mit Anwendungen auf Basis von J2EE, z.B. dem Oracle Application Server Oracle AS 10g, verbindet.

  Die Handbücher zu Oracle AS 10g sind bei Oracle beziehbar.

- Das Handbuch **BeanConnect for CICS** beschreibt das Produkt BeanConnect for CICS. BeanConnect for CICS bietet einen JCA 1.5-konformen Adapter, der CICS-Anwendungen mit Anwendungen auf Basis von J2EE, z.B. dem Oracle Application Server, verbindet.

Web-Anbindung und Anwendungsintegration

Zum Anschließen neuer und bestehender UTM-Anwendungen an das Web mit dem Produkt Web*Transactions* benötigen Sie die folgenden Handbücher:

- Das einführende Handbuch **Web*Transactions* - Konzepte und Funktionen** gibt Ihnen einen Überblick über die Leistungsfähigkeit und Einsatzmöglichkeiten des Produkts Web*Transactions* und informiert Sie über dessen Eigenschaften und Funktionsweisen. Das Handbuch erläutert das Objekt-Konzept und den dynamischen Ablauf einer Web*Transactions*-Sitzung.

- Im Handbuch **Web*Transactions* - Template-Sprache** sind alle Sprachmittel der Template-Sprache WTML erläutert. Es enthält zahlreiche Beispiele, die die Sprachmittel veranschaulichen und Ihnen den Einsatz erleichtern.

- Die Handbücher **Web*Transactions* - Anschluss an openUTM-Anwendungen über UPIC** und **Web*Transactions* - Anschluss an OSD-Anwendungen** beschreiben die notwendigen Schritte, um UTM-Dialog-Anwendungen über die UPIC- bzw. die Terminal-Schnittstelle an das Web anzuschließen. An einem konkreten Beispiel werden diese Schritte nochmals verdeutlicht.

- Wenn Sie UTM-Anwendungen und -Services in Microsoft-Anwendungen integrieren möchten, dann steht Ihnen dafür die mit Web*Transactions* ausgelieferte Komponente Biz*Transactions* mit dem Handbuch **Biz*Transactions* - Anwendungsintegration mit Business Objekten** zur Verfügung. Es beschreibt neben der Installation und Konfiguration auch die Konzepte und Komponenten von Biz*Transactions*. Außerdem wird die Administration und das Entwickeln von integrierten Client-Anwendungen erläutert.

  Für Biz*Transactions* wird mit der Software auch ein Online-Hilfesystem ausgeliefert.

Die genannten Handbücher werden jeweils noch 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.

- BS2000/OSD:

  Sie finden die Informationen auf Ihrem BS2000-Rechner in der Freigabemitteilung (Dateiname SYSFGM.UTM.053.D*)* und eventuell zusätzlich in einer Readme-Datei (Dateiname SYSRME.UTM.053.D*)*. Die Benutzerkennung, unter der sich die Datei befindet, erfragen Sie bitte bei Ihrer zuständigen Systembetreuung. Die Datei können Sie mit dem Kommando /SHOW-FILE oder mit einem Editor ansehen oder auf einem Standarddrucker mit folgendem Kommando ausdrucken:

  `/PRINT-DOCUMENT` *dateiname*`,LINE-SPACING=*BY-EBCDIC-CONTROL`

- UNIX-Systeme:

  Die Readme-Datei und ggf. weitere Dateien wie z.B. eine Handbuchergänzungsdatei finden Sie im *utmpfad* unter `/docs/`*sprache*.

- Windows-Systeme:

  Die Readme-Datei und ggf. weitere Dateien wie z.B. eine Handbuchergänzungsdatei finden Sie im *utmpfad* unter `\Docs\`*sprache*.

1.3 Änderungen gegenüber dem Vorgängerhandbuch

Die wichtigsten Änderung gegenüber dem Vorgängerhandbuch sind:

- Erweiterung der Generierungsparameter von openUTM-LU62: Die bisher im TNSX enthaltenen Parameter wurden ergänzt. Somit wird TNSX jetzt nicht mehr benötigt.
- Neue Startoption, die als Mittelweg zwischen Kaltstart und Warmstart fungiert: `u62_start -k`
- Neue Option bei der Statusanzeige, die den Zustand der Enterprise-Extender-Verbindungen anzeigt: `u62_sta -c`

1.4 Metasyntax

Die in diesem Handbuch verwendete Metasyntax ist dieselbe wie in den Vorgängerhandbüchern:

GROSSBUCHSTABEN
: In Großbuchstaben sind die Namen von Aufrufen, Anweisungen, Kommandos und Operanden dargestellt.

kleinbuchstaben
: In Kleinbuchstaben sind die Platzhalter für Operandenwerte dargestellt.

[ ]
: In eckigen Klammern stehen wahlfreie Angaben, die entfallen können.

{ }
: In geschweiften Klammern stehen alternative Angaben, von denen Sie eine auswählen müssen.

Unterstreichung
: Unterstreichung kennzeichnet den Standardwert.

`Schreibmaschinenschrift`
: In Schreibmaschinenschrift im Fließtext werden Kommandos, Dateinamen, Meldungen und Beispiele ausgezeichnet, die in genau dieser Form eingegeben werden müssen bzw. die genau diesen Namen oder diese Form besitzen.

Kursiv
: Kursivschrift im Fließtext kennzeichnet Variablen.

2 Kopplung mit IBM-Systemen

Bei IBM-Systemen ist verteilte Transaktionsverarbeitung üblicherweise mit dem SNA-Protokoll LU6.2 realisiert.

Folgende IBM-Transaktionsmonitore und Kommunikationsprodukte unterstützen dieses Protokoll:

– CICS (bzw. CICS Transaction Server)

– IMS (ab Version 6)

– TXSeries auf System p5 (früher RS/6000)

– CPI-C-Anwendungsprogramme auf IBM System i5 (früher AS/400).

Ältere IBM-Transaktionsmonitore wie z. B. IMS bis zur Version 5 können nur mittels LU6.1 transaktionsgesichert erreicht werden.

openUTM kann für die verteilte Transaktionsverarbeitung die Protokolle LU6.1, OSI-TP und LU6.2 verwenden.

Bei neu einzurichtenden Kopplungen in die IBM-Welt sollte nur noch LU6.2 verwendet werden, sofern es die Partner-Anwendung unterstützt.

Für eine Kopplung zwischen openUTM und einem LU6.2-Partner wird das Produkt openUTM-LU62 und eine zum Betriebssystem passende SNA-Basis-Software benötigt. openUTM-LU62 ist auf den Betriebssystemen Solaris, Windows, Linux und AIX verfügbar. Folgende Software-Konfigurationen stehen zur Verfügung:

– openUTM-LU62 auf Solaris mit den Produkten TRANSIT-SERVER und TRANSIT-CPIC

– openUTM-LU62 auf Solaris mit SNAP-IX

– openUTM-LU62 auf Windows mit IBM Communications Server for Windows

– openUTM-LU62 auf Linux mit IBM Communications Server for Linux

– openUTM-LU62 auf AIX mit IBM Communications Server for AIX

Im vorliegenden Handbuch werden Namen der IBM-Produkte wie z.B. IBM Communications Server for Linux abgekürzt mit IBM Communications Server.

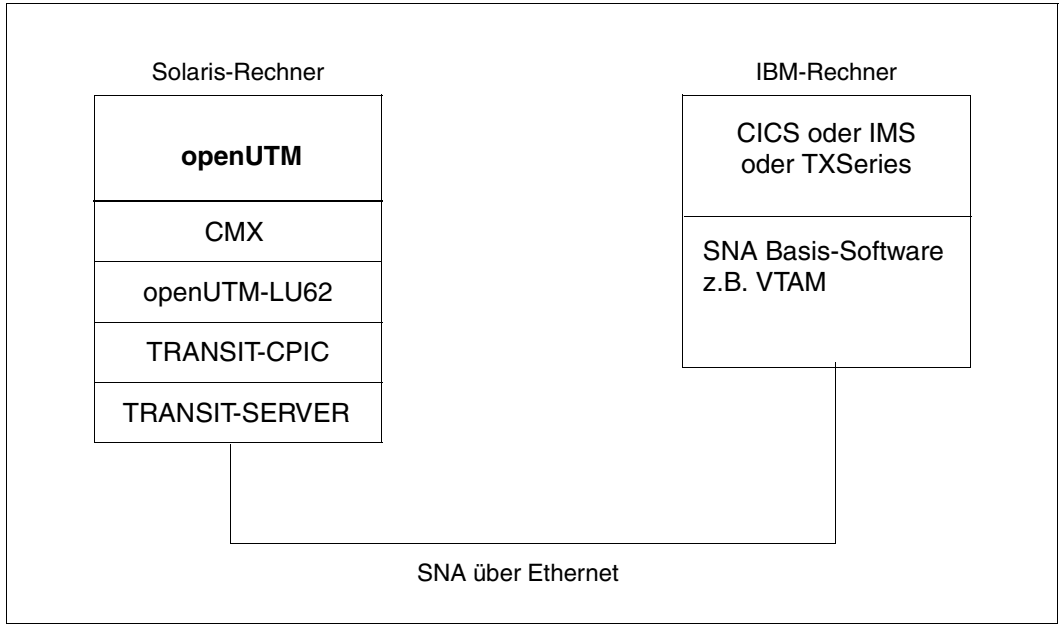
Im Folgenden werden die typischen Konfigurationen bei einer Kopplung zwischen openUTM und einem IBM-Produkt aufgelistet. Dabei werden Produkte unterschiedlicher Hersteller erwähnt.

- IBM Communications Server ist ein Produkt der Firma IBM und erbringt die Funktionen der SNA-Basis-Software.
- SNAP-IX ist ein Produkt der Firma Data Connection Limited und erbringt die Funktionen der SNA-Basis-Software.
- TRANSIT-SERVER, TRANSIT-CLIENT und TRANSIT-CPIC sind Produkte von Fujitsu Siemens Computers und erbringen die Funktionen der SNA-Basis-Software.
- CMX, PCMX und BCAM sind Produkte von Fujitsu Siemens Computers.
- CICS, IMS, TxSeries und VTAM sind Produkte der Firma IBM.

Generell gilt, dass die Namen der IBM-Produkte in manchen Produktversionen leicht abweichen können.

2.1 Direkte LU6.2-Kopplung über TRANSIT auf Solaris

Die folgende Kopplungsmöglichkeit kann eingesetzt werden, wenn die UTM-Anwendung auf einem Solaris-Rechner läuft, der direkt an das SNA-Netz angebunden ist.



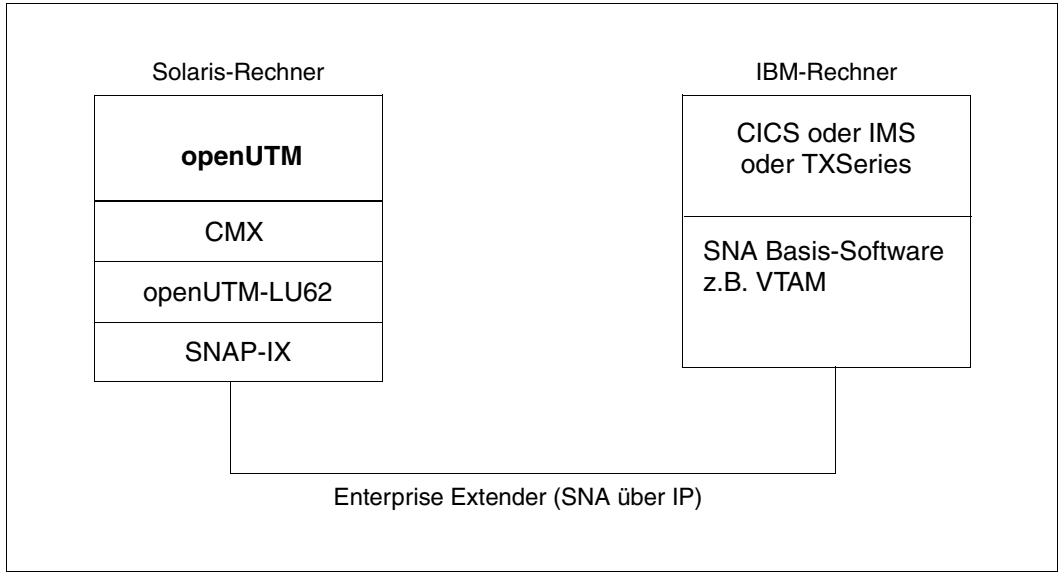
Direkte LU6.2-Kopplung mit openUTM auf Solaris und Einsatz von TRANSIT

Der Solaris-Rechner muss als PU vom Typ 2.1 im SNA-Netz generiert sein.

Im Bild ist eine SNA-Kopplung über Ethernet dargestellt. Zu anderen Kopplungsmöglichkeiten siehe das Handbuch TRANSIT-SERVER.

2.2 Direkte LU6.2-Kopplung über SNAP-IX auf Solaris

Die folgende Kopplungsmöglichkeit kann eingesetzt werden, wenn die UTM-Anwendung auf einem Solaris-Rechner läuft, der direkt an das SNA-Netz angebunden ist.

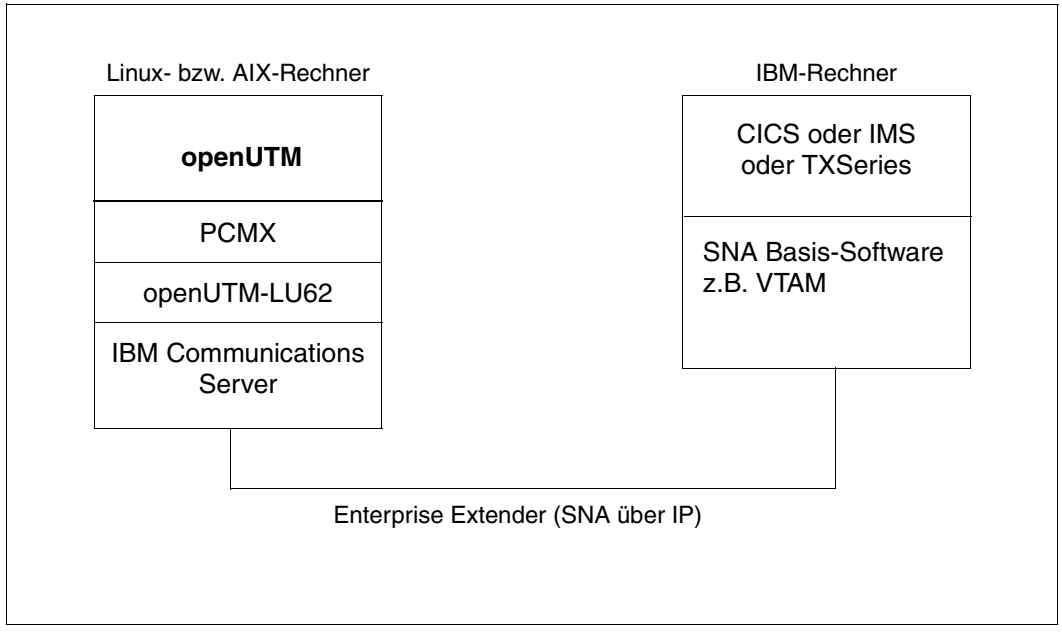


Direkte LU6.2-Kopplung mit openUTM auf Solaris und Einsatz von SNAP-IX

Im Bild ist eine SNA-Kopplung über Enterprise Extender (SNA über IP) dargestellt. Zu anderen Kopplungsmöglichkeiten siehe die Dokumentation zu SNAP-IX.

2.3 Direkte LU6.2-Kopplung über IBM Communications Server auf Linux- oder AIX-Systemen

Die folgende Kopplungsmöglichkeit kann eingesetzt werden, wenn die UTM-Anwendung auf einem Linux- oder AIX-Rechner läuft, der direkt an das SNA-Netz angebunden ist.

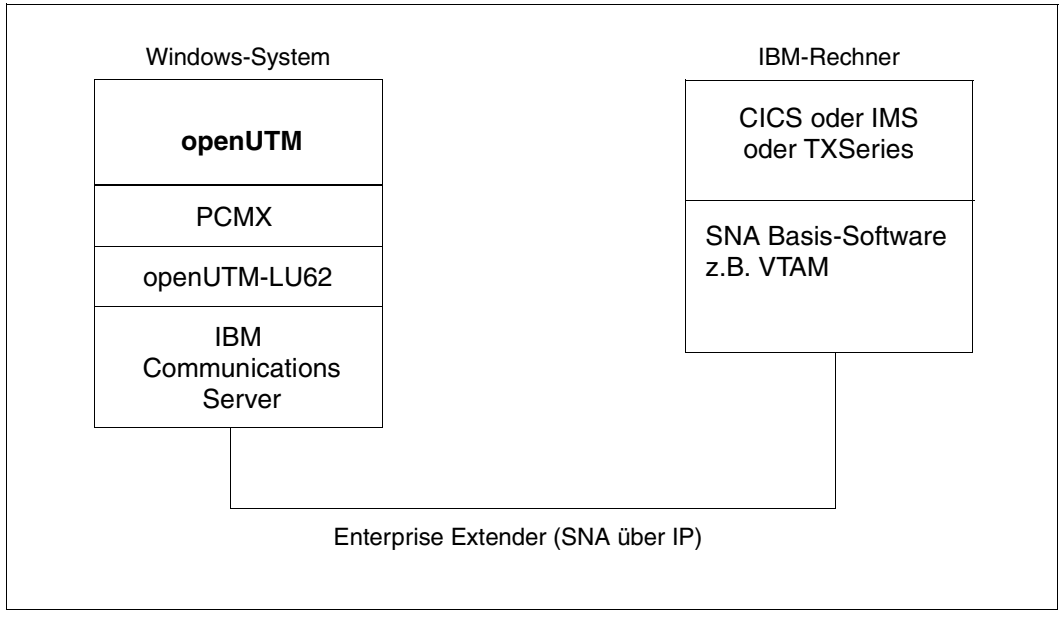


Direkte LU6.2-Kopplung mit openUTM auf Linux- bzw. AIX-Systemen und Einsatz des IBM Communcations Server

Im Bild ist eine SNA-Kopplung über Enterprise Extender (SNA über IP) dargestellt. Zu anderen Kopplungsmöglichkeiten siehe die Dokumentation zu IBM Communications Server.

2.4 Direkte LU6.2-Kopplung über IBM Communications Server auf Windows-Systemen

Die folgende Kopplungsmöglichkeit kann eingesetzt werden, wenn die UTM-Anwendung auf einem Windows-System läuft, das direkt an das SNA-Netz angebunden ist.

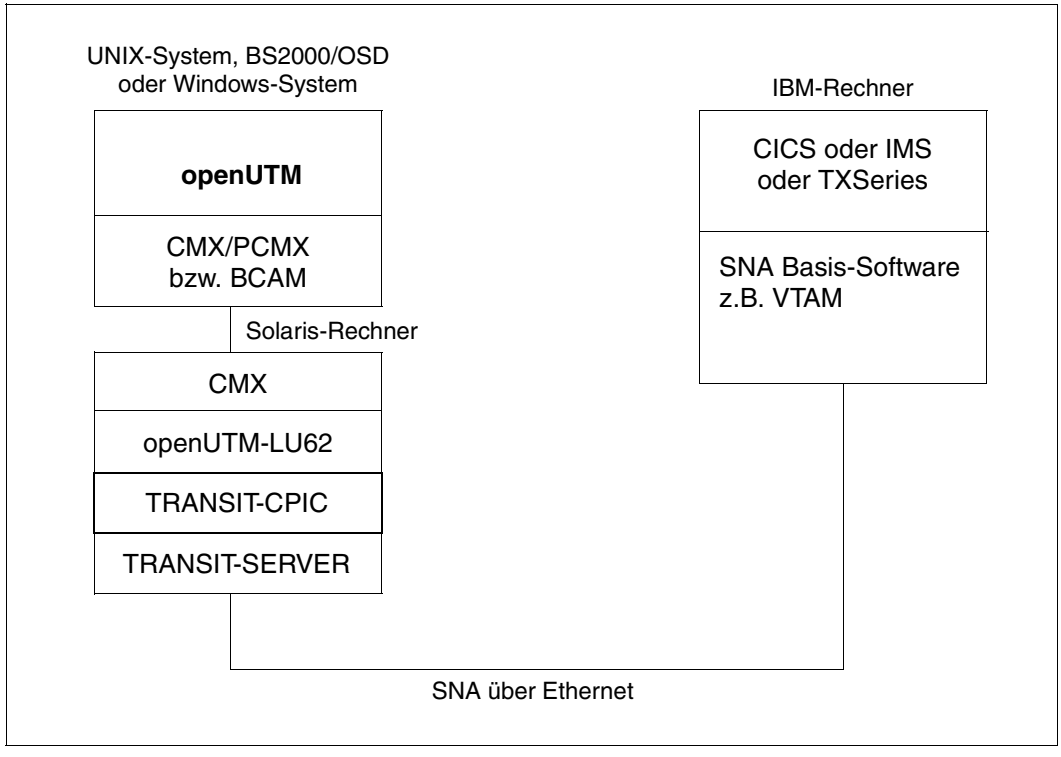


Direkte LU6.2-Kopplung mit openUTM auf Windows-Systemen und Einsatz des IBM Communcations Server

Im Bild ist eine SNA-Kopplung über Enterprise Extender (SNA über IP) dargestellt. Zu anderen Kopplungsmöglichkeiten siehe die Dokumentation zu IBM Communications Server.

2.5 LU6.2-Kopplung über Gateway-Rechner mit TRANSIT auf Solaris

Die folgende Kopplungsmöglichkeit über LU6.2 kann immer dann eingesetzt werden, wenn der Rechner, auf dem die UTM-Anwendung läuft, nicht direkt an das SNA-Netz angebunden werden soll oder kann. Das Betriebssystem dieses Rechners ist beliebig. Es kann z.B. ein BS2000/OSD, ein Solaris- oder Linux-System, ein Windows-System oder ein anderes UNIX-System sein. Als Gateway-Rechner wird ein Solaris-System benötigt.



LU6.2 über Gateway-Rechner mit TRANSIT auf Solaris

Im Falle von openUTM unter BS2000/OSD ist zusätzlich das Produkt OSS(BS2000/OSD) erforderlich.

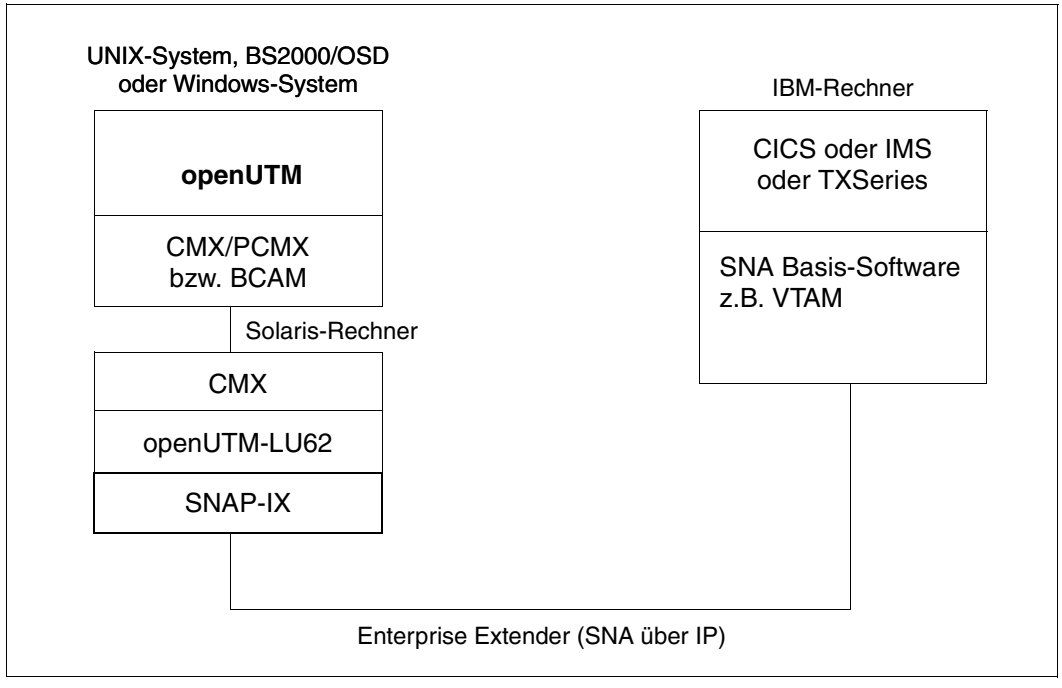
Die Verbindung zwischen den beiden linken Rechnern erfolgt über TCP/IP mit RFC1006.

Der untere Solaris-Rechner muss als PU vom Typ 2.1 im SNA-Netz generiert sein.

Im Bild ist eine SNA-Kopplung über Ethernet dargestellt. Zu anderen Kopplungsmöglichkeiten siehe das Handbuch TRANSIT-SERVER.

2.6 LU6.2-Kopplung über Gateway-Rechner mit SNAP-IX auf Solaris

Die folgende Kopplungsmöglichkeit über LU6.2 kann immer dann eingesetzt werden, wenn der Rechner, auf dem die UTM-Anwendung läuft, nicht direkt an das SNA-Netz angebunden werden soll oder kann. Das Betriebssystem dieses Rechners ist beliebig. Es kann z.B. ein BS2000/OSD, ein Solaris- oder Linux-System, ein Windows-System oder ein anderes UNIX-System sein. Als Gateway-Rechner wird ein Solaris-System benötigt.



LU6.2-Kopplung über Gateway-Rechner mit SNAP-IX auf Solaris

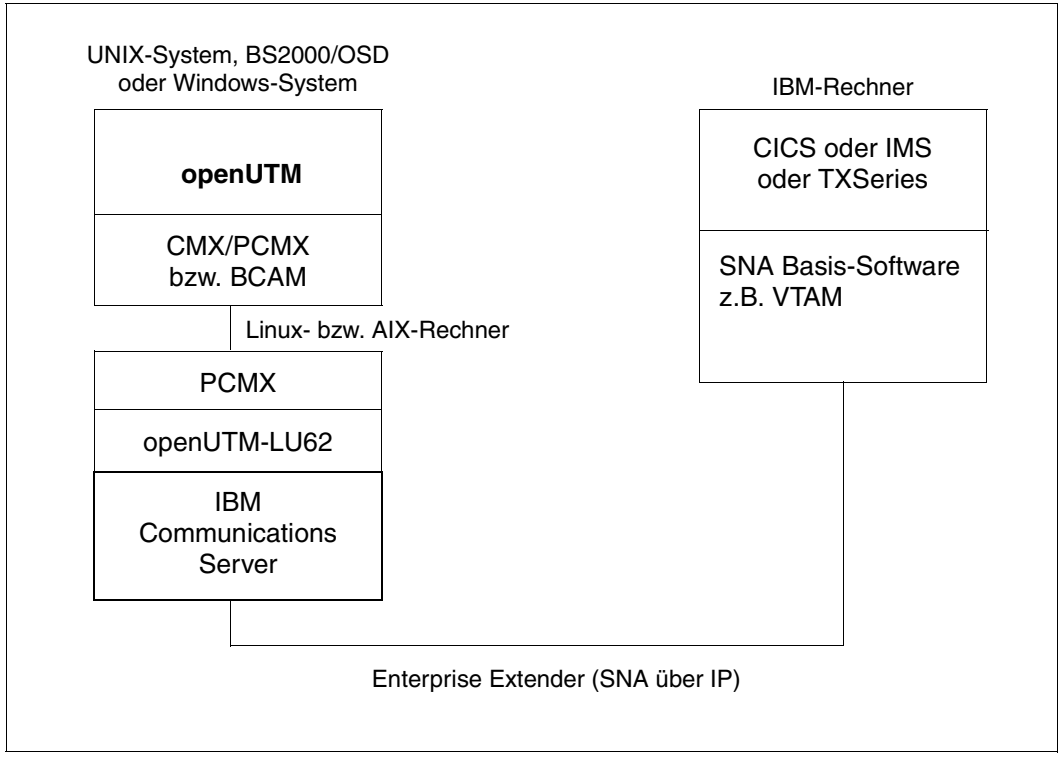
Im Falle von openUTM unter BS2000/OSD ist zusätzlich das Produkt OSS(BS2000/OSD) erforderlich.

Die Verbindung zwischen den beiden linken Rechnern erfolgt über TCP/IP mit RFC1006.

Im Bild ist eine SNA-Kopplung über Enterprise Extender (SNA über IP) dargestellt. Zu anderen Kopplungsmöglichkeiten siehe die Dokumentation zu SNAP-IX.

2.7 LU6.2-Kopplung über Gateway-Rechner mit IBM Communications Server auf Linux- oder AIX-Systemen

Die folgende Kopplungsmöglichkeit über LU6.2 kann immer dann eingesetzt werden, wenn der Rechner, auf dem die UTM-Anwendung läuft, nicht direkt an das SNA-Netz angebunden werden soll oder kann. Das Betriebssystem dieses Rechners ist beliebig. Es kann z.B. ein BS2000/OSD, ein Solaris- oder Linux-System, ein Windows-System oder ein anderes UNIX-System sein. Als Gateway-Rechner wird ein Linux- bzw. AIX-System benötigt.



LU6.2-Kopplung über Gateway-Rechner mit IBM Communications Server auf Linux- bzw. AIX-Systemen

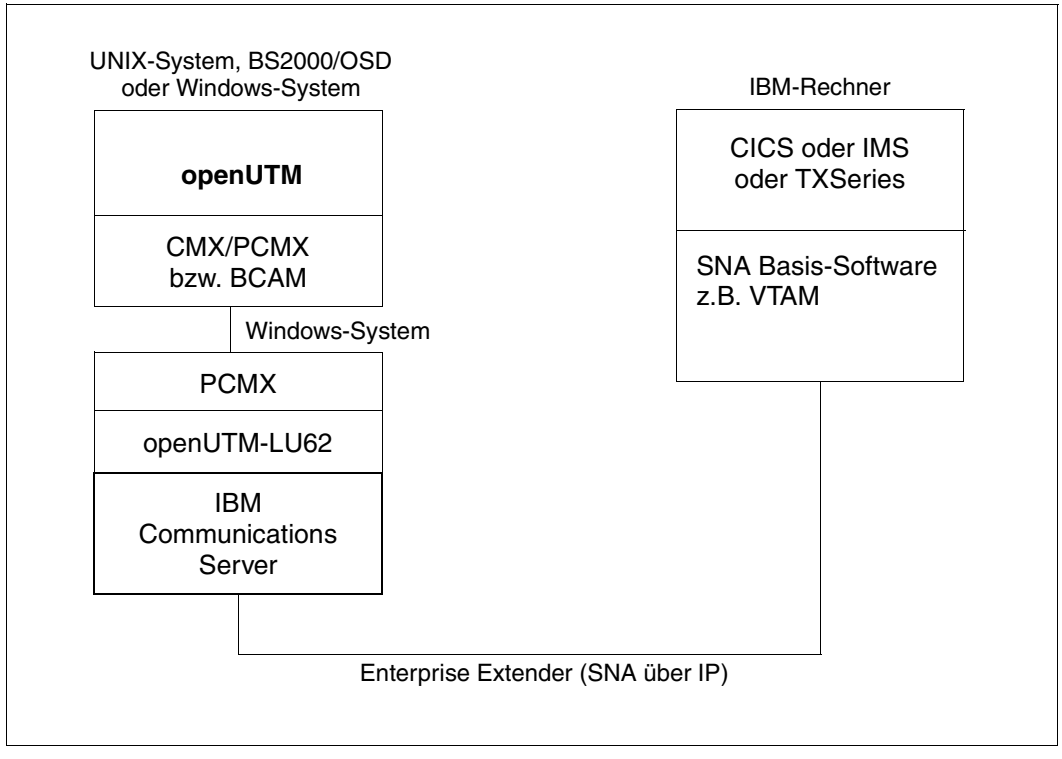
Im Falle von openUTM unter BS2000/OSD ist zusätzlich das Produkt OSS(BS2000/OSD) erforderlich.

Die Verbindung zwischen den beiden linken Rechnern erfolgt über TCP/IP mit RFC1006.

Im Bild ist eine SNA-Kopplung über Enterprise Extender (SNA über IP) dargestellt. Zu anderen Kopplungsmöglichkeiten siehe die Dokumentation zu IBM Communcations Server.

2.8 LU6.2-Kopplung über Gateway-Rechner mit IBM Communications Server auf Windows-Systemen

Die folgende Kopplungsmöglichkeit über LU6.2 kann immer dann eingesetzt werden, wenn der Rechner, auf dem die UTM-Anwendung läuft, nicht direkt an das SNA-Netz angebunden werden soll oder kann. Das Betriebssystem dieses Rechners ist beliebig. Es kann z. B. ein BS2000/OSD, ein UNIX-System oder ein WIndows-System sein. Als Gateway-Rechner wird ein Windows-System benötigt



LU6.2-Kopplung über Gateway-Rechner mit IBM Communications Server auf Windows-Systemen

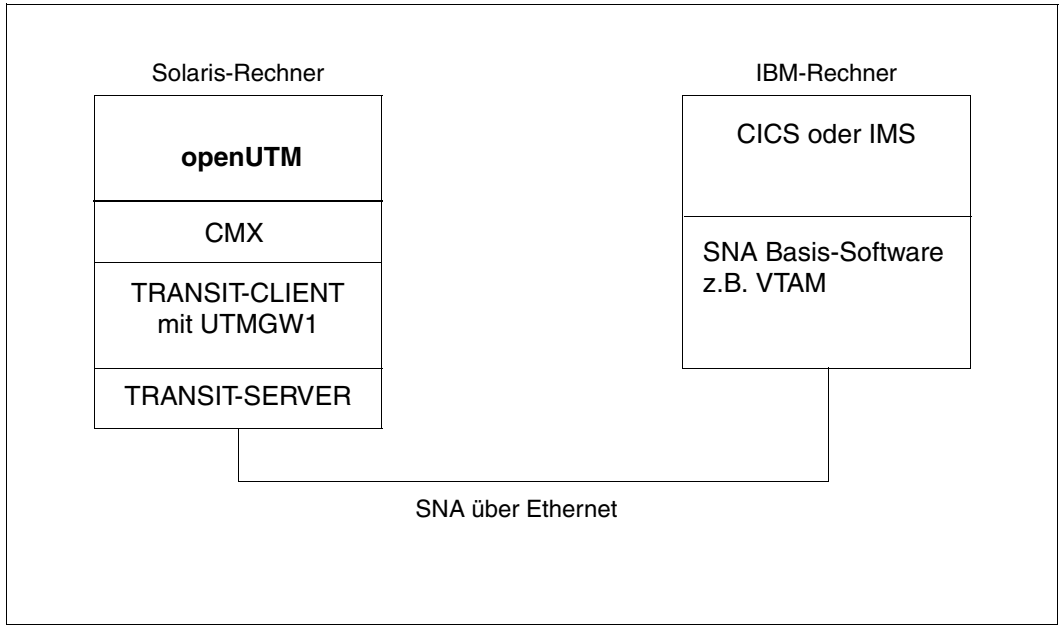
Im Falle von openUTM unter BS2000/OSD ist zusätzlich das Produkt OSS(BS2000/OSD) erforderlich.

Die Verbindung zwischen den beiden linken Rechnern erfolgt über TCP/IP mit RFC1006.

Im Bild ist eine SNA-Kopplung über Enterprise Extender (SNA über IP) dargestellt. Zu anderen Kopplungsmöglichkeiten siehe die Dokumentation zu IBM Communications Server.

2.9 Direkte LU6.1-Kopplung mit openUTM über TRANSIT auf Solaris

Die folgende Kopplungsmöglichkeit kann eingesetzt werden, wenn die UTM-Anwendung auf einem Solaris-Rechner läuft, der direkt an ein SNA-Netz angebunden ist.

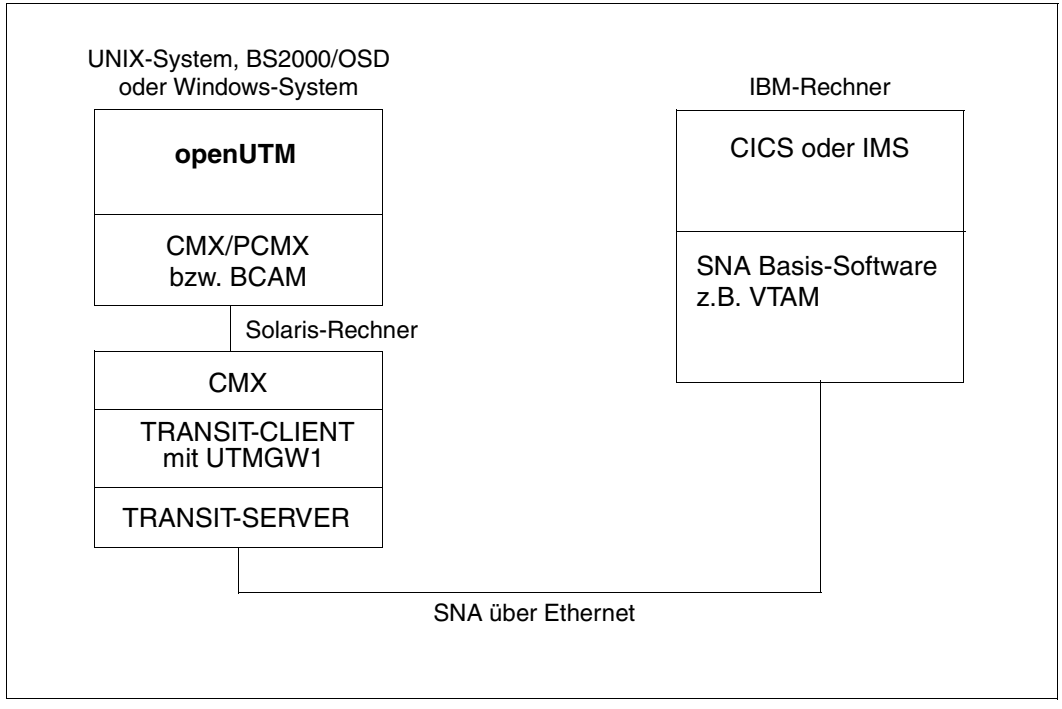


LU6.1-Kopplung über TRANSIT mit openUTM auf Solaris

Im Bild ist eine SNA-Kopplung über Ethernet dargestellt. Zu anderen Kopplungsmöglichkeiten siehe das Handbuch TRANSIT-SERVER.

2.10 LU6.1-Kopplung über Gateway-Rechner mit TRANSIT auf Solaris

Die folgende Kopplungsmöglichkeit über LU6.1 kann immer dann eingesetzt werden, wenn der Rechner, auf dem die UTM-Anwendung läuft, nicht direkt an das SNA-Netz angebunden werden soll oder kann. Das Betriebssystem dieses Rechners ist beliebig. Es kann z.B. BS2000/OSD, ein Solaris- oder Linux-System, ein Windows-System oder ein anderes UNIX-System sein. Als Gateway-Rechner wird ein Solaris-System benötigt.



LU6.1-Kopplung über Gateway-Rechner mit TRANSIT auf Solaris

Die Verbindung zwischen den beiden linken Rechnern erfolgt über TCP/IP mit RFC1006.

Im Bild ist eine SNA-Kopplung über Ethernet dargestellt. Zu anderen Kopplungsmöglichkeiten siehe das Handbuch TRANSIT-SERVER.

3 LU6.2-Kopplungen mit openUTM-LU62

Das Produkt openUTM-LU62 ermöglicht die Kommunikation zwischen UTM-Anwendungen und LU6.2-Anwendungen, d.h. Anwendungen, die das Protokoll LU6.2 verwenden. Beispiele für solche LU6.2-Anwendungen im IBM-Umfeld sind CICS, IMS ab der Version 5, TXSeries- und CPI-C-Anwendungen. Über das Protokoll LU6.2 kann wahlweise eine Kommunikation mit oder ohne Transaktionssicherung betrieben werden.

Im folgenden wird zunächst das allgemeine Konzept von openUTM-LU62 sowie dessen Bedienung beschrieben. Im Kapitel „openUTM-CICS-Kopplung über LU6.2“ auf Seite 75 werden dann die zu berücksichtigenden Details einer openUTM-CICS-Kopplung über LU6.2 dargestellt.

3.1 Konzepte und Funktionen von openUTM-LU62

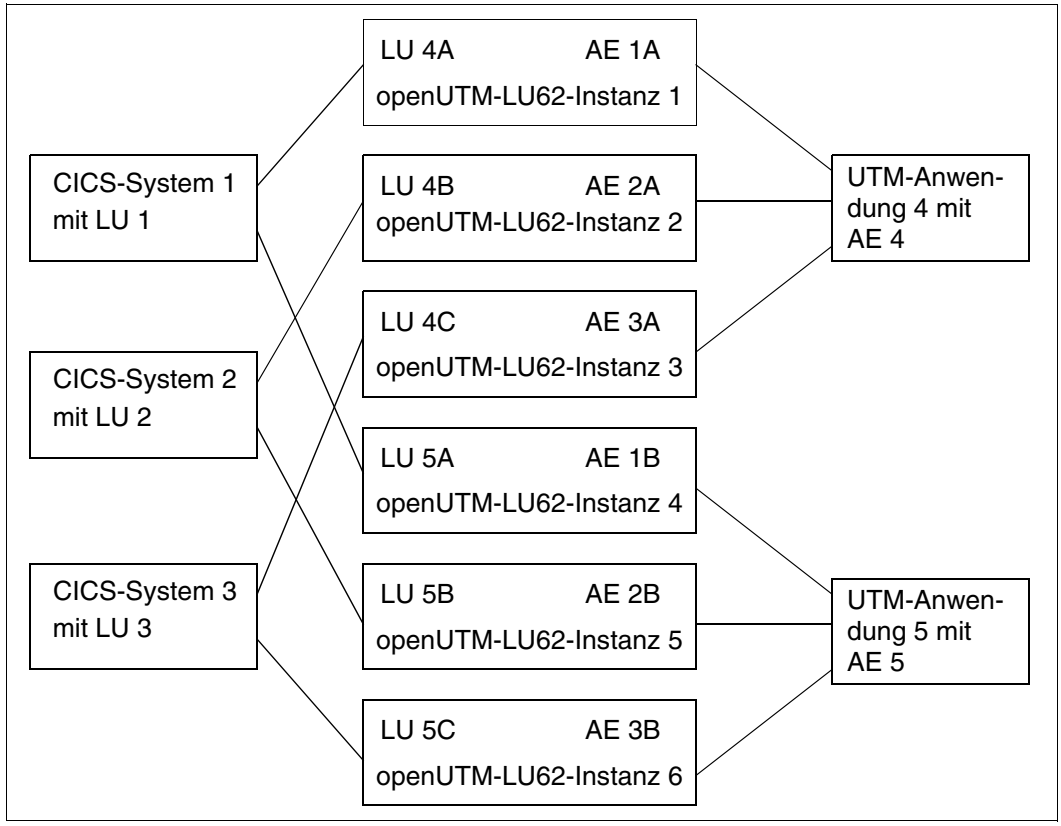
3.1.1 Stellvertreterkonzept

openUTM-LU62 verhält sich gegenüber einer UTM-Anwendung wie eine über OSI-TP gekoppelte Anwendung. openUTM-LU62 realisiert für jede von einer UTM-Anwendung zu erreichende LU6.2-Anwendung, also z. B. für jedes CICS, eine Stellvertreter-OSI-TP-Anwendung.

Aus Sicht der LU6.2-Anwendung ist openUTM-LU62 ein ferner LU6.2-Partner. openUTM-LU62 realisiert für jede von einer LU6.2-Anwendung zu erreichende UTM-Anwendung eine Stellvertreter-LU (Logical Unit).

Pro OSI-CON-Anweisung in einer UTM-Anwendung wird eine Instanz von openUTM-LU62 benötigt. Weitere Informationen zur OSI-CON-Anweisung siehe openUTM-Handbuch „Anwendungen generieren“.

Sollen z.B. von 2 UTM-Anwendungen 3 verschiedene CICS-Systeme erreichen, so sind 6 Instanzen von openUTM-LU62 erforderlich (siehe Abschnitt „INSTANCE-Anweisung“ auf Seite 43). Das Ganze wird an folgendem Bild veranschaulicht:



Kopplung mehrerer Anwendungen mit mehreren Instanzen von openUTM-LU62

Jede UTM-Anwendung benutzt eine Application Entity (AE) als Zugriffspunkt zur OSI-TP-Kommunikation. In der UTM-Generierung wird dieser Zugriffspunkt als Access-Point bezeichnet. Die 3 CICS-Systeme benutzen je eine Logical Unit (LU) als Zugriffspunkt zur LU6.2-Kommunikation.

openUTM-LU62 benutzt Application Entities als Zugriffspunkte zur OSI-TP-Kommunikation und LUs als Zugriffspunkte zur LU6.2-Kommunikation. In einer openUTM-LU62-Instanz ist jeweils genau eine Application Entity und genau eine LU realisiert.

Die Instanz 1 von openUTM-LU62 realisiert die Verbindung LU 1 - AE 4. Die Instanz 2 von openUTM-LU62 realisiert die Verbindung LU 2 - AE 4.

Will also z.B. die UTM-Anwendung 4 eine Verbindung zum CICS-System 1 aufbauen, so muss sie statt dessen eine Verbindung zur AE 1A aufbauen. Will das CICS-System 3 eine Verbindung zur UTM-Anwendung 5 aufbauen, so muss es statt dessen die LU 5C ansprechen.

Somit sind die LUs 4A, 4B, 4C die Stellvertreter für die UTM-Anwendung 4, d.h. für die AE 4, im SNA-Netz. Die AEs 1A, 1B sind die Stellvertreter für das CICS-System 1, d.h. für die LU 1, im ISO-Netz.

3.1.2 Architektur von openUTM-LU62

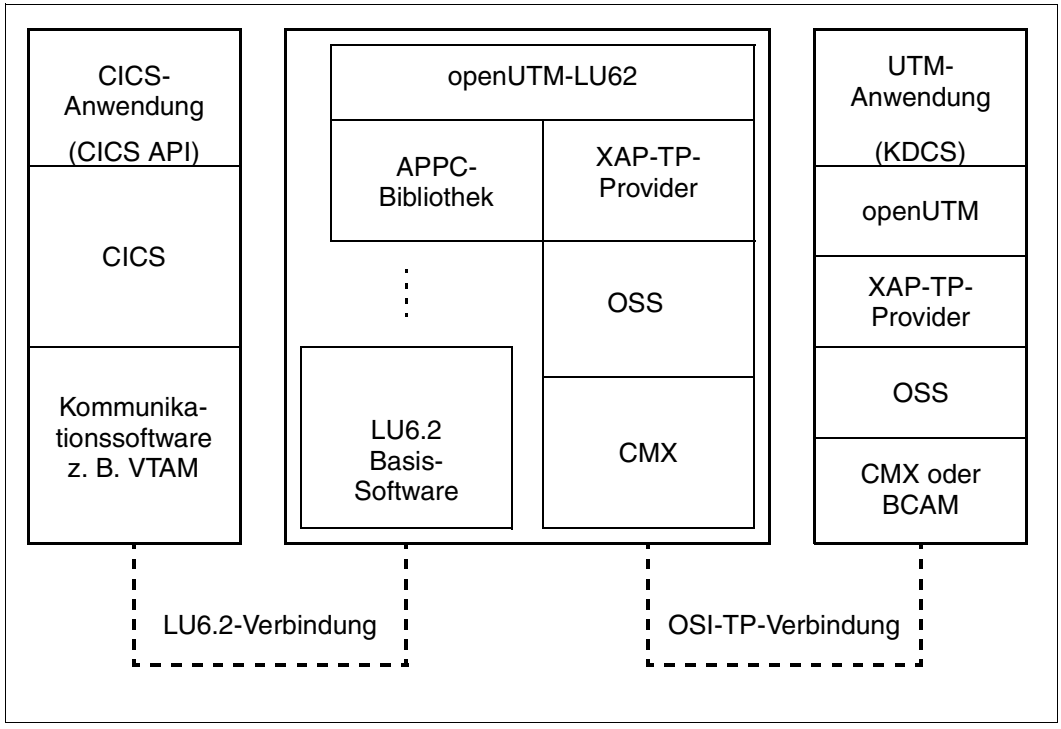
openUTM-LU62 nutzt zur OSI-TP-Kommunikation die von X/Open standardisierte System-Programmschnittstelle XAP-TP. Diese Programmschnittstelle wird auch in den Produkten openUTM und openUTM-Client (Trägersystem OpenCPIC) benutzt. Der XAP-TP-Provider setzt auf der Komponente OSS auf, OSS wiederum setzt in einigen Betriebssystemen auf CMX, bei anderen auf PCMX auf. XAP-TP-Provider und OSS sind im Produkt openUTM-LU62 enthalten.

Zur LU6.2-Kommunikation nutzt openUTM-LU62 die APPC-Programmschnittstelle. Diese Programmschnittstelle wird jeweils durch die LU6.2-Basis-Software realisiert. Als LU6.2-Basissoftware kommen in Frage:

- TRANSIT-SERVER und TRANSIT-CPIC auf Solaris
- SNAP-IX auf Solaris
- IBM Communications Server auf Windows, Linux oder AIX.

openUTM-LU62 setzt nicht nur LU6.2-Protokolle in OSI-TP-Protokolle um und umgekehrt, sondern stellt einen eigenen Knoten im Transaktionsbaum dar, schreibt also Log-Records und führt im Falle eines Verbindungs- oder Rechnerausfalls eigenständig ein Transaktions-Recovery auf OSI-TP-Seite und auf LU6.2-Seite aus.

Im folgenden Bild ist diese Architektur dargestellt:



Architektur von openUTM-LU62

Die fett umrandeten Kästchen stellen Rechnergrenzen dar. Linien innerhalb eines Kästchens bezeichnen Grenzen zwischen den verschiedenen Software-Komponenten, teilweise ist der Name der Programmschnittstelle auf der Linie in Klammern angegeben.

Wie im Abschnitt „Direkte LU6.2-Kopplung über SNAP-IX auf Solaris“ auf Seite 26 beschrieben, können im Falle von openUTM unter UNIX- und Windows-Systemen der mittlere und der rechte Rechner auch identisch sein. Zur Kommunikation zwischen openUTM und openUTM-LU62 wird dann der Local-Loop von CMX über die IP-Adresse 127.0.0.1 benutzt.

3.1.3 Komponenten von openUTM-LU62

openUTM-LU62 besteht aus folgenden Komponenten:

– Das Programm `u62_start` dient zum Starten der openUTM-LU62-Instanzen und der Write-Log-Prozesse. Es überwacht danach die laufenden openUTM-LU62-Prozesse.

– Das Programm `u62_tp` übernimmt die Protokollabbildung zwischen OSI-TP und LU6.2. Es wird von `u62_start` pro openUTM-LU62-Instanz einmal gestartet.

– Das Write-Log-Programm `u62_wlog` sorgt für das Schreiben der Log-Records auf Datei. Es wird pro openUTM-LU62-Instanz als sogenannter Write-Log-Dämon einmal gestartet, wenn Transaktionssicherung verwendet wird.

– Zusätzlich gibt es Dienstprogramme für Generierung (`u62_gen`) und für Administration (`u62_adm`, `u62_sta`).

– Auf Windows-Systemen gibt es zusätzlich das Programm `u62_svc`. Dieses wird verwendet, wenn openUTM-LU62 automatisch beim Systemstart als Service gestartet werden soll.

Alle Programme mit Ausnahme von `u62_sta` liegen auf UNIX-Systemen im Dateiverzeichnis `/opt/lib/utmlu62` und können standardmäßig von jedem Benutzer aufgerufen werden. Es ist möglich, die Rechte zur Administration auf bestimmte Benutzer des UNIX-Systems einzugrenzen, siehe Abschnitt „Administration unter UNIX- und Windows-Systemen“ auf Seite 54.

Das Programm `u62_sta` liegt auf UNIX-Systemen unter `/opt/bin` und kann von jedem Benutzer aufgerufen werden.

Auf Windows-Systemen liegen die Administrationsprogramme im Dateiverzeichnis `Programme\utmlu62`, wobei `Programme\utmlu62` das Standard-Installationsverzeichnis des openUTM-LU62 ist.

Die Interprozesskommunikation zwischen `u62_tp` einerseits und `u62_wlog` bzw. `u62_adm` bzw. `u62_sta` ist auf UNIX-Systemen mittels Named Pipes und Shared Memory realisiert. Auf Windows-Systemen laufen `u62_tp` und `u62_wlog` als Threads im selben Prozess (`u62_tp.exe`). Die Interprozesskommunikation zwischen diesem Prozess und `u62_adm` bzw. `u62_sta` ist mittels Socket und Shared Memory realisiert.

3.1.4 Recovery-Funktionen

Wenn die UTM- und LU6.2-Anwendungsprogramme eine transaktionsgesicherte Kommunikation betreiben, sorgt openUTM-LU62 im Falle eines Verbindungsverlusts oder eines Rechnerabsturzes für ein Recovery der Transaktion, der bei Update-Transaktionen die Konsistenz der beteiligen Datenbanken garantiert.

Geht während des Transaktionsendes die Verbindung zwischen openUTM-LU62 und der LU6.2-Anwendung oder der UTM-Anwendung verloren, so sorgt openUTM-LU62 für eine ordnungsgemäße Beendigung der Transaktion, sobald die Partner-Anwendung wieder erreichbar ist.

openUTM-LU62 schreibt in den kritischen Situationen während der Transaktionsbeendigung Log-Records auf Platte. Wenn die Transaktion komplett beendet ist, werden diese Log-Records wieder gelöscht. Kommt es während des Transaktionsendes zu einem Rechnerabsturz oder zu einer abnormalen Beendigung von openUTM-LU62, so bleiben die Log-Records stehen. Beim nächsten Start von openUTM-LU62 werden die Log-Records dann eingelesen und die offenen Transakionen unter Beteiligung der beiden Anwendungen zu Ende geführt.

Die Log-Records werden auf UNIX-Systemen pro openUTM-LU62-Instanz in der Synclog-Datei `/opt/lib/utmlu62/sync.log.`*loc_lu_alias* gespeichert, wobei *loc_lu_alias* durch den aktuellen Wert der entsprechenden openUTM-LU62-Instanz ersetzt ist. Diese Datei entspricht also in etwa der KDCFILE einer UTM-Anwendung. Auf Windows-Systemen liegt die Datei `sync.log.`*loc_lu_alias* im Dateiverzeichnis `Programme\utmlu62`, wobei *loc_lu_alias* der Wert des Parameters der IINSTANCE-Anweisung ist (siehe Abschnitt „INSTANCE-Anweisung“ auf Seite 43).

openUTM-LU62 bietet auch die Möglichkeit eines Kaltstarts. Beim Kaltstart werden die vorhandenen Log-Records gelöscht. Generell sollte ein Kaltstart nur in Ausnahmefällen verwendet werden. Nach Rechnerabsturz oder abnormaler Beendigung von openUTM-LU62 sollte auf jeden Fall ein Warmstart durchgeführt werden.

Wird jedoch zwischen dem Rechnerabsturz und dem Warmstart von openUTM-LU62 die LU6.2-Anwendung mit Kaltstart neu gestartet, so muss auch die entsprechende openUTM-LU62-Instanz mit Kaltstart hochgefahren werden. In diesem Fall werden die offenen Transaktionen nicht abgeschlossen.

Bei CICS kann im laufenden Betrieb für eine Connection das Kommando „CEMT SET CONNECTION NOTPENDING“ eingegeben werden. Dieses Kommando löscht die in CICS gespeicherten Log-Records für die betreffende Verbindung. Wenn dieses Kommando für die Connection zur UTM-Anwendung abgesetzt wird, hat es hat also dieselben Auswirkungen wie ein Kaltstart.

Wenn durch einen Bedienungsfehler entweder nur CICS oder nur die openUTM-LU62-Instanz mit Kaltstart hochgefahren und das andere System mit Warmstart gestartet wird, so wird dies beim Verbindungsaufbau zwischen openUTM-LU62 und CICS bemerkt. In diesem Fall gibt openUTM-LU62 eine Meldung aus und bricht den Verbindungsaufbau zu CICS ab. Auf CICS-Seite kann man diesen Zustand mittels „CEMT INQUIRE CONNECTION“ abfragen (Anzeige „Xno“).

Wenn eine openUTM-LU62-Instanz warm gestartet wurde und Recovery-Informationen verfügbar sind, CICS dagegen wegen Kaltstart oder „CEMT SET CONNECTION NOTPENDING“ keine Recovery-Informationen mehr hat, so müssen Sie die entsprechende openUTM-LU62-Instanz kalt starten. Wenn die openUTM-LU62-Instanz kalt gestartet wurde und CICS den Aufbau von transaktionsgesicherten Verbindungen wegen Exchange-Log-Name-Fehler ablehnt, müssen Sie in CICS das Kommando „CEMT SET CONNECTION NOTPENDING“ ausführen.

Die Kalt-/Warmstart-Koordination zwischen openUTM-LU62 und der UTM-Anwendung kann von openUTM-LU62 nicht überprüft werden, da es im OSI-TP-Protokoll keine Methoden hierfür gibt. Zwischen openUTM-LU62 und der UTM-Anwendung muss genauso sorgfältig Kalt- und Warmstart abgesprochen werden.

3.1.5 Einschränkungen bei der Protokollabbildung

openUTM-LU62 bildet die Protokolle LU6.2 und OSI-TP aufeinander ab. Da die beiden Protokolle in ihrem Funktionsumfang nicht genau identisch sind, kann openUTM-LU62 nur die gemeinsame Schnittmenge abbilden. Details sind in den Abschnitten Hinweise zur openUTM-Programmierung und Hinweise zur CICS-Programmierung im folgenden Kapitel beschrieben.

3.2 openUTM-LU62 generieren

openUTM-LU62 benötigt zum Start bestimmte Konfigurierungsinformationen, die der Systemverwalter in Form einer Konfigurationsdatei bereitstellen muss. Das Erzeugen der Konfigurationsdatei erfolgt in 2 Schritten:

1. Erstellen einer sogenannten Generierungsdatei mit einem Texteditor. Der Aufbau der Generierungsdatei ist im nachfolgenden Abschnitt beschrieben.
2. Erzeugen der binären Konfigurationsdatei aus der Generierungsdatei. Hierzu muss das Programm `u62_gen` gestartet werden. Genaueres siehe weiter unten.

3.2.1 Aufbau der Generierungsdatei

In der Generierungsdatei müssen alle Stellvertreter-Anwendungen und deren Zuordnung zu den echten Partnern definiert werden. Hierfür gibt es die INSTANCE-Anweisung. Pro openUTM-LU62-Instanz ist eine INSTANCE-Anweisung notwendig. In einer Generierungsdatei können beliebig viele INSTANCE-Anweisungen enthalten sein.

Format der INSTANCE-Anweisung

Das Wort INSTANCE muss allein in einer Zeile stehen.

Alle Parameter der INSTANCE-Anweisung bestehen aus einem Schlüsselwort, dem Zeichen '=' und einem Wert.

Parameter der INSTANCE-Anweisung müssen durch ein Komma getrennt werden, wobei das Komma aber nie am Zeilenanfang stehen darf. Dem letzten Parameter darf kein Komma folgen. Optionale Parameter werden in eckigen Klammern dargestellt. Für fehlende optionale Parameter wird der Standardwert angenommen. Die Reihenfolge der Parameter innerhalb der INSTANCE-Anweisung ist beliebig. Ein Parameter muss immer innerhalb einer Zeile stehen.

Zeilen, die mit einem Stern (*) oder Nummernkreuz (#) beginnen, werden als Kommentar behandelt und von `u62_gen` überlesen.

Zulässige Zeichen für Namen sind alle druckbaren ASCII-Zeichen (Buchstaben, Ziffern, Sonderzeichen) mit Ausnahme der folgenden Sonderzeichen:

– Stern (*)
– Komma (,)
– Semikolon (;)
– Leerzeichen (0x20)
– Tabulator (0x09)

3.2.2 INSTANCE-Anweisung

Mit einer INSTANCE-Anweisung legen Sie die Stellvertreter und die Parameter fest für die Verbindung zwischen einem UTM-Accesspoint und einer LU6.2-Anwendung, d.h. für eine openUTM-LU62-Instanz.

Ab der Version 5.1 von openUTM-LU62 können Sie wählen, ob Sie die TCP/IP-Adress-Information im TNSX oder in der INSTANCE-Anweisung generieren. Wenn die Information in der INSTANCE-Anweisung generiert wird, entfällt die Verwendung von TNSX.

Stellvertreter-LU

Die Stellvertreter-LU wird mit dem Parameter LOC-LU-ALIAS definiert.

Stellvertreter-AE

Wenn Sie die TCP/IP-Adress-Information in der INSTANCE-Anweisung generieren, muss die Stellvertreter-AE mit den Parametern LOC-APT, LOC-AEQ, LOC_TSEL und LOC-LISTENER-PORT generiert werden.

Bei Verwendung von TNSX sind stattdessen die Parameter LOC-APT, LOC-AEQ und LOC-AE zu verwenden.

Verbindung von openUTM-LU62 und LU6.2-Anwendung

Die Verbindung zwischen openUTM-LU62 und der LU6.2-Anwendung wird mit den Parametern REM-LU-ALIAS, MODENAME, ALLOC-TIME und LU62-CODE definiert.

Verbindung zwischen openUTM-LU62 und UTM-Anwendung

Wenn Sie die TCP/IP-Adress-Information in der INSTANCE-Anweisung generieren, sind zur Generierung der Verbindung zwischen openUTM-LU62 und der UTM-Anwendung mindestens die Parameter REM-APT, REM-AEQ, REM-NSEL, REM-TSEL, REM-LISTENER-PORT und CONTWIN erforderlich.

Bei Verwendung von TNSX sind stattdessen die Parameter REM-APT, REM-AEQ, REM-AE und CONTWIN zu verwenden.

Mit den restlichen Parametern (APPL-CONTEXT, ASSOCIATIONS, CONNECT, OSI-TP-CODE, UTM-ASYNC) können weitere Parameter der Verbindung zur UTM-Awendung beschrieben werden.

Syntax der INSTANCE-Anweisung:

```
INSTANCE
        LOC-LU-ALIAS = loc_lu_alias_name,
      [ LOC-LISTENER-PORT = loc_port, ]
      [ LOC-TSEL = { T | A | E }'loc_tsel', ]
      [ LOC-AE = loc_tns_name, ]
        LOC-APT = loc_ap_title,
        LOC-AEQ = loc_ae_qualifier,
        REM-LU-ALIAS = rem_lu_alias_name,
        MODENAME = mode_name,
      [ ALLOC-TIME = alloc_time, ]
      [ LU62-CODE = { ASCII |
                      EBCDIC-500 |
                      EBCDIC-273 |
                      EBCDIC-037 |
                      *NO },
      ]
      [ REM-LISTENER-PORT = rem_port, ]
      [ REM-TSEL = { T | A | E }'rem_tsel',  ]
      [ REM-NSEL = rem_host, ]
      [ REM-AE = rem_tns_name, ]
        REM-APT = rem_ap_title,
        REM-AEQ = rem_ae_qualifier,
      [ APPL-CONTEXT = { UDTCCR | UDTSEC | UDTAC | UDTDISAC }, ]
      [ ASSOCIATIONS = ass_number, ]
      [ CONNECT = conn_number, ]
        CONTWIN = cwin_number,
      [ OSITP-CODE = { ASCII | EBCDIC-BS2 | *NO }, ]
      [ UTM-ASYNC = { YES | NO } ]
```

Befinden sich mehrere INSTANCE-Anweisungen in der Generierungsdatei, so darf das Tripel REM-APT, REM-AEQ, REM-LU-ALIAS nur in einer INSTANCE-Anweisung vorkommen.

Die Parameter haben folgende Bedeutung:

ALLOC-TIME = alloc_time

Zeit in Sekunden, die maximal verstreichen darf, wenn beim Starten eines LU6.2-Auftrags (also z. B. eines CICS-Transaktionscodes) von openUTM aus auf einen Verbindungsaufbau zur LU6.2-Anwendung gewartet werden muss oder wenn beim Starten eines UTM-Transaktionscodes von der LU6.2-Anwendung aus auf einen Verbindungsaufbau zu openUTM gewartet werden muss.

Standardwert: 30 Sekunden.
Mögliche Werte: 0 - 300 Sekunden.
Der Wert 0 bedeutet keine Zeitüberwachung.

APPL-CONTEXT = { UDTCCR | UDTSEC | UDTAC | UDTDISAC }

Name des Application Context, der für die Verbindung zur UTM-Anwendung benutzt wird. Der Wert muss dem Wert des APPLICATION-CONTEXT-Operanden aus der zugehörigen OSI-LPAP-Anweisung in der UTM-Anwendung entsprechen.

Mögliche Werte sind UDTCCR, UDTSEC, UDTAC, UDTDISAC. Diese Werte haben dieselbe Bedeutung wie bei openUTM.

Standardwert ist UDTSEC.

Wenn UDTSEC nicht verwendet wird, ist es nicht möglich, Benutzerkennung und eventuell Passwort beim Conversation-Aufbau zu benutzen.

Bei UDTAC oder UDTDISAC arbeitet openUTM-LU62 ohne Transaktionssicherung.

ASSOCIATIONS = ass_number

Maximale Anzahl paralleler Verbindungen zwischen openUTM-LU62 und UTM-Anwendung. Der Wert sollte dem Wert des ASSOCIATIONS-Parameter aus der zugehörigen OSI-LPAP-Anweisung in der UTM-Generierung entsprechen. Wenn TRANSIT-SERVER verwendet wird, sollte er auch mit dem Wert von SESS-MAX in der XMODE-Anweisung von TRANSIT-SERVER übereinstimmen. Wenn SNAP-IX oder der IBM Communications Server verwendet wird, sollte er mit dem Session-Limit in der Mode-Definition übereinstimmen.

Minimalwert: 1
Standardwert: 1
Maximalwert: 254

CONNECT = conn_number

Anzahl der Verbindungen zur UTM-Anwendung, die beim Start einer openUTM-LU62-Instanz automatisch aufgebaut werden sollen. Zulässige Werte sind ganze Zahlen von 0 bis *ass_number*.

Der Standardwert ist 0.

Kann openUTM-LU62 die gewünschte Anzahl von Verbindungen beim Start nicht aufbauen, erfolgt alle 5 Minuten ein neuer Versuch.

Der CONNECT-Parameter wirkt über den ganzen Lebenslauf des openUTM-LU62-Prozesses, es sei denn, eine Verbindung wird mittels Administrationskommando abgebaut. Wird die bei CONNECT angegebene Zahl von Verbindungen von der Partner-UTM-Anwendung aufgebaut, so baut openUTM-LU62 keine zusätzlichen Verbindungen auf.

CONTWIN = cwin_number

Anzahl der Verbindungen zur UTM-Anwendung, in der openUTM-LU62 Contention-Winner ist. Die Summe aus *cwin_number* und dem Wert des CONTWIN-Parameters aus der zugehörigen OSI-LPAP-Anweisung der UTM-Generierung sollte genau *ass_number* ergeben.

LOC-AE = loc_tns_name

Nur bei Verwendung von TNSX.
Maximal 18-stelliger globaler Name aus dem TNSX (Transport Name Service in UNIX), der die Stellvertreter-AE beschreibt. *loc_tns_name* besteht aus fünf Namensteilen, jeweils durch Punkte getrennt:

NP5.NP4.NP3.NP2.NP1

Sind Namensteile leer, können Punkte am Namensende entfallen.

Im TNSX muss *loc_tns_name* als globaler Name eingetragen sein mit der Eigenschaft „Lokaler Name“. Dabei müssen neben einem T-Selektor für die Schicht 4 auch ein P- und ein S-Selektor eingetragen sein. Diese müssen zu den entsprechenden Parametern einer OSI-CON-Anweisung aus der UTM-Generierung passen. Es empfiehlt sich, Leereinträge für S- und P-Selektoren zu wählen.

LOC-AEQ = loc_ae_qualifier

Gibt den Application Entity Qualifier der Stellvertreter-AE an. *loc_ae_qualifier* muss eine ganze Zahl sein zwischen 1 und 67108863. Sie muss dem Wert des AEQ-Operanden aus der zugehörigen OSI-LPAP-Anweisung der UTM-Generierung entsprechen.

LOC-APT = loc_ap_title

Gibt den Application Process Title der Stellvertreter-AE in Form eines Objektbezeichners an, also in der Form

(n1, n2, n3,...)

Dabei kann *n1* die Werte 0, 1, oder 2 und *n2* die Werte 0 bis 39 annehmen. Der Application Process Title muss dem Wert des APT-Operanden aus der zugehörigen OSI-LPAP-Anweisung der UTM-Generierung entsprechen.

LOC-LISTENER-PORT = loc_port

Nur, wenn kein TNSX verwendet wird, dann aber Pflichtparameter.
TCP-Portnummer der Stellvertreter-AE mit dem Gültigkeitsbereich 102, 1025 - 65535.

LOC-LU-ALIAS = loc_lu_alias_name

Maximal achtstelliger LU-Alias-Name der Stellvertreter-LU. Er muss einem LU-Namen in der Konfiguration der LU6.2-Basis-Software entsprechen (TRANSIT-SERVER, SNAP-IX bzw. IBM Communications Server). Dieser Name muss nur zwischen openUTM-LU62 und der LU6.2-Basis-Software abgestimmt sein.

LOC-TSEL = { T | A | E }'loc_tsel'

Nur, wenn kein TNSX verwendet wird, dann aber Pflichtparameter.
Transport-Selektor der Stellvertreter-AE. Das erste Zeichen gibt das Format an:

T TRANSDATA
Zugelassen sind nur Großbuchstaben, Ziffern und die Sonderzeichen @, # und $, wobei das erste Zeichen keine Ziffer sein darf.

A ASCII

E EBCDIC

Für alle Formate beträgt die maximale Länge des Inhalts (`loc_tsel`) 8 Zeichen.

LU62-CODE =

Beschreibung des Zeichensatzes, in dem die Benutzerdaten der LU6.2-Anwendungsprogramme codiert sind.

ASCII Die LU6.2-Anwendungsprogramme verwenden den ASCII-8-Bit-Zeichensatz ISO 8859-1. Dies ist üblich für Anwendungsprogramme auf UNIX-, Windows- oder OS/2-Systemen.

EBCDIC-500 Die LU6.2-Anwendungsprogramme verwenden IBM-EBCDIC mit Code-Page 500. Dies ist die bei IBM auf Großrechnern oder AS/400 übliche internationale Code-Page sowie die nationale Code-Page für Belgien und die Schweiz.

EBCDIC-273 Die LU6.2-Anwendungsprogramme verwenden IBM-EBCDIC mit Code-Page 273. Dies ist die bei IBM auf Großrechnern oder AS/400 übliche nationale Code-Page für Deutschland und Österreich.

EBCDIC-037 Die LU6.2-Anwendungsprogramme verwenden IBM-EBCDIC mit Code-Page 037. Dies ist die bei IBM auf Großrechnern oder AS/400 übliche nationale Code-Page für USA, Kanada, Niederlande und Portugal.

*NO openUTM-LU62 führt keine Code-Konvertierung der Benutzerdaten durch. Dieser Wert muss angegeben werden, wenn die Anwendungsprogramme Binärdaten, also z.B. binäre Längenfelder übertragen. (Standardwert)

openUTM-LU62 führt eine Codekonvertierung aller Benutzerdaten durch, wenn bei LU62-CODE und OSITP-CODE unterschiedliche Zeichensätze generiert sind. Bei *NO führt openUTM-LU62 keine Codekonvertierung durch. Ist hier *NO angegeben, so muss auch bei OSITP-CODE *NO angegeben werden.

MODENAME = mode_name

Maximal achtstelliger Mode-Name, der auf den Verbindungen zur LU6.2-Anwendung benutzt werden soll. Er muss in der Konfiguration der LU6.2-Basis-Software (TRANSIT-SERVER, SNAP-IX bzw. IBM Communications Server) sowie beim Partnersystem ebenfalls definiert sein. Der Mode-Name SNASVCMG darf nicht verwendet werden.

OSITP-CODE =

Beschreibung des Zeichensatzes, in dem die Benutzerdaten der UTM-Anwendungsprogramme codiert sind.

ASCII Die UTM-Anwendungsprogramme verwenden den ASCII-8-Bit-Zeichensatz ISO 8859-1. Dies ist üblich für UTM-Anwendungsprogramme auf UNIX- oder Windows-Systemen.

EBCDIC-BS2 Die UTM-Anwendungsprogramme verwenden die in BS2000/OSD übliche EBCDIC-Variante EBCDIC.DF.04 Latin 1.

*NO openUTM-LU62 führt keine Code-Konvertierung der Benutzerdaten durch. Dieser Wert muss angegeben werden, wenn die UTM-Anwendungsprogramme Binärdaten, also z.B. binäre Längenfelder übertragen. (Standardwert)

openUTM-LU62 führt eine Codekonvertierung aller Benutzerdaten durch, wenn bei OSITP-CODE und LU62-CODE unterschiedliche Zeichensätze generiert sind. Bei *NO führt openUTM-LU62 keine Codekonvertierung durch. Ist hier *NO angegeben, so muss auch bei LU62-CODE *NO angegeben werden.

REM-AE = rem_tns_name

Nur bei Verwendung von TNSX.
Maximal 18-stelliger globaler Name aus dem TNSX (Transport Name Service in UNIX-Systemen), der die Application Entity (d.h den Access-Point) der zugehörigen UTM-Anwendung beschreibt. *rem_tns_name* besteht aus fünf Namensteilen, jeweils durch Punkte getrennt:

NP5.NP4.NP3.NP2.NP1

Sind Namensteile leer, können Punkte am Namensende entfallen.

Im TNSX muss *rem_tns_name* als globaler Name eingetragen sein mit den Eigenschaften „Transportadresse“ und „Transportsystem“. Dabei müssen neben einem T-Selektor für die Schicht 4 auch ein P- und ein S-Selektor eingetragen sein. Diese müssen zu den entsprechenden Parametern der ACCESS-POINT-Anweisung aus der UTM-Generierung passen. Es empfiehlt sich, Leereinträge für S- und P-Selektoren zu wählen.

REM-AEQ = rem_ae_qualifier

Gibt den Application Entity Qualifier der UTM-Anwendung an. *rem_ae_qualifier* muss eine ganze Zahl sein zwischen 1 und 67108863. Sie muss dem Wert des AEQ-Operanden aus der zugehörigen ACCESS-POINT-Anweisung der UTM-Generierung entsprechen.

REM-APT = rem_ap_title

Gibt den Application Process Title der UTM-Anwendung in Form eines Objektbezeichners an, also in der Form

(n1, n2, n3,...)

Dieser muss dem Wert des APT-Operanden aus der zugehörigen UTMD-Anweisung der UTM-Generierung entsprechen.

REM-LISTENER-PORT = rem_port

Nur, wenn kein TNSX verwendet wird, dann aber Pflichtparameter.
TCP-Portnummer der Partner-UTM-Anwendung mit dem Gültigkeitsbereich 102, 1025 - 65535.

REM-LU-ALIAS = rem_lu_alias_name

Maximal achtstelliger LU-Alias-Name für die Remote-LU, unter dem die LU6.2-Anwendung (also z. B. die CICS-Region) erreichbar ist. Er muss einem RLU-Namen in der Konfiguration der LU6.2-Basis-Software entsprechen (TRANSIT-SERVER, SNAP-IX bzw. IBM Communications Server). Dieser Name muss nur zwischen openUTM-LU62 und der LU6.2-Basis-Software abgestimmt sein.

REM-NSEL = rem_host

Nur, wenn kein TNSX verwendet wird.
(DNS-)Name des Hosts, auf dem die Partner-UTM-Anwendung läuft.

Der Host-Name darf keinen *Punkt* (Domänennamen) enthalten und nicht länger als 16 Zeichen sein.

Liegt der Partnerrechner in einer anderen Domäne, so muss z.B. auf UNIX in der Datei `/etc/resolv.conf` der Parameter `search` um den relevanten Domänennamen erweitert werden.

REM-TSEL = { T | A | E }'rem_tsel'

Nur, wenn kein TNSX verwendet wird, dann aber Pflichtparameter.
Transport-Selektor der Partner-UTM-Anwendung. Das erste Zeichen gibt das Format an:

T TRANSDATA
Zugelassen sind nur Großbuchstaben, Ziffern und die Sonderzeichen @, # und $ zugelassen, wobei das erste Zeichen keine Ziffer sein darf.

A ASCII

E EBCDIC

Für alle Formate beträgt die maximale Länge des Inhalts (rem_tsel) 8 Zeichen.

UTM-ASYNC =

Dieser Parameter bestimmt, wie ein von einem UTM-Anwendungsprogramm gestarteter Vorgang ohne Functional Unit Commit und ohne Functional Unit Handshake (d.h. bei APRO mit KCOF=B) an den LU6.2-Partner weitergegeben wird.

NO openUTM-LU62 startet diese Vorgänge mit Sync-Level 0 (NONE).

YES openUTM-LU62 startet diese Vorgänge mit Sync-Level 1 (CONFIRM).

Der Wert YES ist notwendig, wenn aus dem UTM-Anwendungsprogramm mittels APRO AM ein Vorgang beim LU6.2-Partner ohne Transaktionssicherung gestartet werden soll.
Verwendet eine UTM-Anwendung APRO DM mit KCOF=B und APRO AM mit KCOF=B, so sollte für beide Typen der Kommunikation jeweils eine eigene Instanz von openUTM-LU62 eingerichtet werden, wobei die erste Instanz mit UTM-ASYNC=NO und die zweite Instanz mit UTM-ASYNC=YES generiert wird.

3.2.3 Generierungsprogramm starten

Mit dem Generierungsprogramm `u62_gen` erzeugen Sie aus einer Generierungsdatei eine binäre Konfigurationsdatei. Gleichzeitig wird die Generierungsdatei auf logische Fehler überprüft. `u62_gen` darf auf UNIX-Systemen nur vom Systemverwalter aufgerufen werden.

Auf UNIX-Systemen können Sie `u62_gen` direkt aus der Shell aufrufen. Auf Windows-Systemen empfiehlt es sich, mit „Start > Programme > openUTM-LU62 > Kommando-Eingabe“ eine DOS-Shell zu erzeugen. Unter dieser DOS-Shell kann dann das folgende Kommando ohne den Vorspann `/opt/lib/utmlu62/` aufgerufen werden.

Aufrufsyntax:

```
/opt/lib/utmlu62/u62_gen [-f conf_file] [gen_file]
```

Beschreibung:

gen_file Name der Generierungsdatei, aus der die Generierungsparameter gelesen werden sollen. Standardwert ist `/opt/lib/utmlu62/gen/genfile` (UNIX-Systeme) bzw. `Programme\utmlu62\gen\genfile` (Windows-Systeme).

-f *conf_file* (File) Name der Konfigurationsdatei, in die Generierungsdaten in binärer Form geschrieben werden sollen. Ist *conf_file* nicht angegeben, erfolgt die Ausgabe in die Datei `/opt/lib/utmlu62/conffile` (UNIX-Systeme) bzw. `Programme\utmlu62\conffile` (Windows-Systeme).

Das Programm `u62_tp` liest während seiner Initialisierung die Konfigurationsdaten aus der Datei `conffile`. Diese Konfigurationsdatei kann auch bei laufendem `u62_tp` mittels `u62_gen` neu erzeugt werden. Die neuen Konfigurationsdaten werden jedoch erst beim nächsten Start der jeweiligen openUTM-LU62-Instanz wirksam.

`u62_gen` löscht keine für das Recovery benötigten Log-Records. Zum Thema Recovery siehe auch Abschnitt „Recovery-Funktionen“ auf Seite 40.

3.2.4 Generierungsdatei wiederherstellen

Wollen Sie aus einer binären Konfigurationsdatei eine lesbare Generierungsdatei erstellen, um z.B. den Generierungslauf zu überprüfen, so können Sie das mit folgendem Kommando erreichen:

Aufrufsyntax:

```
/opt/lib/utmlu62/u62_gen -r [conf_file]
```

Beschreibung:

-r (Restore)

conf_file Name der binären Konfigurationsdatei. Ist conf_file nicht angegeben, so wird die Datei `/opt/lib/utmlu62/conffile` gelesen.

`u62_gen` gibt die Daten nach `stdout` aus.

Für Windows-Systeme gelten dieselben Anmerkungen wie im Abschnitt „Generierungsprogramm starten“ auf Seite 52.

3.2.5 Namen der verwendeten Generierungsdatei anzeigen

Haben Sie den Namen der verwendeten Generierungsdatei vergessen, so können Sie sich mit dem folgenden Kommando den Dateinamen anzeigen. Das Kommando zeigt zusätzlich die internen Versionsnummern der Programme von openUTM-LU62 an.

Aufrufsyntax:

```
/opt/lib/utmlu62/u62_adm -v
```

Für Windows-Systeme gelten dieselben Anmerkungen wie im Abschnitt „Generierungsprogramm starten“ auf Seite 52.

3.3 openUTM-LU62 administrieren

3.3.1 Administration unter UNIX- und Windows-Systemen

Administration unter UNIX-Systemen

Unter UNIX-Systemen können Sie alle Administrationskommandos wie unten angegeben direkt aus der Shell aufrufen.

openUTM-LU62 ist nach der Installation durch alle Benutzer administrierbar. Soll der Kreis der Administratoren von openUTM-LU62 eingeschränkt werden, so kann man in der Datei `/opt/lib/utmlu62/u62_users` eine Liste mit allen zugelassenen Benutzerkennungen erstellen. Diese Liste hat folgende Syntax:

– Benutzerkennungen sind durch Komma, Leerzeichen, Tabulator oder Zeilenende (Newline) voneinander getrennt. Der Benutzer *root* hat immer Administrationsberechtigung und braucht nicht eingetragen zu werden.

– Kommentare werden durch ein #-Zeichen eingeleitet und reichen bis zum Zeilenende.

Wenn diese Datei keinen einzigen Eintrag enthält oder nicht vorhanden ist, sind alle Benutzer administrationsberechtigt.

Administration unter Windows-Systemen

Unter Windows-Systemen empfiehlt es sich, mit „Start > Programme > openUTM-LU62 > Kommando-Eingabe“ eine DOS-Shell zu erzeugen. Unter dieser DOS-Shell können dann die im folgenden beschriebenen Kommandos ohne den Vorspann `/opt/lib/utmlu62/` aufgerufen werden.

3.3.2 openUTM-LU62 starten

Mit `u62_start` wird openUTM-LU62 als Hintergrundprozess gestartet.

Aufrufsyntax:

```
/opt/lib/utmlu62/u62_start
    [ -l loc_lu_alias_name ]
    [ -ton [,trace_optionen] ]
    [ { -c | -k } ]
```

Beschreibung:

-l *loc_lu_alias_name*

Wenn der Schalter -l nicht angegeben ist, ermittelt `u62_start` anhand der Konfigurationsdatei eine Liste aller openUTM-LU62-Instanzen und startet alle diese Instanzen. Ist der Schalter -l angegeben, so wird nur die zur LU *loc_lu_alias_name* gehörende Instanz gestartet. *loc_lu_alias_name* muss einem Namen aus der openUTM-LU62-Konfigurierung entsprechen.

-ton[,*trace_optionen*]

(Trace on) Der Trace wird eingeschaltet. Die Ausführlichkeit des Trace kann durch *trace_optionen* gesteuert werden.

Bei *trace_optionen* kann angegeben werden, ob und mit welcher Ausführlichkeit der Instanz-Trace und ob der XAP-Trace eingeschaltet werden soll. Mit dem Instanz-Trace (IN) werden alle internen Abläufe des Programms `u62_tp` protokolliert. Für diesen Instanz-Trace gibt es verschiedene Trace-Levels, die durch Zahlen dargestellt werden. Jedes höhere Level schließt die Traces aller niedereren Levels mit ein. Mit dem XAP-TP-Trace (XAP) werden alle internen Abläufe der Komponenten XAP-TP-Provider und OSS protokolliert. Im Einzelnen sind folgende Angaben bei *trace_optionen* möglich:

in=1 Der Instanz-Trace wird mit dem niedrigsten Level eingeschaltet. Es werden nur die für einen Protokoll-Trace erforderlichen Trace-Einträge geschrieben. Zum Protokoll-Trace siehe Seite 66. Dieses Trace-Level ist im Allgemeinen für eine erste Fehlerdiagnose ausreichend.

in=2 Der Instanz-Trace wird mit dem ausführlichen Level eingeschaltet.

in=3 Der Instanz-Trace wird mit einer zusätzlichen Zeitmessung für alle Aktivitäten gestartet. Dieses Level ist zur Diagnose von Performance-Problemen gedacht.

in Der Instanz-Trace wird mit dem ausführlichen Level eingeschaltet (wie in=2).

xap Der XAP-TP-Trace wird eingeschaltet.

Sollen beide Traces eingeschaltet werden, so muss in, xap als *trace_optionen* angegeben werden.

Wird *trace_optionen* weggelassen, so werden der Instanz-Trace mit ausführlichem Level und der XAP-TP-Trace eingeschaltet.

{ -c | -k }

Ohne Angabe wird standardmäßig ein Warmstart durchgeführt (`Start-Typ = w`).

-c (Cold) Es wird ein Kaltstart durchgeführt. Der Kaltstart löscht alle Recovery-Informationen. Diesen Parameter sollten Sie nur in Ausnahmefällen benutzen. Siehe hierzu Seite 40.

-k (Lukewarm) Es wird ein Warm-Start durchgeführt, bei dem allerdings alle noch offenen Transaktionen verworfen werden.

Die Log-Namen bleiben dagegen erhalten. Dies bedeutet, dass die LU6.2-Partner-Anwendung nichts von dem Verwerfen der offenen Transaktionen erfährt.

Dies ist zum Beispiel dann notwendig, wenn das Partnersystem entweder durch Fehler oder durch administrative Eingriffe die offenen Transaktionen bereits verworfen hat, aber ein Kaltstart wegen der Auswirkungen auf andere Aufgaben im Partnersystem nicht praktikabel ist. Das Partnersystem muss in diesem Fall einen Kaltstart durchführen.
Zum Beispiel erfordert in IMS ein Kaltstart das Beenden und den Neustart des gesamten IMS und damit eine vollständige Unterbrechung des Betriebs, was in Produktivsystemen naturgemäß nicht erwünscht ist. Durch Starten mit dem Parameter `-k` können einseitig hängengebliebene Transaktionen bereinigt werden, ohne dass im Partnersystem weitere Maßnahmen, insbesondere eine Unterbrechung notwendig sind.

Beispiel: Start mit Zeitmessung im Instanz-Trace und XAP-TP-Trace auf UNIX-Systemen:

```
/opt/lib/utmlu62/u62_start -ton,in=3,xap
```

Nach dem erfolgreichen Start erscheint auf dem Bildschirm die folgende Meldung:

```
u62_start 17  * * *   openUTM-LU62 gestartet:   * * *
                lokaler LU-Alias-Name = TO2CGEI4,
                PID = 1771,
                Start-Typ = w
```

Die Werte bei lokaler LU-Alias-Name und PID sind dabei durch aktuelle Werte ersetzt.

Normalerweise wird man openUTM-LU62 immer beim Systemstart starten und beim System-Shutdown beenden wollen. Auf UNIX-Systemen wird dies erreicht durch die Datei `/etc/rc2.d/S30u62` bzw. eine gleichnamige Datei in einem anderen System-RC-Verzeichnis.

Auf Windows-Systemen kann während der Installation ausgewählt werden, ob openUTM-LU62 als Service gestartet werden soll. Wird dies ausgewählt, so startet das Betriebssystem immer beim Hochfahren automatisch alle Instanzen von openUTM-LU62. Sollen bei diesem automatischen Start Parameter übergeben werden, so müssen diese über den Registrierungseditor des Betriebssystems in die Variable U62_SERVICE_ARGS (im Verzeichnis `HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\openUTM-LU62\CurrentVersion`) eingetragen werden. Wenn Sie eine Instanz über die Dienste-Administration des Windows-Systems starten, wird U62_SERVICE_ARGS nicht ausgewertet. Stattdessen müssen die Startparameter von Hand eingetragen werden.

Wollen Sie ohne Neuinstallation erreichen, dass openUTM-LU62 als Service gestartet wird, so können Sie hierzu auch folgendes Kommando verwenden:

```
u62_start -r
```

Um den Start als Service rückgängig zu machen, können Sie folgendes Kommando verwenden:

```
u62_start -u
```

3.3.3 openUTM-LU62 beenden

Mit `u62_adm -e` (End) wird openUTM-LU62 gestoppt.

Aufrufsyntax:

```
/opt/lib/utmlu62/u62_adm -e [ -l loc_lu_alias_name ]
```

Beschreibung:

-l *loc_lu_alias_name*
: Wenn der Schalter -l nicht angegeben ist, werden alle openUTM-LU62-Instanzen gestoppt. Ist der Schalter -l angegeben, so wird nur die zur LU *loc_lu_alias_name* gehörende Instanz gestoppt.

Auf Windows-Systemen können Sie stattdessen auch „Start > Programme > openUTM-LU62 > Beenden von openUTM-LU62“ verwenden.

Vor dem Beenden von TRANSIT sollten Sie immer openUTM-LU62 beenden.

3.3.4 Zustandsinformationen anzeigen

Mit `u62_sta` können Sie sich über den aktuellen Zustand von openUTM-LU62 informieren. Dieses Kommando kann von jedem Benutzer des UNIX-Systems verwendet werden. Auf Windows-Systemen können Sie stattdessen auch „Start > Programme > openUTM-LU62 > Status-Anzeige“ verwenden.

Aufrufsyntax:

```
u62_sta [ { -l loc_lu_alias_name | -c } ] [ -b ]
```

Beschreibung:

-l *loc_lu_alias_name*
: Mit dem Schalter `-l` werden nur Informationen über die zur LU *loc_lu_alias_name* gehörende Instanz ausgegeben.

-c
: Mit dem Schalter `-c` wird nur die Anzahl aller LU62-Sessions der lokalen Default-LU mit dem Mode CPSVCMG aufgelistet.

  Wird weder der Schalter `-l` noch `-c` angegeben, werden Zustandsinformationen über alle openUTM-LU62-Instanzen ausgegeben.

-b
: Mit dem Schalter `-b` werden die Daten in binärer Form entsprechend einer vorgegebenen C-Struktur ausgegeben. Dieser Schalter kann eingesetzt werden für Programme, die die Ausgabe von `u62_sta` weiter verarbeiten wollen. Die C-Struktur ist definiert im Include-File `/opt/lib/utmlu62/status.h` (UNIX-Systeme)
  bzw.
  `Programme\utmlu62\status.h` (Windows-Systeme).

`u62_sta` gibt zwei Überschriftszeilen und pro openUTM-LU62-Instanz eine Zeile aus. Die Ausgabe sieht etwa wie folgt aus:

```
LLU name RLU name local OSI addr     remote OSI addr    XLN atot absy stot sbsy
================================================================================
TO2CGEI4 CICST6   SMP22804.utmlu62   SMP22800.utmlu62   ok    4    1    4    1
```

Mit dem Schalter `-c` erzeugt `u62_sta` z.B. folgende Ausgabe:

```
LLU name   RLU name       Number of CPSVCMG sessions
=====================================================
DEC1NO5C   P391.P391SSP    2
```

Bedeutung der Ausgabefelder:

LLU name Name der lokalen LU, wie er in der Generierungsdatei mit LOC-LU-ALIAS angegeben wurde.

RLU name Name der Partner-LU, wie er in der Generierungsdatei mit REM-LU-ALIAS angegeben wurde.

local OSI addr

TNSX-Name der lokalen Application Entity, wie er in der Generierungsdatei mit LOC-AE angegeben wurde.

Wird kein TNSX verwendet:
Lokaler T-Selektor ohne Formatangabe mit dem lokalen Port in Klammern.
Beispiel: `LOC_TSEL(23766)`

remote OSI addr

TNSX-Name der Partner-Application-Entity, wie er in der Generierungsdatei mit REM-AE angegeben wurde.

Wird kein TNSX verwendet:
T-Selektor der Partner-Anwendung ohne Formatangabe mit dem lokalen Port in Klammern. Dahinter folgt der Host-Name und, falls noch Platz ist, der TCP-Port.

Beispiel:
```
REM_TSEL(host:102)
REM_TSEL(host)
```

Ist der Host-Name zu lang, wird er abgeschnitten und mit ... darauf hingewiesen:
```
REM_TSEL(hostn...)
```

XLN Exchange-Log-Name-Zustand.

Wenn Sync-Level 2 vereinbart ist, muss nach Neustart der openUTM-LU62-Instanz oder nach einem Verbindungsabbruch ein Exchange-Log-Name zwischen openUTM-LU62 und der LU6.2-Anwendung durchgeführt werden. Erst wenn das Exchange-Log-Name erfolgreich beendet wurde, können Transaktionen gestartet werden. openUTM-LU62 führt diesen Exchange-Log-Name selbständig aus. Bei Fehlern kann jedoch ein manueller Eingriff notwendig sein.

Folgende Zustände sind möglich:

- Kein Exchange-Log-Name notwendig, da Sync-Level 2 nicht verwendet wird.

run Der Exchange-Log-Name läuft gerade.

wai Der Exchange-Log-Name steht noch aus. Folgende Ursachen sind möglich:

- Es besteht keine Verbindung zur LU6.2-Anwendung.
- Der vorangegangene Exchange-Log-Name wurde mit einem Kommunikationsfehler abgebrochen.
- Die Konfigurierungen von openUTM-LU62 und der LU6.2-Basis-Software (TRANSIT-SERVER, SNAP-IX bzw. IBM Communications Server) sind inkompatibel, d.h. bei laufendem openUTM-LU62 wurde die Konfigurierung der LU6.2-Basis-Software geändert.

ok Der Exchange-Log-Name wurde erfolgreich durchgeführt.

err Schwerwiegender Fehler beim Exchange-Log-Name.

rst (Restart) Wird als XLN-Status angezeigt, solange das Recovery zur UTM-Anwendung noch nicht abgeschlossen ist. Der Austausch der Log-Namen mit dem LU6.2-Partner konnte erfolgreich durchgeführt werden.

Dieser Zustand hält nach dem Start der Instanz nur kurze Zeit an, kann allerdings auch längere Zeit bestehen bleiben, wenn der LU6.2-Partner für das Recovery der Transaktion verantwortlich ist.

Wenn `err` oder `rst` angezeigt wird oder `wai` trotz bestehender Verbindung, so stehen weitere Informationen in der Datei

`/opt/lib/utmlu62/PROT/prot.`*luname* (UNIX-Systeme)

bzw.

`Programme\utmlu62\PROT\prot.`*luname*`.txt` (Windows-Systeme).

atot (Associations total) Anzahl der aufgebauten parallelen Verbindungen zwischen der openUTM-LU62-Instanz und der UTM-Anwendung.

absy (Associations busy) Anzahl der belegten parallelen Verbindungen zur UTM-Anwendung. Eine parallele Verbindung ist belegt, wenn gerade eine Programm-Programm-Kommunikation über diese Verbindung läuft.

stot (Sessions total) Anzahl der aufgebauten LU6.2-Sessions zwischen der openUTM-LU62-Instanz und dem LU6.2-Partner. Soll eine Kommunikation zum LU6.2-Partner möglich sein, so müssen mindestens 3 Sessions aufgebaut sein. 2 Sessions werden immer zur internen Verwaltung benötigt.

sbsy (Sessions busy)Anzahl der belegten Sessions. Eine Session ist belegt, wenn gerade eine Conversation (eine Programm-Programm-Kommunikation) über diese Session läuft.

Weitere Zustandsinformationen, insbesondere in Fehlerfällen befinden sich in der Datei

`/opt/lib/utmlu62/PROT/prot.`*luname* (UNIX-Systeme)
bzw.
`Programme\utmlu62\PROT\prot.`*luname*`.txt` (Windows-Systeme).

3.3.5 Verbindungen aufbauen

Mit dem Kommando `u62_adm` können Sie auch Verbindungen zur UTM-Anwendung oder zur LU6.2-Anwendung aufbauen.

Aufrufsyntax:

```
/opt/lib/utmlu62/u62_adm
    { -co | -cs }
    [ -l loc_lu_alias_name ]
```

Beschreibung:

-co (Connect OSI-TP) Es wird eine parallele Verbindung über OSI-TP zur UTM-Anwendung aufgebaut.

-cs (Connect SNA) Es wird eine SNA-LU6.2-Session zur LU6.2-Anwendung aufgebaut.

-l *loc_lu_alias_name*
Wenn der Schalter -l nicht angegeben ist, wird von jeder openUTM-LU62-Instanz eine Verbindung aufgebaut. Ist der Schalter -l angegeben, so wird nur von der zu dieser LU gehörenden Instanz eine Verbindung aufgebaut.

Ob der Verbindungsaufbau erfolgreich war, müssen Sie mit einem nachfolgenden `u62_sta` überprüfen.

3.3.6 Verbindungen abbauen

Mit dem Kommando `u62_adm` können Sie auch Verbindungen zur UTM-Anwendung oder zur LU6.2-Anwendung abbauen.

Syntax:

```
/opt/lib/utmlu62/u62_adm
    { -ao | -as | -do | -ds }
    [ -l loc_lu_alias_name ]
```

Beschreibung:

-ao (Abort OSI-TP) Es werden alle bestehenden parallelen Verbindungen zur UTM-Anwendung abgebaut, unabhängig davon, ob sie belegt sind oder nicht.

-as (Abort SNA) Es werden alle bestehenden SNA-Sessions zur LU6.2-Anwendung abgebaut, unabhängig davon, ob sie durch eine Conversation (Programm-Programm-Kommunikation) belegt sind oder nicht.

-do (Disconnect OSI-TP) Es werden alle bestehenden parallelen Verbindungen zur UTM-Anwendung abgebaut, die nicht belegt sind.

-ds (Disconnect SNA) Es werden alle bestehenden SNA-Sessions zur LU6.2-Anwendung abgebaut. Bei durch Conversations belegten Sessions wird der Abbau bis zum Conversation-Ende verzögert. Freie Sessions werden sofort abgebaut.

-l *loc_lu_alias_name*
Wenn der Schalter -l nicht angegeben ist, werden von jeder openUTM-LU62-Instanz die Verbindungen abgebaut. Ist der Schalter -l angegeben, so werden nur von der zu dieser LU gehörenden Instanz die Verbindungen abgebaut.

Ob der Verbindungsabbau erfolgreich war, müssen Sie mit einem nachfolgenden u62_sta überprüfen.

3.3.7 Trace ein- und ausschalten

Traceinformationen sind ein Hilfsmittel für die Fehlerdiagnose in openUTM-LU62. Bei eingeschaltetem Trace schreibt das Programm `u62_tp` laufend interne Informationen in spezielle Trace-Dateien.

Aufrufsyntax:

```
/opt/lib/utmlu62/u62_adm -ton[,trace_optionen ] [ -l loc_lu_alias_name ]
/opt/lib/utmlu62/u62_adm -tof[,trace_optionen ] [ -l loc_lu_alias_name ]
/opt/lib/utmlu62/u62_adm -tfl[,trace_optionen ] [ -l loc_lu_alias_name ]
```

Beschreibung:

-ton (Trace On) Der Trace wird eingeschaltet.

-tof (Trace Off) Der Trace wird ausgeschaltet.

-tfl (Trace Flush) Der Trace wird weitergeschaltet, d.h. die aktuelle Trace-Datei wird geschlossen und eine neue angelegt.

trace_optionen Steuert die Art und Ausführlichkeit des Trace. Folgende *trace_optionen* sind möglich. Es können auch mehrere Trace-Optionen, durch Komma getrennt, angegeben werden.

- in — Der Instanz-Trace wird ein-, aus- bzw. weitergeschaltet. Wird diese Trace-Option bei -ton angegeben, so wird der Instanz-Trace mit dem ausführlichen Level (in=2) eingeschaltet.
- xap — Der XAP-TP-Trace wird ein-, aus- bzw. weitergeschaltet.
- in=1 — (Nur bei -ton) Der Instanz-Trace wird mit dem niedrigsten Level eingeschaltet. Es werden nur die für einen Protokoll-Trace erforderlichen Trace-Einträge geschrieben. Zum Protokoll-Trace siehe Seite 66. Dieses Trace-Level ist im allgemeinen für eine erste Fehlerdiagnose ausreichend.
- in=2 — (Nur bei -ton) Der Instanz-Trace wird mit dem ausführlichen Level eingeschaltet.
- in=3 — (Nur bei -ton) Der Instanz-Trace wird mit einer zusätzlichen Zeitmessung für alle Aktivitäten gestartet. Dieses Level ist zur Diagnose von Performance-Problemen gedacht.

Wird *trace_optionen* weggelassen, so werden die Optionen `in,xap` angenommen.

-l *loc_lu_alias_name*

Wenn der Schalter -l nicht angegeben ist, wirkt das Kommando auf alle openUTM-LU62-Instanzen. Ist der Schalter -l angegeben, so wirkt das Kommando nur auf die zu dieser LU gehörenden Instanz.

Beispiel: Einschalten des ausführlichem Trace mit Zeitmessung auf UNIX-Systemen:

```
/opt/lib/utmlu62/u62_adm -ton,in=3,xap
```

Wenn Sie den Trace von Anfang an laufen lassen wollen, können Sie dies durch Option -t beim Kommando `u62_start` erreichen.

Die Traces werden in folgenden Dateien abgelegt:

- Instanz-Trace-Datei: `/opt/lib/utmlu62/PROT/inlog.`*loc_lu_alias.suff*
- XAP-TP-Trace-Datei: `/opt/lib/utmlu62/PROT/xaplog.`*loc_lu_alias.suff1.suff2*

Auf Windows-Systemen liegen die Dateien entsprechend unter `Programme\utmlu62\PROT`.

Das Suffix *suff* nimmt zu Beginn den Wert 0 an. Bei Erreichen einer Dateigröße von etwa einem halben Megabyte oder bei Eingabe des oben beschriebenen Flush-Kommandos wird die Trace-Datei geschlossen und eine neue Trace-Datei mit einem um 1 erhöhten Suffix angelegt. Nach Schließen der Datei mit *suff*=9 beginnt *suff* wieder bei 0.

Beim XAP-TP-Trace wird *suff1* nur durch Eingabe eines Flush-Kommandos hochgezählt, *suff2* bei Erreichen der maximalen Dateigröße.

Der Wert 10 für das Rundschreiben der Traces kann auf UNIX-Systemen geändert werden durch Setzen der Shell-Variablen U62_TRC_FILES vor dem Start von openUTM-LU62. Wollen Sie also z.B. 20 Trace-Dateien anlegen, so sollten Sie folgendes Kommando eingeben:

```
U62_TRC_FILES=20
export U62_TRC_FILES
/opt/lib/utmlu62/u62_start -t
```

Mit der Shell-Variablen U62_TRC_LEN kann man die Länge der Benutzerdaten im Trace beschränken.

Mit der Shell-Variablen U62_TRC_COMPRESS kann man eine Komprimierung der Trace-Dateien auf UNIX-Systemen erreichen. Hierzu muss man U62_TRC_COMPRESS auf den Wert „compress“ oder „gzip“ setzen. Zu beachten ist dabei, dass die Programme „compress“ bzw. „gzip“ über die PATH-Variable zu finden sind.

Auf Windows-Systemen sind anstelle der Shell-Variablen die gleichnamigen Variablen über den Registrierungseditor des Betriebssystems zu setzen. Diese Variablen befinden sich im Verzeichnis `HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\openUTM-LU62\CurrentVersion`.

3.3.8 Trace auswerten

Zur Auswertung des Instanz-Trace steht das folgende Kommando zur Verfügung:

```
/opt/lib/utmlu62/u62_adm -f [ -p] -o outfile tracefile [ tracefile2 ... ]
```

Beschreibung

-f tracefile — (File) Es werden die Instanz-Trace-Dateien mit dem Namen *tracefile* (bzw. *tracefile2* usw.) in lesbare Form aufbereitet und ausgegeben.

-o outfile — Name der Ausgabedatei.

-p — Es wird zusätzlich ein Protokoll-Trace in die Datei *outfile*`.flow` ausgegeben. `u62_adm` ruft zu diesem Zweck die awk-Prozedur `u62_flow` auf. Dieser Protokoll-Trace wird im folgenden Abschnitt genauer beschrieben.

Beispiel:

Sind im Dateiverzeichnis des UNIX-Systems `/opt/lib/utmlu62/PROT` die Trace-Dateien `inlog.TO2CGEI4.0` und `inlog.TO2CGEI4.1` vorhanden, so können Sie diese mit folgenden Kommandos auswerten:

```
cd /opt/lib/utmlu62/PROT
../u62_adm -fpo outfile inlog.TO2CGEI4.0 inlog.TO2CGEI4.1
```

Es werden dadurch die Dateien `outfile` und `outfile.flow` im Dateiverzeichnis `/opt/lib/utmlu62/PROT` angelegt.

Zur Auswertung des XAP-TP-Trace steht das folgende Kommando zur Verfügung:

```
/opt/lib/oss/step tracefile [ tracefile2 ... ]
```

`step` bereitet die XAP-TP-Traces auf und gibt sie in lesbarer Form nach `stdout` aus.

Auf Windows-Systemen steht `step` unter `Programme\utmlu62`.

3.3.9 Dump erzeugen

In manchen Fehlersituationen kann es erforderlich sein, als Hilfsmittel für die Fehlerdiagnose Zustandsinformationen von openUTM-LU62 in Form eines Dumps zu sichern. Hierzu dient folgendes Kommando:

```
/opt/lib/utmlu62/u62_adm -b in, xap
```

Dadurch werden folgende Dateien angelegt:

`/opt/lib/utmlu62/PROT/in.dump.`*loc_lu_alias*`.1`
`/opt/lib/utmlu62/PROT/xap.dump.`*loc_lu_alias*`.1.DIAGOO`

Auf Windows-Systemen liegen die Dateien entsprechend unter `Programme\utmlu62\PROT`.

Mit jedem Aufruf von `u62_adm -b` wird der Zähler, der im obigen Beispiel auf `1` steht, um `1` erhöht.

3.3.10 Protokoll-Trace

Im Protokoll-Trace finden Sie eine verkürzte Darstellung des Protokollflusses auf LU6.2- und auf OSI-TP-Seite. Da openUTM-LU62 für die LU6.2-Kommunikation die Programmschnittstelle APPC und für die OSI-TP-Kommunikation die Programmschnittstelle XAP-TP benutzt, sind in diesem Trace auch einige Eigenschaften dieser beiden Schnittstellen sichtbar.

Auf der LU6.2-Seite werden bei jeder Nachricht der Name des APPC-Aufrufs, die von der LU6.2-Basis-Software (TRANSIT-SERVER, SNAP-IX bzw. IBM Communications Server) vergebene TP-ID, die Verarbeitungsrichtung und eventuell weitere Parameter im Trace ausgegeben. Die Verarbeitungsrichtung ist mit einem Pfeil gekennzeichnet. Ein Pfeil nach links kennzeichnet eine von openUTM-LU62 ausgehende Nachricht, ein Pfeil nach rechts eine bei openUTM-LU62 eingehende Nachricht. Beim Start einer openUTM-LU62-Instanz und nach einem Verbindungsverlust werden zunächst Verwaltungsdaten zwischen openUTM-LU62 und dem LU6.2-Partner ausgetauscht. Hierzu wird der Transaktionscode X'06F2' verwendet. Diese Protokollflüsse sind mit einem einfachen Pfeil (--->) gekennzeichnet. Protokollflüsse, die von den Anwendungsprogrammen kommen, sind mit einem Doppelpfeil (===>)gekennzeichnet.

Auf der OSI-TP-Seite werden bei jeder Nachricht der Name des XAP-TP-Aufrufs, der vom XAP-TP-Provider vergebene Instanz-Deskriptor (fd), die Verarbeitungsrichtung und eventuelle weitere Parameter im Trace ausgegeben. Die Verarbeitungsrichtung ist mit einem Pfeil gekennzeichnet. Ein Pfeil nach rechts kennzeichnet eine von openUTM-LU62 ausgehende Nachricht, ein Pfeil nach links eine bei openUTM-LU62 eingehende Nachricht. Beim Start einer openUTM-LU62-Instanz und während der Commit-Phase wird an der XAP-TP-Schnittstelle eine sogenannte Kontrollinstanz verwendet. Diese Protokollflüsse sind mit einem einfachen Pfeil (--->) gekennzeichnet. Alle anderen Protokollflüsse sind mit einem Doppelpfeil (===>)gekennzeichnet.

Um die Zuordnung zwischen einer LU6.2-Conversation und einer parallelen Verbindung über XAP-TP zu erleichtern, enthält jede Nachricht eine Korrelationsnummer (corr). Protokollflüsse, die keiner Conversation zugeordnet sind, haben die Korrelationsnummer 0. Außerdem wird zu jeder Nachricht jeweils die Uhrzeit und die Nummer der entsprechenden Zeile in der urprünglichen Ausgabedatei ausgegeben.

Benutzerdaten sind im Protokoll-Trace nicht enthalten. Diese sind nur in der ursprünglichen Ausgabedatei zu finden.

Im folgenden Beispiel eines Protokoll-Trace sind zusätzliche Kommentare enthalten. Diese Kommentare beziehen sich teilweise auf die im folgenden Kapitel verwendeten Begriffe, z.B. die Namen von CICS-Kommandos.

Das Beispiel enthält folgenden Ablauf:

1. Exchange Logname
2. Aufruf einer CICS-Transaktion durch ein UTM-Anwendungsprogramm
3. Aufruf einer UTM-Transaktion durch ein CICS-Anwendungsprogramm

Exchange Logname

```
             CICS                 openUTM-LU62            openUTM
=============================================================================
                                  TP_RESTART_COMPLETE_IND (fd = 0)
 10:46:22, line 506, corr 0     <---------------------------------------
```

openUTM-LU62 wurde gestartet. Die Anmeldung an den XAP-TP-Provider war erfolgreich.

```
 RECEIVE_ALLOCATE (tpid = 00067478)
-----------------------------------------> 10:46:22, line 1138, corr 0
 tp_name    = X“06F2“,
 sync_level = CONFIRM_SYNC_LEVEL,
 conv_type  = BASIC_CONVERSATION

 RECEIVE_IMMEDIATE (tpid = 00067478)
-----------------------------------------> 10:46:23, line 1264, corr 0
 what_rcvd = DATA_COMPLETE, len = 33
 GDS_ID    = Exchange Logname (Req, warm)
```

CICS hat eine Conversation mit dem TP-Namen X’06F2’ aufgebaut und einen Exchange-Log-Name gesendet.

```
 RECEIVE_AND_POST (tpid = 00067478)
-----------------------------------------> 10:46:23, line 1338, corr 0
 what_rcvd = SEND

 SEND_DATA (tpid = 00067478)
<-----------------------------------------  10:46:24, line 1444, corr 0
 send_type = SEND_DATA_DEALLOC_SYNC_LEVEL, len = 58,
 GDS_ID    = Exchange Logname (Rpl+, cold)
```

openUTM-LU62 beantwortet den Exchange-Log-Name. Erst ab diesem Zeitpunkt ist eine transaktionsgesicherte Kommunikation zwischen openUTM-LU62 und CICS möglich.

```
 GET_LU_STATUS (tpid = 00067478)
<-----------------------------------------  10:46:24, line 1487, corr 0
 zero_sess = YES,
 active_sess = 19
```

Aufruf einer CICS-Transaktion durch ein UTM-Anwendungsprogramm

```
                                  TP_BEGIN_DIALOGUE_IND (fd = 10)
 10:46:40, line 2380, corr 1    <=======================================
```

Ein UTM-Anwendungsprogramm wurde gestartet und will eine Verbindung zu einem CICS-Anwendungsprogramm aufbauen.

```
 GET_LU_STATUS (tpid = 00722838)
<---------------------------------------  10:46:41, line 3069, corr 0
 zero_sess = NO,
 active_sess = 29

 MC_ALLOCATE (tpid = 00984982)
<=======================================  10:46:41, line 3257, corr 1
 tp_name    = UPS1,
 sync_level = SYNCPT,
 plu_alias  = CICST6,
 security   = NONE
```

openUTM-LU62 baut eine Conversation zum gewünschten CICS-Anwendungsprogramm mit dem Transaktionscode UPS1 auf. An sync_level = SYNCPT sieht man, dass es sich um eine transaktionsgesicherte Verbindung handelt. Es werden keine Benutzerkennung und kein Passwort gesendet.

```
                                   TP_DATA_IND (fd = 10)
10:46:42, line 3353, corr 1      <=======================================
                                  udata_len = 19
```

Die UTM-Anwendung sendet 19 byte Daten. In diesen 19 byte ist noch ein OSI-TP-Header enthalten.

```
 MC_SEND_DATA (tpid = 00984982)
<=======================================  10:46:42, line 3448, corr 1
 send_type = NONE,
 data_type = APPLICATION, len = 2
```

openUTM-LU62 sendet die Daten an das CICS-Anwendungsprogramm weiter. Hier sieht man, dass es sich nur um 2 byte Benutzerdaten handelt. Der OSI-TP-Header war also 17 byte lang. (Die Länge des OSI-TP-Header kann unterschiedlich sein.)

```
                                  TP_DEFERRED_END_DIALOGUE_IND (fd = 10)
 10:46:42, line 3488, corr 1    <=======================================

                                  TP_PREPARE_IND (fd = 10)
 10:46:42, line 3557, corr 1    <=======================================
```

Das UTM-Anwendungsprogramm hat nach MPUT NE ein CTRL PE und ein PEND KP abgesetzt. Der CTRL PE ist als TP_DEFERRED_END_DIALOGUE_IND und TP_PREPARE_IND zu sehen.

```
 MC_SEND_DATA (tpid = 00984982)
<========================================  10:46:42, line 3757, corr 1
 send_type = SEND_DATA_P_TO_R_FLUSH,
 data_type = PS_HEADER (Prepare)
```

openUTM-LU62 sendet ein Prepare an das CICS-Anwendungsprogramm und fordert es somit zum Einleiten des Transaktionsendes auf.

```
 MC_RECEIVE_IMMEDIATE (tpid = 00984982)
========================================>  10:46:45, line 3835, corr 1
 what_rcvd = PS_HEADER_COMPLETE (Request Commit)

 MC_RECEIVE_AND_POST (tpid = 00984982)
========================================>  10:46:45, line 3902, corr 1
 what_rcvd = SEND
```

Das CICS-Anwendungsprogramm hat das Transaktionsende mit dem CICS-Kommando SYNCPOINT bestätigt. CICS sendet deshalb ein Request Commit und gibt das Senderecht ab.

```
                                TP_PREPARE_ALL_REQ (fd = 0)
 10:46:45, line 3932, corr 1   ======================================>

                                TP_READY_ALL_IND (fd = 0)
 10:46:45, line 4009, corr 1   <======================================
                                udata_len = 99

                                TP_COMMIT_REQ (fd = 0)
 10:46:45, line 4261, corr 1  ======================================>
```

openUTM-LU62 sendet den Request Commit als TP_COMMIT_REQUEST an die UTM-Anwendung weiter. Dies beendet die erste Phase des 2-Phase-Commit-Protokolls. Die vorausgehenden XAP-TP-Aufrufe TP_PREPARE_ALL_REQ und TP_READY_ALL_IND sind nur lokaler Natur.

```
                                TP_COMMIT_IND (fd = 0)
 10:46:47, line 4337, corr 1   <======================================

                                TP_COMMIT_IND (fd = 10)
 10:46:47, line 4407, corr 1  <======================================
```

Die UTM-Anwendung startet nun die zweite Phase des 2-Phase-Commit-Protokolls. openUTM-LU62 erhält zweimal ein TP_COMMIT_IND. openUTM hätte die Transaktion zu diesem Zeitpunkt auch noch zurücksetzen können. Nach dem TP_COMMIT_IND ist dies jedoch nicht mehr möglich.

```
 MC_SEND_DATA (tpid = 00984982)
<========================================  10:46:47, line 4517, corr 1
 send_type = SEND_DATA_P_TO_R_FLUSH,
 data_type = PS_HEADER (Committed)
```

Aufgrund des erhaltenen TP_COMMIT_IND sendet openUTM-LU62 ein Committed an CICS.

```
 MC_RECEIVE_IMMEDIATE (tpid = 00984982)
========================================>  10:46:47, line 4596, corr 1
 what_rcvd = PS_HEADER_COMPLETE (Forget)

 MC_RECEIVE_AND_POST (tpid = 00984982)
========================================>  10:46:47, line 4695, corr 1
 primary_rc = DEALLOC_NORMAL
```

CICS antwortet mit einem Forget. Der anschließende DEALLOC_NORMAL zeigt an, dass die Conversation nach dem Transaktionsende wie vom UTM-Anwendungsprogramm beim CTRL PE gewünscht beendet wurde.

```
                                  TP_DONE_REQ (fd = 0)
 10:46:47, line 4802, corr 1    ======================================>

                                  TP_COMMIT_COMPLETE_IND (fd = 0)
 10:46:47, line 4877, corr 1    <======================================

                                  TP_COMMIT_COMPLETE_IND (fd = 10)
 10:46:47, line 4949, corr 1   <======================================
```

openUTM-LU62 sendet den erhaltenen Forget als TP_DONE_REQ an die UTM-Anwendung. Nach Erhalt des zweiten TP_COMMIT_COMPLETE_IND sind die Transaktion und der OSI-TP-Dialog beendet.

Aufruf einer UTM-Transaktion durch ein CICS-Anwendungsprogramm

```
 RECEIVE_ALLOCATE (tpid = 00133014)
=======================================>  10:47:45, line 5489, corr 2
 tp_name    = UCS1,
 sync_level = SYNCPT,
 conv_type  = MAPPED_CONVERSATION
```

Ein CICS-Anwendungsprogramm wurde gestartet und will eine Verbindung zu einem UTM-Anwendungsprogramm mit dem Transaktionscode UCS1 aufbauen.

```
                                 APM_ALLOCATE_REQ (fd = 1)
 10:47:45, line 5615, corr 2    ======================================>

                                 APM_ALLOCATE_CNF+ (fd = 1)
 10:47:45, line 5687, corr 2    <======================================

                                 TP_BEGIN_DIALOGUE_REQ (fd = 1)
 10:47:45, line 5848, corr 2    ======================================>
                                udata_len = 28
```

openUTM-LU62 baut eine Verbindung zum gewünschten UTM-Anwendungsprogramm auf.

```
 MC_RECEIVE_IMMEDIATE (tpid = 00133014)
=======================================>  10:47:45, line 5975, corr 2
 what_rcvd = DATA_COMPLETE, len = 2
```

Das CICS-Anwendungsprogramm sendet 2 byte Daten.

```
                                 TP_DATA_REQ (fd = 1)
 10:47:45, line 6005, corr 2    ======================================>
                                udata_len = 23
```

openUTM-LU62 sendet die Daten an das UTM-Anwendungsprogramm weiter. Hier sieht man, dass openUTM-LU62 einen OSI-TP-Header von 21 byte voransetzt.

```
 MC_RECEIVE_IMMEDIATE (tpid = 00133014)
=======================================>  10:47:45, line 6135, corr 2
 what_rcvd = PS_HEADER_COMPLETE (Request Commit)

 MC_RECEIVE_AND_POST (tpid = 00133014)
=======================================>  10:47:45, line 6197, corr 2
 what_rcvd = SEND
```

Das CICS-Anwendungsprogramm hat mit dem SYNCPOINT-Kommando das UTM-Anwendungsprogramm aufgefordert, die Transaktion zu beenden. CICS sendet ein Request Commit und gibt das Senderecht ab, verwendet also in diesem Fall kein echtes 2-Phase-Commit-Protokoll, sondern ein vereinfachtes Protokoll, bei dem es später keine Möglichkeit mehr hat, die Transaktion zurückzusetzen. CICS verwendet dieses vereinfachte Protokoll immer dann, wenn das CICS-Anwendungsprogramm nur einen Auftragnehmer hat und vor dem SYNCPOINT-Kommando kein ISSUE PREPARE absetzt.

```
                                  TP_DEFERRED_END_DIALOGUE_REQ (fd = 1)
10:47:45, line 6227, corr 2    ======================================>
```

Was im Protokoll-Trace bei Erhalt des Request Commit nicht erkennbar war, wird hier deutlich: Das CICS-Anwendungsprogramm hat die Nachricht vor dem SYNCPOINT mit SEND LAST gesendet und damit das Ende der Conversation eingeleitet. openUTM-LU62 sendet deshalb ein TP_DEFERRED_END_DIALOGUE_REQ an die UTM-Anwendung.

```
                                   TP_PREPARE_ALL_REQ (fd = 0)
10:47:45, line 6274, corr 0     =====================================>
```

Danach sendet openUTM-LU62 die Aufforderung zum Transaktionsende an die UTM-Anwendung.

```
                                   TP_READY_ALL_IND (fd = 0)
10:47:47, line 6350, corr 2      <=====================================
                                  udata_len = 97
```

Das UTM-Anwendungsprogramm hat das Transaktionsende durch Absetzen von PEND FI bestätigt. Bei openUTM-LU62 trifft deshalb ein TP_READY_ALL_IND ein.

```
                                   TP_COMMIT_REQ (fd = 0)
10:47:47, line 6605, corr 2     =====================================>

                                   TP_COMMIT_IND (fd = 0)
10:47:47, line 6682, corr 2     <=====================================

                                   TP_COMMIT_IND (fd = 1)
10:47:47, line 6752, corr 2     <=====================================

                                   TP_DONE_REQ (fd = 0)
10:47:47, line 6827, corr 2     =====================================>

                                   TP_COMMIT_COMPLETE_IND (fd = 0)
10:47:48, line 6903, corr 2     <=====================================

                                   TP_COMMIT_COMPLETE_IND (fd = 1)
10:47:48, line 6975, corr 2    <=====================================
```

Alle Aufrufe von TP_COMMIT_REQ bis zum TP_COMMIT_COMPLETE_IND dienen dazu, das Transaktionsende protokollkonform durchzuführen.

```
 MC_SEND_DATA (tpid = 00133014)
<=======================================  10:47:48, line 7244, corr 2
 send_type = NONE,
 data_type = PS_HEADER (Committed)

 MC_DEALLOCATE (tpid = 00133014)
<=======================================  10:47:48, line 7286, corr 2
 dealloc_type = FLUSH
```

Nach erfolgtem Transaktionsende auf OSI-TP-Seite sendet openUTM-LU62 ein Committed an CICS und beendet anschließend die Conversation mit MC_DEALLOCATE. Damit sind Transaktion und Conversation auch auf LU6.2-Seite beendet.

4 openUTM-CICS-Kopplung über LU6.2

In diesem Kapitel finden Sie die Aspekte der openUTM-CICS-Kopplung über LU6.2 bezüglich Generierung und Programmierung.

4.1 openUTM-CICS-Kopplung generieren

4.1.1 Definitionen in CICS

In CICS sind folgende Definitionen erforderlich:

– Festlegung der APPLID

– Für jede Instanz von openUTM-LU62 eine Connection und eine Session-Definition

Der CICS-Systemverwalter hat mehrere Möglichkeiten zur Definition von Connections und Sessions, z.B. RDO (Resource Definition Online) mittels CEDA, DFHCSDUP oder Makros.

APPLID

In der System Initialization Table (SIT) muss der Name der CICS-Anwendung angegeben werden:

DFHSIT APPLID=cics_netname

1-8 Zeichen langer Name, mit dem der SNA-Netzname für die CICS-Anwendung festgelegt wird. Dieser Name muss mit den Definitionen in VTAM (VBUILD TYPE=APPL) abgestimmt sein.

DEFINE CONNECTION

Mit dieser Definition wird das ferne System, in diesem Fall die Stellvertreter-LU in openUTM-LU62 beschrieben.

```
DEFINE  CONNECTION   connection_name
        GROUP        group_name
        NETNAME      netname
        ACCESSMETHOD VTAM
        PROTOCOL     APPC
        SINGLESESS   NO
        DATASTREAM   USER
        RECORDFORMAT U
        AUTOCONNECT  { NO | YES | ALL }
        INSERVICE    YES
        ATTACHSEC    { LOCAL | IDENTIFY | VERIFY | MIXIDPE }
```

Die Operanden haben folgende Bedeutung:

CONNECTION connection_name
: 1-4 Zeichen langer Name zur Bezeichnung der UTM-Anwendung. Dieser Name existiert nur innerhalb von CICS. Er muss in den CICS-Programmen beim ALLOCATE im Parameter SYSID angegeben werden, wenn das CICS-Programm einen UTM-Transaktionscode aufrufen will.

GROUP group_name
: 1-8 Zeichen langer Name. Der CICS-Systemverwalter muss diesen Namen vergeben. Der Name kommt nur in den CICS-Definitionen vor.

NETNAME netname
: 1-8 Zeichen langer Name der Stellvertreter-LU. Dieser Name muss mit dem in VTAM definierten LU-Namen und dem in der LU6.2-Basis-Software definierten Namen übereinstimmen. Im Falle von TRANSIT als LU6.2-Basis-Software ist dies der zweite Teil des bei XLU NETNAME vergebenen Namens. Abhängig vom verwendeten SNA-Protokoll kann die Definition von LU-Namen in VTAM eventuell entfallen.

ACCESSMETHOD VTAM
: Zugriffsmethode VTAM.

PROTOCOL APPC
: Es wird eine LU6.2-Verbindung definiert.

SINGLESESS NO
: Es werden parallele Sessions zugelassen. openUTM-LU62 benötigt parallele Sessions.

DATASTREAM USER
: Die Benutzerdaten werden vollständig vom Anwendungsprogramm definiert.

RECORDFORMAT U

Die Benutzerdaten werden als Chain von RUs gesendet.

AUTOCONNECT { NO | YES | ALL }

Bei Angabe von YES oder ALL werden die Sessions beim Start von CICS aufgebaut. Bei NO unterbleibt dies.

INSERVICE YES

Die Connection ist verfügbar.

ATTACHSEC { LOCAL | IDENTIFY | VERIFY | MIXIDPE }

Bei Angabe von LOCAL (Standardwert) darf die UTM-Anwendung keine Benutzerkennung und kein Passwort beim APRO angeben (KCSECTYP=N).
Bei Angabe von IDENTIFY muss die UTM-Anwendung beim APRO KCSECTYP=S angeben.
Bei Angabe von VERIFY muss die UTM-Anwendung beim APRO KCSECTYP=P und eine Benutzerkennung und ein Passwort angeben.
Bei Angabe von MIXIDPE kann die UTM-Anwendung wählen zwischen KCSECTYP=S und KCSECTYP=P.

DEFINE SESSIONS

Mit dieser Definition werden die Eigenschaften der Verbindung zur Stellvertreter-LU in openUTM-LU62 beschrieben.

```
DEFINE  SESSIONS    session_name
        GROUP       group_name
        CONNECTION  connection_name
        MODENAME    modename
        PROTOCOL    APPC
        MAXIMUM     max1 , max2
        SENDSIZE    ru_size
        RECEIVESIZE ru_size
        AUTOCONNECT { NO | YES | ALL }
        INSERVICE   YES
        BUILDCHAIN  YES
        IOAREALEN   data_length
```

Die Operanden haben folgende Bedeutung:

SESSIONS session_name
: Name der Sessions-Definition. Dieser Name kommt nur in den CICS-Definitionen vor.

GROUP group_name
: 1-8 Zeichen langer Name. Der CICS-Systemverwalter muss diesen Namen vergeben. Der Name kommt nur in den CICS-Definitionen vor. Es sollte derselbe Name wie in der zugehörigen Connection-Definition gewählt werden.

CONNECTION connection_name
: Name der zugehörigen Connection-Definition.

MODENAME modename
: 1-8 Zeichen langer Modename für die Verbindung zwischen CICS und openUTM-LU62. Der Name muss auch in VTAM und der LU6.2-Basis-Software (TRANSIT-SERVER, SNAP-IX bzw. IBM Communications Server) und openUTM-LU62 als Modename definiert sein.

PROTOCOL APPC
: Es wird eine LU6.2-Verbindung definiert.

MAXIMUM max1 , max2
: Mit *max1* wird die Maximalzahl von Sessions zwischen den beiden LUs festgelegt. Der Wert muss zwischen 1 und 254 liegen. Bei Verwendung von TRANSIT-SERVER sollte er mit dem Parameter SESS-MAX in der zugehörigen XMODE-Anweisung der TRANSIT-Konfigurierung übereinstimmen. Bei Verwendung von SNAP-IX oder des IBM Communications Server sollte *max1* mit dem Session-Limit in der Mode-Definition des IBM Communications Server übereinstimmen.

Mit *max2* wird die Maximalzahl von Sessions festgelegt, bei denen die CICS-Seite Contention-Winner ist. Bei Verwendung von TRANSIT-SERVER sollte er mit dem Parameter SESS-LOS in der zugehörigen XMODE-Anweisung der TRANSIT-Konfigurierung übereinstimmen. Bei Verwendung von SNAP-IX oder des IBM Communications Server muss er nicht mit dessen Konfiguration abgestimmt werden.

Wenn die Maximalwerte von CICS und LU6.2-Basis-Software nicht übereinstimmen, einigen sich die Systeme auf den niedrigeren Wert.

SENDSIZE ru_size RECEIVESIZE ru_size

Damit wird die maximale RU(Request Unit)-Size für die LU6.2-Kommunikation zwischen CICS und openUTM-LU62 festgelegt.
Es sollte der Standardwert 4096 verwendet werden. Kleinere RUs führen zu höherer Netzbelastung. Die Werte für SENDSIZE und RECEIVESIZE sollten gleich gewählt werden.

Der Wert sollte dem Wert im MODE-Makro von VTAM entsprechen. Bei Verwendung von TRANSIT-SERVER sollte er mit dem dort konfigurierten Wert SRU-MAX und RRU-MAX übereinstimmen. Bei Verwendung von SNAP-IX oder des IBM Communications Server sollte er mit dem dort beim Mode-Namen definierten Wert für die maximale RU-Size übereinstimmen. Dabei ist zu beachten, dass die RU-Size in VTAM und TRANSIT-SERVER in anderer Form angegeben werden muss. Der Wert 4096 ist bei TRANSIT als 89 zu schreiben. Weitere Details finden Sie im TRANSIT-Handbuch.

AUTOCONNECT { NO | YES | ALL }

Bei Angabe von YES oder ALL werden die Sessions beim Start von CICS aufgebaut. Bei NO unterbleibt dies.

INSERVICE YES

Die Connection ist verfügbar.

BUILDCHAIN YES

Der Parameter muss immer auf den Standardwert YES gesetzt werden.

IOAREALEN data_length

Mit *value1* wird die Mindestgröße der Terminal-Input-Output-Area von CICS festgelegt. Dieser Wert sollte mindestens so groß gewählt werden wie die maximale Nachrichtenlänge der Anwendungsprogramme.

4.1.2 VTAM-Generierung

In den Fällen, bei denen die LUs in VTAM definiert werden, müssen die von openUTM-LU62 verwendeten Stellvertreter-LUs als independent LUs, also mit LOCADDR=0 generiert sein. Außerdem muss der Mode-Name in VTAM generiert werden. Siehe weiter unten das vollständige Generierungsbeispiel.

4.1.3 TRANSIT-Generierung für openUTM-CICS-Sessions

Bei Verwendung von TRANSIT-SERVER muss in TRANSIT zum einen die SNA-Verbindung zwischen dem UNIX-System und dem SNA-Host konfiguriert werden. Zum anderen muss dort auch die LU6.2-Verbindung zwischen CICS und openUTM-LU62 konfiguriert werden. Siehe weiter unten das vollständige Generierungsbeispiel.

4.1.4 SNAP-IX-Generierung für openUTM-CICS-Sessions

Bei Verwendung von SNAP-IX muss in diesem Produkt zum einen die SNA-Verbindung zwischen dem UNIX-System und dem SNA-Host konfiguriert werden. Zum anderen muss dort auch die LU6.2-Verbindung zwischen CICS und openUTM-LU62 konfiguriert werden. Siehe weiter unten das vollständige Generierungsbeispiel.

4.1.5 Generierung des IBM Communications Server für openUTM-CICS-Sessions

Bei Verwendung des IBM Communications Server muss in diesem Produkt zum einen die SNA-Verbindung zwischen dem UNIX-/Windows-System und dem SNA-Host konfiguriert werden. Zum anderen muss dort auch die LU6.2-Verbindung zwischen CICS und openUTM-LU62 konfiguriert werden. Siehe weiter unten das vollständige Generierungsbeispiel.

4.1.6 openUTM-LU62-Generierung

Die openUTM-LU62-Generierung ist ausführlich im vorhergehenden Kapitel beschrieben. Zum Abgleich der Generierungsparameter mit den anderen Generierungen siehe weiter unten das vollständige Generierungsbeispiel.

4.1.7 TNSX-Generierung

Auf dem Rechner, auf dem openUTM-LU62 läuft, müssen die TCP/IP-Adress-Informationen für die Kommunikation zur UTM-Anwendung entweder in den TNSX (Transport Name Server UNIX Systems) oder direkt in die openUTM-LU62 eingetragen werden. Die Adress-Informationen bestehen aus IP-Adressen, Port-Nummern und T-Selektoren. Siehe auch weiter unten das vollständige Generierungsbeispiel.

4.1.8 openUTM-Generierung

Aus Sicht von openUTM muss eine LU6.2-Kopplung genauso wie eine OSI-TP-Kopplung generiert werden. Sie sollten dabei allerdings berücksichtigen, dass pro Access-Point maximal 254 Verbindungen über openUTM-LU62 aufgebaut werden können. Dies bedeutet, dass bei einer LU6.2-Verbindung der Wert des ASSOCIATIONS-Parameters in der OSI-LPAP-Anweisung nicht größer als 254 sein sollte.

4.1.9 Definieren von CICS-Transaktionen

Lokale Transaktionen werden über RDO bei CICS wie folgt definiert:

```
DEFINE  TRANSACTION  transaction_code
        GROUP        group_name
        PROGRAM      program_name
        INDOUBT      WAIT
```

Die Operanden haben folgenden Bedeutung:

TRANSACTION transaction_code
: 1-4 Zeichen langer Transaktionscode für die CICS-Transaktion. Der Name muss im Feld KCRN beim UTM-Aufruf APRO angegeben werden.

GROUP group_name
: 1-8 Zeichen langer Name. Der CICS-Systemverwalter muss diesen Namen vergeben. Er kommt nur in den CICS-Definitionen vor.

PROGRAM program_name
: Name des CICS-Programms.

INDOUBT WAIT
: Diese Angabe ist notwendig, um nach Sessionfehlern eine ordnungsgemäße Synchronisation mit openUTM zu erreichen.

Ferne Transaktionen müssen in CICS nicht definiert werden. Der Transaktionscode wird wie bei openUTM in den CICS-Programmen angegeben. Im Gegensatz zur LU6.1-Schnittstelle bietet CICS bei LU6.2 auch die Möglichkeit, Transaktionscodes mit mehr als 4 Zeichen direkt im Programm anzugeben. Dies ist wichtig, da UTM-Transaktionscodes bis zu 8 Zeichen lang sein können.

4.1.10 Verwendung von Benutzerkennungen

Bei verteilter Verarbeitung ist es oft sinnvoll, dass der menschliche Endbenutzer nur an einer Stelle im Gesamtsystem Benutzerkennung und Passwort eingibt und damit seine Berechtigung für bestimmte Aktionen nachweist. Die Benutzerkennung wird dann beim Aufbau der Auftraggeber-Auftragnehmer-Beziehung dem anderen Partner mitgeteilt, so dass der Benutzer in den anderen Systemen ebenfalls nur bestimmte Berechtigungen hat. Das Passwort dagegen wird üblicherweise nicht über das Netz übertragen.

Damit ein derartiges Verfahren funktionieren kann, müssen die Benutzerkennungen unter allen beteiligten Systemen eindeutig sein. Die Systemverwalter der beteiligten Systeme müssen also zumindest teilweise die gleichen Benutzerkennungen eintragen. Bei openUTM werden Benutzerkennungen als USER in der KDCDEF-Generierung angegeben, bei CICS werden Benutzerkennungen üblicherweise mit dem Produkt RACF verwaltet.

Sollen zusammen mit den Aufträgen auch Benutzerkennungen übertragen werden, so muss bei openUTM-LU62 und bei openUTM der Application-Kontext UDTSEC angegeben werden.

Bei Verwendung des LU6.2-Protokolls müssen sich Anwendungsprogramme meist nicht um das Weitersenden der Benutzerkennung kümmern. UTM-Anwendungsprogramme können bei APRO durch Setzen von KCSECTYP=S die Benutzerkennung an das CICS-System weitersenden.

CICS-Programme senden immer die Benutzerkennung weiter. Wie oben erwähnt, kann sie aber nur dann an die openUTM-Anwendung übertragen werden, wenn UDTSEC als Application-Kontext generiert wurde. Ist UDTSEC nicht als Application-Kontext generiert und sendet das CICS-Programm eine Benutzerkennung, so baut openUTM-LU62 die LU6.2-Conversation ab.

Werden in openUTM-Umgebung andere Benutzerkennungen als in CICS-Umgebung verwendet, so kann das UTM-Anwendungsprogramm als Auftraggeber selbst Benutzerkennung und Passwort setzen. Hierfür muss der Parameter KCSECTYP=P bei APRO gesetzt werden. Die Benutzerkennung muss in KCUSERID, das Passwort in KCPSWORD eingetragen werden. Zu beachten ist dabei, dass KCUIDTYP und KCPWDTYP auf „P“ oder „T“ gesetzt werden. Der Typ „Octet String“ wird von openUTM-LU62 nicht unterstützt. Es ist zu beachten, dass beim APRO in diesem Fall alle nicht verwendeten Felder des 2. Parameterbereichs gelöscht werden müssen.

Von einem CICS-Programm kann dagegen immer nur die Benutzerkennung gesendet werden, mit dem es aufgerufen wurde. Ein CICS-Programm kann kein Passwort an eine UTM-Anwendung senden.

Werden von CICS aus mehrere UTM-Transaktionen gleichzeitig mit derselben User-Id aufgerufen, so muss in der UTM-Generierung in der USER-Anweisung der Wert RESTART=NO angegeben werden.

Außerdem muss die UTM-Anwendung so generiert sein, dass Benutzer zu einer Zeit mehrfach unter derselben Benutzerkennung angemeldet sein können (Generierungsanweisung SIGNON, Pararameter MULTI-SIGNON=YES).

4.1.11 Vollständiges Generierungsbeispiel

Im Folgenden wird ein komplettes Beispiel einer Kopplung von einer UTM-Anwendung auf einem UNIX-System mit CICS/ESA aufgelistet. Die Verbindung zwischen CICS und dem UNIX-System ist hierbei mit SNA über Ethernet realisiert. Generierungsparameter, die übereinstimmen müssen, sind am rechten Rand mit Zahlen (1) bis (43) markiert. Diese Parameter sind am Ende des Abschnitts noch genauer erklärt.

Das Beispiel enthält beide Richtungen der Kommunikation, den Aufruf eines CICS-Programms durch openUTM und umgekehrt.

CICS-Programm

Adressierung im Auftraggeber-Programm:

```
EXEC CICS ALLOCATE
    SYSID (IDI4)                                          (1)
EXEC CICS CONNECT PROCESS
    CONVID (id)
    PROCNAME (UCS1)                                       (2)
    PROCLENGTH (4)
    SYNCLEVEL (2)
```

CICS-Definitionen

System Initialization:

```
DFHSIT APPLID=A9CICST6                                          (3)
```

CEDA-Definitionen:

```
   CEDA  View Connection( IDI4 )
    Connection     : IDI4                                       (1)
    Group          : KNAS1IND
    DEscription    :
   CONNECTION IDENTIFIERS
    Netname        : TO2CGEI4                                   (4)
    INDsys         :
   REMOTE ATTRIBUTES
    REMOTESYSTem   :
    REMOTEName     :
    REMOTESYSNet   :
   CONNECTION PROPERTIES
    ACcessmethod   : Vtam
    PRotocol       : Appc
    Conntype       :
    SInglesess     : No
    DAtastream     : User
    RECordformat   : U
    Queuelimit     : No
    Maxqtime       : No
   OPERATIONAL PROPERTIES
    AUtoconnect    : No
    INService      : Yes
   SECURITY
    SEcurityname   :
    ATtachsec      : Local
    BINDPassword   :
    BINDSecurity   : No
    Usedfltuser    : No
   RECOVERY
    PSrecovery     : Sysdefault
```

```
CEDA  View Sessions( CGEI4    )
 Sessions       : CGEI4
 Group          : KNAS1IND
 DEscription    :
SESSION IDENTIFIERS
 Connection     : IDI4                                          (1)
 SESSName       :
 NETnameq       :
 MOdename       : MODDIS89                                      (5)
SESSION PROPERTIES
 Protocol       : Appc
 MAximum        : 015 , 015                                     (6)
 RECEIVEPfx     :
 RECEIVECount   :
 SENDPfx        :
 SENDCount      :
 SENDSize       : 04096                                         (7)
 RECEIVESize    : 04096                                         (7)
 SESSPriority   : 000
 Transaction    :
OPERATIONAL PROPERTIES
 Autoconnect    : Yes
 INservice      :
 Buildchain     : Yes
 USERArealen    : 000
 IOarealen      : 00000 , 00000
 RELreq         : No
 DIscreq        : No
 NEPclass       : 000
RECOVERY
 RECOVOption    : Sysdefault
 RECOVNotify    : None


CEDA  View PROGram( UTMCICS1 )
 PROGram        : UTMCICS1                                      (8)
 Group          : KNAS1IND
 DEscription    :
 Language       : CObol
 RELoad         : No
 RESident       : No
 USAge          : Normal
 USElpacopy     : No
 Status         : Enabled
 RSl            : 00
 Cedf           : Yes
 DAtalocation   : Below
 EXECKey        : User
```

```
CEDA  View TRANSaction( UPS1 )
 TRANSaction    : UPS1                                              (9)
 Group          : KNAS1IND
 DEscription    :
 PROGram        : UTMCICS1                                          (8)
 TWasize        : 00000
 PROFile        : DFHCICST
 PArtitionset   :
 STAtus         : Enabled
 PRIMedsize     : 00000
 TASKDATALoc    : Below
 TASKDATAKey    : User
 STOrageclear   : No
 RUnaway        : System
 SHutdown       : Disabled
 ISolate        : Yes
RECOVERY
 DTimout        : No
 INdoubt        : Wait
 RESTart        : No
 SPurge         : No
 TPUrge         : No
 DUmp           : Yes
 TRACe          : Yes
 COnfdata       : No
```

VTAM-Generierung:

Start Options:

```
SSCPNAME=M88,                                                    (10)
NETID=DESNI000                                                   (43)
```

Major Nodes:

```
P02CGE    PU    ADDR=C1,                                         (17)
                DISCNT=NO,
                DLOGMOD=SNX32702,
                IDBLK=017,                                       (11)
                IDNUM=2000E,                                     (12)
                ISTATUS=ACTIVE,
                MAXDATA=1033,                                    (13)
                MAXOUT=7,
                MAXPATH=2,
                MODETAB=MOD3270,
                PACING=0,
                PUTYPE=2,
                SSCPFM=USSSCS,
                USSTAB=USSSCS,
                VPACING=0
          PATH  DIALNO=000440002222000E,                         (14)
                GRPNAM=G56TRN00
T02CGEI4  LU    LOCADDR=0,                                        (4)
                USSTAB=ISTINCDT,     ACF/VTAM - USS-TABLE
                DLOGMOD=MODDIS89,                                (5)
                MODETAB=MODLU62,                                 (15)
                RESSCB=32,
                PACING=3,
                VPACING=2,
                SSCPFM=FSS
```

Application Definitions:

```
A0CICS   VBUILD TYPE=APPL
A9CICST6 APPL   EAS=160,
                ACBNAME=A9CICST6,
                PARSESS=YES,
                AUTH=(ACQ,BLOCK, PASS)
```

Logon Mode Table:

```
MODLU62  MODETAB                                                    (15)
MODDIS89 MODEENT LOGMODE=MODDIS89,                                   (5)
             FMPROF=X'13',
             TSPROF=X'07',
             PRIPROT=X'B0',
             SECPROT=X'B0',
             COMPROT=X'50B1',
             RUSIZES=X'8989',                                        (7)
             TYPE=X'00',
             PSERVIC=X'060200000000000000002CD0',
             PSNDPAC=X'04',
             SRCVPAC=X'04',
             SSNDPAC=X'04',
             COS=NORMCOS
MODEEND
```

TRANSIT-Generierung

```
 XLINK  LO2CGE,                                                     (16)
        TYP=LAN,
        XID=0172000E                                          (11),(12)
 XPU    PO2CGE,                                                     (17)
        TYP=PEER,
        LINK=LO2CGE,                                                (16)
        DMAC=400037450001,                                          (18)
        MAXDATA=1033                                                (13)
 XLU    TO2CGEI4,                                                   (19)
        TYP=6,
        SESS-CTR=IND,
        SESS-LMT=20,                                                 (6)
        PUCONNECT=APHSTART,
        NETNAME=DESNI000.TO2CGEI4,                            (4),(43)
        SEC-PAIR=NO_CHECK NO_PASS,
        PAIR_EXT=CICST6 MODDIS89 SYNCLEVEL ALREADY_VERIFIED   (5),(20)
 XRLU   CICST6,                                                     (20)
        NETNAME=DESNI000.A9CICST6,                            (3),(43)
        PU=PO2CGE                                                   (17)
 XMODE  MODDIS89,                                                    (5)
        SESS-MAX=15,                                                 (6)
        SESS-LOS=2,
        SESS-WIN=2,
        SESS-AUTO=2,
        SRU-MAX=89,                                                  (7)
        RRU-MAX=89                                                   (7)
```

openUTM-LU62-Generierung ohne Verwendung von TNSX

```
INSTANCE
       LOC-LU-ALIAS      = TO2CGEI4,                                (19)
       LOC-TSEL          = T'SMP22804',                             (36)
       LOC-LISTENER-PORT = 22804,                                   (37)
       LOC-APT           = (1,2,3),                                 (22)
       LOC-AEQ           = 1,                                       (23)
       REM-LU-ALIAS      = CICST6,                                  (20)
       MODENAME          = MODDIS89,                                 (5)
       LU62-CODE         = EBCDIC-273,
       REM-TSEL          = T'SMP22800',                             (29)
       REM-LISTENER-PORT = 22800,                                   (30)
       REM-NSEL          = localhost,                               (33)
       REM-APT           = (1,2,4),                                 (25)
       REM-AEQ           = 1,                                       (26)
       APPL-CONTEXT      = UDTSEC,                                  (27)
       ASSOCIATIONS      = 15,                                       (6)
       CONNECT           = 4,
       CONTWIN           = 5,                                       (28)
       OSITP-CODE        = ASCII
```

openUTM-LU62-Generierung bei Verwendung von TNSX

```
INSTANCE
       LOC-LU-ALIAS      = TO2CGEI4,                                (19)
       LOC-AE            = SMP22804,                                (21)
       LOC-APT           = (1,2,3),                                 (22)
       LOC-AEQ           = 1,                                       (23)
       REM-LU-ALIAS      = CICST6,                                  (20)
       MODENAME          = MODDIS89,                                 (5)
       LU62-CODE         = EBCDIC-273,
       REM-AE            = SMP22800.utmlu62,                        (24)
       REM-APT           = (1,2,4),                                 (25)
       REM-AEQ           = 1,                                       (26)
       APPL-CONTEXT      = UDTSEC,                                  (27)
       ASSOCIATIONS      = 15,                                       (6)
       CONNECT           = 4,
       CONTWIN           = 5,                                       (28)
       OSITP-CODE        = ASCII
```

TNSX-Generierung

```
SMP22800.utmlu62\                                                    (24)
      PSEL V''                                                       (31)
      SSEL V''                                                       (32)
      TA RFC1006 127.0.0.1 PORT 22800 T'SMP22800'        (29),(30),(33)
SMP22804\                                                            (21)
        PSEL V''                                                     (34)
        SSEL V''                                                     (35)
        TSEL RFC1006 T'SMP22804'                                     (36)
        TSEL LANINET A'22804'                                        (37)
```

openUTM-Generierung

```
UTMD APT = (1,2,4)                                                   (25)
ACCESS-POINT OSIREP8,                                          -     (40)
       P-SEL = *NONE,                                          -     (31)
       S-SEL = *NONE,                                          -     (32)
       T-SEL = SMP22800,                                       -     (30)
       T-PROT = RFC1006,                                       -
       TSEL-FORMAT = T,                                        -
       LISTENER-PORT = 22800,                                  -     (29)
       AEQ = 1                                                       (26)
OSI-CON SMP22804,                                              -
       P-SEL = *NONE,                                          -     (34)
       S-SEL = *NONE,                                          -     (35)
       T-SEL = C'SMP22804',                                    -     (36)
       N-SEL = C'local',                                       -     (38)
       LOCAL-ACCESS-POINT = OSIREP8,                           -     (40)
       T-PROT = RFC1006,                                       -
       LISTENER-PORT = 22804,                                  -     (37)
       TSEL-FORMAT = T,                                        -
       OSI-LPAP = ACICS04                                            (41)
OSI-LPAP ACICS04,                                              -     (41)
       APPLICATION-CONTEXT = UDTSEC,                           -     (27)
       APT = (1,2,3),                                          -     (22)
       AEQ = 1,                                                -     (23)
       ASS-NAMES = OSIC4,                                      -
       ASSOCIATIONS = 15,                                      -      (6)
       CONNECT = 2,                                            -
       CONTWIN = 10                                                  (28)
LTAC CICSUPS1,                                                 -     (42)
       LPAP = ACICS04,                                         -     (41)
       RTAC = UPS1                                                    (9)
TAC UCS1,                                                      -      (2)
       PROGRAM = ucs1lu62
```

UTM-Anwendungsprogramm

Adressierung im Auftraggeber-Programm:

```
KCOP=APRO
KCOM=DM
KCRN=CICSUPS1                                                    (42)
KCPI=>A
KCOF=C
KDCS()
```

Änderungen bei Verwendung von openUTM unter BS2000/OSD

Wird anstelle von openUTM unter UNIX-Systemen openUTM unter BS2000/OSD verwendet, so ändert sich das Generierungsbeispiel wie folgt:

openUTM-LU62-Generierung ohne Verwendung von TNSX

```
INSTANCE
       LOC-LU-ALIAS      = TO2CGEI4,                             (19)
       LOC-TSEL          = T'SMP22804',                          (36)
       LOC-LISTENER-PORT = 22804,                                (37)
       LOC-APT           = (1,2,3),                              (22)
       LOC-AEQ           = 1,                                    (23)
       REM-LU-ALIAS      = CICST6,                               (20)
       MODENAME          = MODDIS89,                              (5)
       LU62-CODE         = EBCDIC-273,
       REM-TSEL          = T'SMP22800',                          (29)
       REM-LISTENER-PORT = 102,                                  (30)
       REM-NSEL          = utmhost,                              (33)
       REM-APT           = (1,2,4),                              (25)
       REM-AEQ           = 1,                                    (26)
       APPL-CONTEXT      = UDTSEC,                               (27)
       ASSOCIATIONS      = 15,                                    (6)
       CONNECT           = 4,
       CONTWIN           = 5,                                    (28)
       OSITP-CODE        = ASCII
```

TNSX-Generierung

```
SMP22800.utmlu62\                                                    (24)
        PSEL V''                                                     (31)
        SSEL V''                                                     (32)
        TA RFC1006 123.45.67.89 PORT 102 T'SMP22800'      (33),(29),(30)
SMP22804\                                                            (21)
        PSEL V''                                                     (34)
        SSEL V''                                                     (35)
        TSEL RFC1006 T'SMP22804'                                     (36)
        TSEL LANINET A'22804'                                        (37)
```

BCAM-Generierung

```
BCIN LC0090,IPADR=(123,45,67,90)                              (38),(39)
BCMAP FU=DEF, SUBFU=GLOBAL, ES=LC0090,                         -
        NAME=SMP22804, PPORT#=22804, PTSEL-I='SMP22804'       (36),(37)
```

openUTM-Generierung

```
ACCESS-POINT OSIREP8,                                          -   (40)
    P-SEL = *NONE,                                             -   (31)
    S-SEL = *NONE,                                             -   (32)
    T-SEL = SMP22800,                                          -   (29)
    AEQ = 1                                                        (26)
OSI-CON SMP22804,
    P-SEL = *NONE,                                             -   (34)
    S-SEL = *NONE,                                             -   (35)
    T-SEL = C'SMP22804',                                       -   (36)
    N-SEL = C'LC0090',                                         -   (38)
    LOCAL-ACCESS-POINT = OSIREP8,                              -   (40)
    OSI-LPAP = ACICS04                                             (41)
```

Der Rest der openUTM-Generierung ist unverändert. Ebenso sind alle anderen Teile der Beispielgenerierung unverändert.

Änderungen bei Verwendung von openUTM-LU62 für Windows-Systeme

Bei Verwendung von openUTM-LU62 auf Windows-Systemen ändert sich lediglich die TRANSIT-Generierung. Da in diesem Fall anstelle von TRANSIT das Produkt IBM Communications Server for Windows verwendet wird, müssen die Generierungsparameter in der Syntax dieses Produktes definiert werden. Normalerweise werden die Generierungsparameter über Menüs eingetragen. Hieraus kann der Anwender eine Generierungsdatei (.acg) erzeugen und dabei Namen und Ablageort wählen.
Standardmäßig wird diese Datei im Verzeichnis
`<Installationsverzeichnis des IBM Communications Servers for Windows>\ PRIVATE`
abgelegt.

Generierung von IBM Communications Server for Windows

```
NODE=(
     ANYNET_SUPPORT=NONE
     COMPRESS_IN_SERIES=0
     CP_ALIAS=PO2CGE                                              (17)
     DEFAULT_PREFERENCE=NATIVE
     DISCOVERY_SUPPORT=DISCOVERY_CLIENT
     DLUR_SUPPORT=NORMAL
     FQ_CP_NAME=DESNI000.PO2CGE                                (17),(43)
     MAX_LS_EXCEPTION_EVENTS=200
     NODE_ID=0172000E                                          (11),(12)
     NODE_TYPE=END_NODE
     REGISTER_WITH_CDS=1
     REGISTER_WITH_NN=ALL
     TP_SECURITY_BEHAVIOR=VERIFY_EVEN_IF_NOT_DEFINED
)
PORT=(
     PORT_NAME=LAN0_04                                            (16)
     ACTIVATION_DELAY_TIMER=0
     DELAY_APPLICATION_RETRIES=1
     DLC_DATA=00000000000004
     DLC_NAME=LAN
     IMPLICIT_BRANCH_EXTENDER_LINK=0
     IMPLICIT_CP_CP_SESS_SUPPORT=1
     IMPLICIT_DEACT_TIMER=0
     IMPLICIT_DSPU_SERVICES=NONE
     IMPLICIT_HPR_SUPPORT=1
     IMPLICIT_LIMITED_RESOURCE=NO
     IMPLICIT_LINK_LVL_ERROR=0
     LINK_STATION_ROLE=NEGOTIABLE
     MAX_ACTIVATION_ATTEMPTS=0
     MAX_IFRM_RCVD=8
     MAX_RCV_BTU_SIZE=1033                                        (13)
     PORT_TYPE=SATF
```

```
        RETRY_LINK_ON_DISCONNECT=1
        RETRY_LINK_ON_FAILED_START=1
        RETRY_LINK_ON_FAILURE=1
        PORT_LAN_SPECIFIC_DATA=(
             ACK_DELAY=100
             ACK_TIMEOUT=10000
             ADAPTER_NUMBER=0
             BUSY_STATE_TIMEOUT=15
             IDLE_STATE_TIMEOUT=30
             INB_LINK_ACT_LIM=128
             LOCAL_SAP=04
             MAX_RETRY=10
             OUTSTANDING_TRANSMITS=16
             OUT_LINK_ACT_LIM=127
             POLL_TIMEOUT=8000
             POOL_SIZE=32
             REJECT_RESPONSE_TIMEOUT=10
             TEST_RETRY_INTERVAL=8
             TEST_RETRY_LIMIT=5
             TOT_LINK_ACT_LIM=255
             XID_RETRY_INTERVAL=8
             XID_RETRY_LIMIT=5
        )
   )
   LINK_STATION=(
        LS_NAME=P02CGE                                                (17)
        ACTIVATE_AT_STARTUP=1
        ACTIVATION_DELAY_TIMER=-1
        ADJACENT_BRANCH_EXTENDER_NODE=PROHIBITED
        ADJACENT_NODE_TYPE=SUBAREA_LEN
        AUTO_ACTIVATE_SUPPORT=0
        BRANCH_EXTENDER_LINK=0
        CP_CP_SESS_SUPPORT=0
        DEFAULT_NN_SERVER=0
        DELAY_APPLICATION_RETRIES=1
        DEPENDENT_LU_COMPRESSION=0
        DEPENDENT_LU_ENCRYPTION=OPTIONAL
        DEST_ADDRESS=40003745000104                                   (18)
        DISABLE_REMOTE_ACT=0
        DSPU_SERVICES=NONE
        FQ_ADJACENT_CP_NAME=DESNI000.M88                        (10),(43)
        HPR_LINK_LVL_ERROR=0
        HPR_SUPPORT=0
        INHERIT_PORT_RETRY_PARMS=1
        LIMITED_RESOURCE=NO
        LINK_DEACT_TIMER=0
        LINK_STATION_ROLE=USE_ADAPTER_DEFAULTS
        MAX_ACTIVATION_ATTEMPTS=-1
```

```
     MAX_IFRM_RCVD=0
     MAX_SEND_BTU_SIZE=1033                                        (13)
     NODE_ID=0172000E                                         (11),(12)
     PORT_NAME=LAN0_04                                             (16)
     RETRY_LINK_ON_DISCONNECT=1
     RETRY_LINK_ON_FAILED_START=1
     RETRY_LINK_ON_FAILURE=1
     REVERSE_ADDRESS_BYTES=0
     SOLICIT_SSCP_SESSION=0
     TG_NUMBER=0
     USE_DEFAULT_TG_CHARS=1
     USE_PU_NAME_IN_XID=0
)
DLUR_DEFAULTS=(
     DEFAULT_PU_NAME=P02CGE                                        (17)
     DLUS_RETRY_LIMIT=3
     DLUS_RETRY_TIMEOUT=5
)
LOCAL_LU=(
     LU_NAME=T02CGEI4                                               (4)
     PORT_NAME=LAN0_04                                             (16)
     DEFAULT_POOL=0
     LU_ALIAS=T02CGEI4                                             (19)
     LU_SESSION_LIMIT=100                                           (6)
     NAU_ADDRESS=0
     ROUTE_TO_CLIENT=0
     SYNCPT_SUPPORT=1
)
MODE=(
     MODE_NAME=MODDIS89                                             (5)
     AUTO_ACT=5
     COMPRESS_IN_SERIES=0
     COMPRESSION=PROHIBITED
     COS_NAME=#CONNECT
     ENCRYPTION_SUPPORT=NONE
     DEFAULT_RU_SIZE=1
     MAX_INCOMING_COMPRESSION_LEVEL=NONE
     MAX_NEGOTIABLE_SESSION_LIMIT=128
     MAX_OUTGOING_COMPRESSION_LEVEL=NONE
     MAX_RU_SIZE_UPPER_BOUND=4096                                   (7)
     MIN_CONWINNERS_SOURCE=5
     PLU_MODE_SESSION_LIMIT=15                                      (6)
     RECEIVE_PACING_WINDOW=1
)
```

```
MODE=(
     MODE_NAME=SNASVCMG
     AUTO_ACT=0
     COMPRESS_IN_SERIES=0
     COMPRESSION=PROHIBITED
     COS_NAME=SNASVCMG
     ENCRYPTION_SUPPORT=NONE
     DEFAULT_RU_SIZE=0
     MAX_INCOMING_COMPRESSION_LEVEL=NONE
     MAX_NEGOTIABLE_SESSION_LIMIT=2
     MAX_OUTGOING_COMPRESSION_LEVEL=NONE
     MAX_RU_SIZE_UPPER_BOUND=512
     MIN_CONWINNERS_SOURCE=1
     PLU_MODE_SESSION_LIMIT=2
     RECEIVE_PACING_WINDOW=1
)
PARTNER_LU=(
     FQ_PLU_NAME=DESNI000.A9CICST6                          (3),(43)
     ADJACENT_CP_NAME=DESNI000.M88                         (10),(43)
     CONV_SECURITY_VERIFICATION=1
     MAX_MC_LL_SEND_SIZE=32767
     PARALLEL_SESSION_SUPPORT=1
     PARTNER_LU_ALIAS=CICST6                                   (20)
     PREFERENCE=USE_DEFAULT_PREFERENCE
)
ADJACENT_NODE=(
     FQ_CP_NAME=DESNI000.M88                               (10),(43)
     LU_ENTRY=(
          WILDCARD_LU=0
          FQ_LU_NAME=DESNI000.A9CICST6                      (3),(43)
     )
)
SPLIT_STACK=(
     POOL_NAME=<Keine>
     STARTUP=1
)
SHARED_FOLDERS=(
     EXTENSION_LIST=(
     )
     CACHE_SIZE=256
)
VERIFY=(
     CFG_MODIFICATION_LEVEL=12
     CFG_VERSION_LEVEL=1
     CFG_LAST_SCENARIO=20
)
```

Änderung bei Verwendung von SNAP-IX, IBM Communications Server for Linux oder IBM Communications Server for AIX

Bei Verwendung von SNAP-IX, IBM Communications Server for Linux oder IBM Communications Server for AIX anstelle von TRANSIT müssen Generierungsparameter in der Syntax dieses Produktes definiert werden.

Generierung von SNAP-IX bzw. IBM Communications Server for Linux bzw. IBM Communications Server for AIX

```
[define_node_config_file]
major_version = 5
minor_version = 1
update_release = 1
revision_level = 47

[define_node]
cp_alias = PO2CGE                                                    (17)
description = SNAP-IX Knoten
fqcp_name = DESNI000.PO2CGE                                      (17),(43)
node_type = LEN_NODE
mode_to_cos_map_supp = YES
mds_supported = YES
node_id = <0172000E>                                             (11),(12)
max_locates = 1500
dir_cache_size = 255
max_dir_entries = 0
locate_timeout = 0
reg_with_nn = YES
reg_with_cds = YES
mds_send_alert_q_size = 100
cos_cache_size = 24
tree_cache_size = 40
tree_cache_use_limit = 40
max_tdm_nodes = 0
max_tdm_tgs = 0
max_isr_sessions = 1000
isr_sessions_upper_threshold = 900
isr_sessions_lower_threshold = 800
isr_max_ru_size = 16384
isr_rcv_pac_window = 8
store_endpt_rscvs = NO
store_isr_rscvs = NO
store_dlur_rscvs = NO
cos_table_version = VERSION_0_COS_TABLES
send_term_self = NO
disable_branch_awareness = NO
```

```
cplu_syncpt_support = YES
cplu_attributes = NONE
dlur_support = NO
pu_conc_support = YES
nn_rar = 128
max_ls_exception_events = 0
ptf_flags = NONE

[define_ethernet_dlc]
dlc_name = ETHER0
description = ""
neg_ls_supp = YES
initially_active = NO
adapter_number = 0
lan_type = 802_3_DIX

[define_ethernet_port]
port_name = ETSAP04                                              (16)
description = ""
dlc_name = ETHER0
port_type = PORT_SATF
port_number = 0
max_rcv_btu_size = 1033                                          (13)
tot_link_act_lim = 64
inb_link_act_lim = 0
out_link_act_lim = 0
ls_role = LS_NEG
implicit_dspu_services = NONE
implicit_dspu_template = ""
implicit_ls_limit = 0
act_xid_exchange_limit = 9
nonact_xid_exchange_limit = 5
ls_xmit_rcv_cap = LS_TWS
max_ifrm_rcvd = 7
target_pacing_count = 7
max_send_btu_size = 1033                                         (13)
mac_address = <000000000000>
lsap_address = 0x04
implicit_cp_cp_sess_support = NO
implicit_limited_resource = NO
implicit_deact_timer = 30
implicit_hpr_support = NO
implicit_link_lvl_error = NO
implicit_uplink_to_en = NO
effect_cap = 399360O
connect_cost = 0
byte_cost = 0
security = SEC_NONSECURE
```

```
prop_delay = PROP_DELAY_LAN
user_def_parm_1 = 128
user_def_parm_2 = 128
user_def_parm_3 = 128
initially_active = YES
window_inc_threshold = 1
test_timeout = 10
test_timer_retry = 5
xid_timer = 10
xid_timer_retry = 5
ack_timeout = 5000
p_bit_timeout = 5000
t2_timeout = 100
rej_timeout = 10
busy_state_timeout = 30
idle_timeout = 30
max_retry = 3

[define_ethernet_ls]
ls_name = M88
description = IBM-Host Muenchen
port_name = ETSAP04                                                (16)
adj_cp_name = DESNI000.M88                                       (10),(43)
adj_cp_type = END_NODE
mac_address = <400037450001>                                       (18)
lsap_address = 0x04
auto_act_supp = NO
tg_number = 0
limited_resource = NO
solicit_sscp_sessions = NO
pu_name = <0000000000000000>
disable_remote_act = NO
default_nn_server = NO
dspu_services = NONE
dspu_name = <0000000000000000>
dlus_name = <00000000000000000000000000000000000000>
bkup_dlus_name = <00000000000000000000000000000000000000>
hpr_supported = NO
hpr_link_lvl_error = NO
link_deact_timer = 30
use_default_tg_chars = YES
ls_attributes = SNA
local_node_id = <00000000>
cp_cp_sess_support = NO
effect_cap = 3993600
connect_cost = 0
byte_cost = 0
security = SEC_NONSECURE
```

```
prop_delay = PROP_DELAY_LAN
user_def_parm_1 = 128
user_def_parm_2 = 128
user_def_parm_3 = 128
target_pacing_count = 7
max_send_btu_size = 1033                                          (13)
ls_role = USE_PORT_DEFAULTS
max_ifrm_rcvd = 0
dlus_retry_timeout = 0
dlus_retry_limit = 0
branch_link_type = NONE
adj_brnn_cp_support = ALLOWED
dddlu_offline_supported = NO
initially_active = YES
restart_on_normal_deact = NO
react_timer = 30
react_timer_retry = 65535
test_timeout = 10
test_timer_retry = 5
xid_timer = 10
xid_timer_retry = 5
ack_timeout = 5000
p_bit_timeout = 5000
t2_timeout = 100
rej_timeout = 10
busy_state_timeout = 30
idle_timeout = 30
max_retry = 3

[define_partner_lu]
plu_alias = ""
description = (Auto defined - default LU)
fqplu_name = DESNIOOO.M88                                   (10),(43)
plu_un_name = <0000000000000000>
parallel_sess_supp = YES
max_mc_ll_send_size = 0
conv_security_ver = NO

[define_partner_lu]
plu_alias = CICST6                                                (20)
description = ""
fqplu_name = DESNIOOO.A9CICST6                               (3),(43)
plu_un_name = A9CICST6                                             (3)
parallel_sess_supp = YES
max_mc_ll_send_size = 0
conv_security_ver = NO
```

```
[define_local_lu]
lu_alias = TO2CGEI4                                              (19)
list_name = ""
description = ""
lu_name = TO2CGEI4                                                (4)
lu_session_limit = 100                                            (6)
pu_name = <0000000000000000>
nau_address = 0
default_pool = NO
syncpt_support = YES
lu_attributes = NONE
sscp_id = 0
disable = NO
sys_name = ""
timeout = 60
back_level = NO

[define_mode]
mode_name = MODDIS89                                              (5)
description = ""
max_neg_sess_lim = 100
plu_mode_session_limit = 15                                       (6)
min_conwin_src = 1
min_conloser_src = 0
auto_act = 0
receive_pacing_win = 4
max_receive_pacing_win = 0
default_ru_size = YES
max_ru_size_upp = 4096                                            (7)
max_ru_size_low = 0
cos_name = #CONNECT

[define_directory_entry]
resource_name = DESNI000.M88                                  (10),(43)
resource_type = ENCP_RESOURCE
description = (Auto defined - remote node)
parent_name = <00000000000000000000000000000000>
parent_type = ENCP_RESOURCE

[define_directory_entry]
resource_name = DESNI000.M88                                  (10),(43)
resource_type = LU_RESOURCE
description = (Auto defined - default LU)
parent_name = DESNI000.M88                                    (10),(43)
parent_type = ENCP_RESOURCE
```

```
[define_directory_entry]
resource_name = DESNI000.A9CICST6                                  (3),(43)
resource_type = LU_RESOURCE
description = ""
parent_name = DESNI000.M88                                        (10),(43)
parent_type = ENCP_RESOURCE
```

Parameter, die übereinstimmen müssen

Die wesentlichen Parameter in diesem Beispiel sind:

Parameter	**CICS Host**	**openUTM-LU62 UNIX-System**	**openUTM Endsystem**
LU-Name	DESNI000.A9CICST6	DESNI000.T02CGEI4	
LU-Alias-Name	CICST6	T02CGEI4	
CP-Name	DESNI000.M88	DESNI000.P02CGE	
IDBLK/IDNUM		017 2000E	
MAC-Adresse	400037450001	40002222000E	
MODE-Name	MODDIS89		
IP-Adresse		123.45.67.90	123.45.67.89
T-Selektor		SMP22804	SMP22800
Port-Nummer		22804	22800
AP-Title		(1,2,3)	(1,2,4)
AE-Qualifier		1	1
APPL-Context		UDTSEC	

Erläuterung im Detail:

(1) Wenn ein CICS-Programm eine UTM-Anwendung adressieren will, so muss es beim Kommando EXEC CICS ALLOCATE im Parameter SYSID den in CICS definierten Connection-Namen verwenden. Außerdem muss in CICS eine Session definiert sein, die auf diese Connection verweist.

(2) Zum zweiten muss das CICS-Programm beim Kommando EXEC CICS CONNECT PROCESS im Parameter PROCNAME den Transaktionscode des UTM-Anwendungsprogramms angeben.

(3) Der bei CICS generierte APPLID-Name ist der LU-Name von CICS. Er muss zusammen mit der in VTAM generierten Netzidentifikation in der TRANSIT-Generierung bei XRLU bzw. in der Generierung von SNAP-IX oder des IBM Communications Server bei der Definition der Partner-LU angegeben werden.

(4) Der LU-Name der Stellvertreter-LU muss bei VTAM im LU-Makro, bei CICS in der Connection-Definition und bei TRANSIT in der XLU-Anweisung bzw. bei SNAP-IX oder beim IBM Communications Server in der Definition der lokalen LU angegeben werden. In TRANSIT muss an dieser Stelle auch noch die Netzidentifikation angegeben werden. Wird SNAP-IX oder der IBM Communications Server verwendet und soll mit Transaktionssicherung gearbeitet werden, so muss in SNAP-IX bzw. im IBM Communications Server bei der Definition der lokalen LU Syncpoint-Support generiert werden.

Bei Verwendung von Enterprise Extender entfällt die Definition des LU-Namens in VTAM.

(5) Zwischen VTAM, CICS und dem System, auf dem openUTM-LU62 läuft, muss ein Mode-Name vereinbart werden. Dieser Mode-Name muss in VTAM, in der Connection-Definition von CICS, in der XMODE- und XLU-Anweisung von TRANSIT bzw. in der MODE-Definition von SNAP-IX bzw. des IBM Communications Server und in der openUTM-LU62-Generierung angegeben werden.

(6) Es muss festgelegt werden, wieviele LU6.2-Sessions zwischen CICS und openUTM-LU62 und wieviele parallele Verbindungen zwischen openUTM-LU62 und der UTM-Anwendung maximal aufgebaut werden dürfen. Diese beiden Werte sollten identisch gewählt werden, um Stausituationen zu vermeiden. Die Maximalzahl der Sessions muss eingetragen werden bei der Session-Definition in CICS und bei der XMODE-Anweisung in TRANSIT bzw. bei der MODE-Definition von SNAP-IX bzw. des IBM Communications Server. Die Maximalzahl der parallelen Verbindungen muss in der openUTM-LU62-Generierung und der openUTM-Generierung jeweils mit dem Parameter ASSOCIATIONS angegeben werden.

In TRANSIT-SERVER muss bei XLU ein Session-Limit angegeben werden, ebenso bei SNAP-IX bzw. beim IBM Communications Server in der Definition der lokalen LU. Dieses Session-Limit sollte um mindestens 2 größer gewählt werden als die oben angegebene Maximalzahl von Sessions.

(7) Zwischen CICS, VTAM und TRANSIT (bzw. SNAP-IX oder IBM Communications Server) muss die maximale RU-Size vereinbart werden. Sie muss bei CICS in der Session-Definition, bei VTAM, TRANSIT, SNAP-IX und IBM Communications Server in der Mode-Definition angegeben werden. Sie sollten hier nach Möglichkeit den Standardwert 4096 verwenden (siehe auch Abschnitt „DEFINE SESSIONS“ auf Seite 78).

(8) Wenn von der UTM-Anwendung aus ein CICS-Programm angesprochen werden soll, muss dieses Programm CICS in einer Program-Definition bekannt gemacht werden. Außerdem muss eine Transaction-Definition in CICS auf den Programmnamen verweisen.

(9) Der diesem CICS-Programm zugeordnete CICS-Transaktionscode muss in einer Transaction-Definition von CICS festgelegt werden. Um das Programm von der UTM-Anwendung ansprechen zu können, muss dann in der openUTM-Generierung eine LTAC-Anweisung mit dem CICS-Transaktionscode als RTAC-Parameter vorkommen.

(10) In den Start Options von VTAM wird unter anderem mit dem Parameter SSCPNAME der CP-Name des Host-Systems generiert. In SNAP-IX bzw. im IBM Communications Server muss dieser CP-Name ebenfalls generiert werden. In TRANSIT muss kein CP-Name angegeben werden.

(11), (12)
Wenn die SNA-Verbindung über Ethernet realisiert ist, müssen die Parameter IDBLK und IDNUM zwischen VTAM und TRANSIT (bzw. SNAP-IX oder IBM Communications Server) vereinbart werden.

(13) Bei Kommunikation mit SNA über Ethernet sollte ebenfalls die maximale Frame-Size zwischen VTAM und TRANSIT bzw. SNAP-IX oder IBM Communications Server vereinbart werden. Wenn die Generierungsparameter nicht übereinstimmen, einigen sich die beiden Partner manchmal, aber nicht immer auf den kleineren der beiden Werte.

(14) In VTAM muss die MAC-Adresse des Systems angegeben werden, auf dem openUTM-LU62 läuft.

(15) In der Logon Mode Table von VTAM wird der Name einer Mode-Tabelle verwendet. Dieser Name muss bei der Mode-Definition und bei der LU-Definition angegeben werden.

(16) Innerhalb von TRANSIT muss für den SNA-Ethernet-Anschluss ein Link-Name vergeben werden. Der Name ist bei der XLINK- und bei der XPU-Anweisung anzugeben. Bei SNAP-IX und beim IBM Communications Server wird stattdessen der Port-Name verwendet.

(17) Innerhalb von TRANSIT muss für das Partnersystem, auf dem CICS läuft, ein PU-Name vergeben werden. Der Name ist bei der XPU- und bei der XRLU-Anweisung anzugeben. Man wählt diesen Namen üblicherweise identisch zum PU-Namen in VTAM. Bei SNAP-IX und beim IBM Communications Server muss der PU-Name als CP-Name bei der Node-Definition eingetragen werden.

(18) In TRANSIT bzw. SNAP-IX oder IBM Communications Server muss die MAC-Adresse angegeben werden, über die der SNA-Host erreicht werden soll.

(19) Die Stellvertreter-LU erhält in TRANSIT bzw. SNAP-IX oder IBM Communications Server einen LU-Alias-Namen. Dieser Name muss nicht mit dem bei (4) definierten LU-Namen identisch sein. Er muss in der openUTM-LU62-Generierung angegeben werden.

(20) Das CICS-System erhält in TRANSIT bzw. SNAP-IX oder IBM Communications Server einen LU-Alias-Namen. Dieser Name muss nicht mit dem bei (3) definierten LU-Namen identisch sein. Er muss bei TRANSIT in der XLU- und in der XRLU-Anweisung, bei SNAP-IX und beim IBM Communications Server in der Definition der Partner-LU und bei der openUTM-LU62-Generierung im Parameter REM-LU-ALIAS angegeben werden.

(21) Nur bei Verwendung von TNSX.
Die Stellvertreter-AE von openUTM-LU62 muss in der openUTM-LU62-Generierung und im TNSX definiert werden.

(22), (23)
Für die Stellvertreter AE muss ein Application Process Title und ein Application Entity Qualifier festgelegt werden. Beide sind in der openUTM-LU62-Generierung und in der openUTM-Generierung anzugeben.

(24) Nur bei Verwendung von TNSX.
Die von der UTM-Anwendung verwendete AE muss in der openUTM-LU62-Generierung und im TNSX definiert werden.

(25), (26)
Für die UTM-Anwendung muss ein Application Process Title und ein Application Entity Qualifier festgelegt werden. Beide sind in der openUTM-LU62-Generierung und in der openUTM-Generierung anzugeben. Die UTM-Anwendung und die Stellvertreter-AE müssen unterschiedliche Application Process Title haben. Die Application Entity Qualifier können jedoch identisch sein.

(27) Zwischen openUTM-LU62 und der UTM-Anwendung muss ein Application Context vereinbart werden. Es empfiehlt sich, hier den Standardwert UDTSEC zu verwenden. Wird UDTAC verwendet, so muss im Falle von TRANSIT bei PAIR_EXT der dritte und vierte Parameter jeweils auf NONE gesetzt werden. Bei Verwendung von UDTAC sind keine verteilten Transaktionen möglich. Andere Werte sind nicht zulässig.

(28) Die zwischen openUTM-LU62 und der UTM-Anwendung verfügbaren parallelen Verbindungen müssen mittels Generierung in Contention-Winner- und Contention-Loser-Verbindungen aufgeteilt werden. Die CONTWIN-Werte aus der openUTM-LU62-Generierung und der openUTM-Generierung sollten aufaddiert genau die generierte Anzahl von parallelen Verbindungen ergeben.

(29), (30)
Der T-Selektor sowie die Port-Nummer des Access-Point der UTM-Anwendung müssen auch im TNSX-Eintrag oder direkt in der openUTM-LU62-Generierung (24) eingetragen sein. Damit wird über CMX die Verbindung zwischen openUTM-LU62 und der UTM-Anwendung hergestellt.

(31), (32)
Entsprechend den ISO-Protokollvorschriften muss für die von der UTM-Anwendung verwendete AE ein S-Selektor und ein P-Selektor vergeben werden. Diese S- und P-Selektoren müssen im TNSX und in der openUTM-Generierung angegeben werden. Es empfiehlt sich, leere S- und P-Selektoren zu verwenden.
Falls kein TNSX verwendet wird, sind S-und P-Selektor immer leer.

(33) Der Host-Name des Systems, auf dem die UTM-Anwendung läuft, muss eingetragen werden

- im TNSX als IP-Adresse bei dem Namen, der in der Generierung von openUTM-LU62 bei REM-AE verwendet wird oder
- direkt in der openUTM-LU62-Generierung als Host-Name im DNS (bzw. in `/etc/hosts`).

Wenn openUTM-LU62 und die UTM-Anwendung auf demselben Rechner laufen, kann hier localhost verwendet werden.

(34), (35)
Entsprechend den ISO-Protokollvorschriften muss für die Stellvertreter-AE ein S-Selektor und ein P-Selektor vergeben werden. Diese S- und P-Selektoren müssen im TNSX und in der openUTM-Generierung angegeben werden. Es empfiehlt sich, leere S- und P-Selektoren zu verwenden.
Falls kein TNSX verwendet wird, sind S-und P-Selektor immer leer.

(36) Der T-Selektor der Stellvertreter-AE muss eingetragen werden

- im TNSX oder
- direkt in der openUTM-LU62-Generierung und
- in der OSI-CON-Anweisung der UTM-Generierung.

(37) Die Port-Nummer der Stellvertreter-AE muss eingetragen werden

- im TNSX oder
- direkt in der openUTM-LU62-Generierung und
- in der OSI-CON-Anweisung der UTM-Generierung.

Für jede Instanz von openUTM-LU62 muss eine eigene Port-Nummer vergeben werden.

(38) In der OSI-CON-Anweisung der UTM-Generierung muss der Name des Rechners eingetragen werden, auf dem openUTM-LU62 läuft. Auf UNIX-Systemen ist dies üblicherweise der DNS-Rechnername (z.B. *local* für den eigenen Rechner), auf BS2000-Systemen der BCAM-Name.

(39) Die IP-Adresse des Rechners, auf dem openUTM-LU62 läuft, ist üblicherweise im DNS eingetragen. Bei Verwendung von openUTM unter BS2000/OSD wird die IP-Adresse mit dem Kommando BCIN eingetragen.

(40) Die von der UTM-Anwendung verwendete AE erhält in der openUTM-Generierung einen symbolischen Namen. Dieser kommt innerhalb der openUTM-Generierung an zwei Stellen vor.

(41) Die Stellvertreter-AE und somit das CICS-System erhält in openUTM einen symbolischen Namen, den sogenannten LPAP-Namen. Dieser Name wird in der openUTM-Generierung mit einer OSI-LPAP-Anweisung generiert. Die OSI-CON-Anweisung muss auf diesen Namen verweisen. Wenn von der UTM-Anwendung ein CICS-Programm adressiert werden soll, muss dieser Name auch in einer LTAC-Anweisung oder im UTM-Programm beim APRO-Aufruf angegeben werden.

(42) Wenn von der UTM-Anwendung aus ein CICS-Programm adressiert werden soll, so muss die openUTM-Generierung eine entsprechende LTAC-Anweisung enthalten, die mit den Parametern LPAP und RTAC auf das CICS-System und den dortigen Transaktionscode verweist. Das UTM-Anwendungsprogramm muss beim APRO diesen LTAC-Namen verwenden. Zusätzlich kann auch der LPAP-Name beim APRO-Aufruf angegeben werden.

(43) Die Netzidentfikation des SNA-Netzes wird in den Startparametern von VTAM festgelegt. Dieser Name muss auch an mehreren Stellen bei TRANSIT bzw. SNAP-IX bzw. IBM Communications Server angegeben werden.

4.2 openUTM-CICS-Kopplung programmieren

Die LU6.2-Kommunikation zwischen CICS und openUTM kann bei CICS nur mit Hilfe von DTP (Distributed Transaction Processing) programmiert werden. Asynchronous Processing mittels START und RETRIEVE ist bei LU6.2 nicht möglich.

CICS-Programme werden üblicherweise mit CICS-Kommandos der Art

```
EXEC CICS command option
```

programmiert. In den folgenden Abschnitten sind die wichtigsten CICS-Kommandos für die LU6.2-Kommunikation beschrieben. Anstelle der üblichen CICS-Kommandos können auch die CPI-C-Aufrufe verwendet werden. Da CPI-C auch in Nicht-CICS-Umgebung eine wichtige Programmschnittstelle für die LU6.2-Kommunikation ist, wird ab Seite 151 auch die Verwendung dieser Schnittstelle beschrieben.

Für die LU6.2-Kommunikation stehen bei CICS die folgenden Kommandos zur Verfügung:

- ALLOCATE
- CONNECT PROCESS
- CONVERSE
- EXTRACT ATTRIBUTES
- EXTRACT PROCESS
- FREE
- ISSUE ABEND
- ISSUE CONFIRMATION
- ISSUE ERROR
- ISSUE PREPARE
- ISSUE SIGNAL
- RECEIVE
- SEND
- SYNCPOINT
- SYNCPOINT ROLLBACK
- WAIT CONVID

Bei vielen dieser Kommandos gibt es mehrere Varianten. In den CICS-Handbüchern ist die LU6.2-Variante dann immer mit APPC bezeichnet, z.B. SEND (APPC).

Bei openUTM stehen ähnlich wie bei CICS mehrere Programmschnittstellen zur Auswahl. Die folgende Beschreibung geht meist davon aus, dass die UTM-Anwendungen mittels KDCS programmiert sind. Es wird weiter unten aber auch auf CPI-C eingegangen.

4.2.1 CICS-Kommandos für CICS-Auftraggeber-Vorgänge

ALLOCATE

Mit diesem Kommando versucht ein CICS-Auftraggeber-Vorgang eine Session für einen Auftragnehmer-Vorgang zu belegen.

Das Kommando hat folgende Syntax.

```
EXEC CICS ALLOCATE SYSID(name)
                   [ PROFILE(name) ]
                   [ NOQUEUE ]
```

SYSID(name)
: Bezeichnet den Namen für die ferne UTM-Anwendung, wie er in einer Connection-Definition festgelegt wurde.

PROFILE(name)
: Bezeichnet ein bestimmtes Profil für die Kommunikation mit dem Partner. Profile werden mittels CICS-Definitionen festgelegt. Anstelle von SYSID und PROFILE kann auch der Parameter PARTNER angegeben werden. Dieser Partner muss dann per CICS-Definition eingerichtet sein.

NOQUEUE
: Bedeutet, dass nicht gewartet werden soll, falls nicht sofort eine Session verfügbar ist.

CICS gibt nach erfolgreichem ALLOCATE-Kommando im Feld EIBRSRCE eine CONVID zurück. Diese CONVID muss in allen nachfolgenden Aufrufen angegeben werden, die sich auf die hier belegte Session beziehen.

CONNECT PROCESS

Mit diesem Kommando beginnt die LU6.2-Conversation.

Das Kommando hat folgende Syntax:

```
EXEC CICS CONNECT PROCESS CONVID(name)
                          PROCNAME(data-area)
                          PROCLENGTH(data-value)
                          SYNCLEVEL(data-value)
```

CONVID(name)
: Bezeichnet die durch ALLOCATE reservierte Session.

PROCNAME(data-area) und PROCLENGTH(data-value)
: Bezeichnen den UTM-Transaktionscode.

SYNCLEVEL(data-value)
: Bezeichnet das Synchronisations-Level des LU6.2-Protokolls. Der Parameter entspricht dem Feld KCOF beim UTM-Aufruf APRO.
0 entspricht KCOF=B (Basisfunktionen).
1 entspricht KCOF=H (Handshake-Funktionen).
2 entspricht KCOF=C (Commit-Funktionen).

Die Verwendung der Parameter PIPLIST und PIPLENGTH zum Senden von PIP-Daten (program initialization parameter) ist nicht möglich.

SEND

Dieses Kommando sendet eine Dialogteilnachricht.

Das Kommando hat folgende Syntax:

```
EXEC CICS SEND CONVID(name)
               FROM(data-area)
               LENGTH(data-value)
               [ { INVITE | LAST } ]
               [ { CONFIRM | WAIT } ]
```

CONVID(name)
: Bezeichnet die durch ALLOCATE reservierte Session.

FROM(data-area) und LENGTH(data-value)
: Bezeichnen die Adresse und die Länge der Nachricht.

INVITE
: Veranlasst, dass mit dem Senden der Nachricht das Senderecht (change direction) an den Partner übertragen wird. Das Eintreffen des Senderechts wird von openUTM als das Ende der Dialog-Nachricht des Partners interpretiert. Das UTM-Anwendungsprogramm kann dies erkennen, wenn es beim nächsten MGET den Returncode 10Z erhält.

LAST
: Veranlasst, dass nach dem Senden der Nachricht der Vorgang beendet wird. Das UTM-Anwendungsprogramm erkennt dies an der Anzeige KCVGST=C.

CONFIRM
: Veranlasst, dass nach dem Senden der Nachricht eine Verarbeitungsquittung vom UTM-Anwendungsprogramm gefordert wird. Das UTM-Anwendungsprogramm erkennt dies an der Anzeige KCRMGT=H. Der Parameter darf nur bei Sync-Level 1 oder 2 angegeben werden.

WAIT
: Veranlasst, dass das CICS-Kommando erst zurückkehrt, wenn CICS die Nachricht tatsächlich abgesendet hat. Folgt ein SYNCPOINT-Kommando auf das SEND-Kommando, so sollten Sie diesen Parameter nicht verwenden.

RECEIVE

Dieses Kommando wird verwendet, um eine von einem UTM-Vorgang gesendete Nachricht zu empfangen.

Das Kommando hat folgende Syntax:

```
EXEC CICS RECEIVE CONVID(name)
                  INTO(data-area)
                  LENGTH(data-value)
                  [ MAXLENGTH(data-value) ]
                  [ NOTRUNCATE ]
```

CONVID(name)
: Bezeichnet die durch ALLOCATE reservierte Session.

INTO(data-area) und MAXLENGTH(data-value)
: Bezeichnet die Adresse und Länge des Puffers zur Aufnahme der Nachricht.

LENGTH(data-value)
: Hier gibt CICS die Länge der erhaltenen Nachricht zurück.

NOTRUNCATE
: Der Rest der Nachricht wird nur übertragen, wenn er kürzer als MAXLENGTH ist. Ist NOTRUNCATE nicht angegeben, so wird eine Nachricht bis zur Länge MAXLENGTH zugestellt. Der Rest wird verworfen. Ist NOTRUNCATE angegeben, so können weitere Teile mit RECEIVE angefordert werden.

CONVERSE

Dieses Kommando bewirkt, dass eine Nachricht an den Partner gesendet und auf die Antwort gewartet wird. Dieses Kommando hat dieselbe Wirkung wie die Kommandofolge SEND INVITE, RECEIVE.

Das Kommando hat folgende Syntax:

```
EXEC CICS CONVERSE CONVID(name)
                   FROM(data-area)
                   FROMLENGTH(data-value)
                   INTO(data-area)
                   TOLENGTH(data-value)
                   [ MAXLENGTH(data-value) ]
                   [ NOTRUNCATE ]
```

Die Parameter haben die gleiche Bedeutung wie bei dem SEND- bzw. RECEIVE-Kommando beschrieben.

ISSUE ABEND

Mit diesem Kommando wird der Vorgang zum UTM-Anwendungsprogramm abnormal beendet. Der Aufruf entspricht dem UTM-Aufruf CTRL AB.

Das Kommando hat folgende Syntax:

```
EXEC CICS ISSUE ABEND CONVID(name)
```

CONVID(name)
: Bezeichnet die durch ALLOCATE reservierte Session.

ISSUE CONFIRMATION

Mit diesem Kommando wird eine positive Verarbeitungsquittung an das UTM-Anwendungsprogramm gesendet. Das Kommando ist nur dann erlaubt, wenn Sync-Level 1 oder 2 eingestellt wurde und zuvor das UTM-Anwendungsprogramm eine Verarbeitungsquittung angefordert hat. Das UTM-Anwendungsprogramm erkennt die Ankunft der Verarbeitungsquittung an der Anzeige KCRMGT=C.

Das Kommando hat folgende Syntax:

```
EXEC CICS ISSUE CONFIRMATION CONVID(name)
```

CONVID(name)
: Bezeichnet die durch ALLOCATE reservierte Session.

ISSUE ERROR

Mit diesem Kommando wird eine Fehlermeldung oder negative Verarbeitungsquittung an das UTM-Anwendungsprogramm gesendet. Das UTM-Anwendungsprogramm erhält beim MGET KCRMGT=E.

Das Kommando hat folgende Syntax:

```
EXEC CICS ISSUE ERROR CONVID(name)
```

CONVID(name)
: Bezeichnet die durch ALLOCATE reservierte Session.

SYNCPOINT

Mit diesem Kommando werden alle Auftragnehmer-Vorgänge aufgefordert, das Transaktionsende einzuleiten. Das Kommando ist nur bei Sync-Level 2 erlaubt. Das UTM-Anwendungsprogramm erkennt dies beim MGET an der Anzeige KCTAST=P.

Das Kommando hat folgende Syntax:

```
EXEC CICS SYNCPOINT
```

SYNCPOINT ROLLBACK

Mit diesem Kommando wird die Transaktion zurückgesetzt. Das Kommando ist nur bei Sync-Level 2 erlaubt. Das UTM-Anwendungsprogramm erkennt das Rücksetzen der Transaktion beim MGET an der Anzeige KCTAST=R.

Das Kommando hat folgende Syntax:

```
EXEC CICS SYNCPOINT ROLLBACK
```

ISSUE PREPARE

Mit diesem Kommando wird der Partner aufgefordert, das Transaktionsende einzuleiten. Das Kommando ist nur bei Sync-Level 2 erlaubt.

Das Kommando hat folgende Syntax:

```
EXEC CICS ISSUE PREPARE CONVID(name)
```

CONVID(name)
: Bezeichnet die durch ALLOCATE reservierte Session.

ISSUE PREPARE unterscheidet sich von SYNCPOINT dadurch, dass nur einer von eventuell mehreren Auftragnehmer-Vorgängen zum Einleiten des Transaktionsendes aufgefordert wird. Dem ISSUE PREPARE muss später noch ein SYNCPOINT oder SYNCPOINT ROLLBACK folgen.

Bei einer CICS-CICS-Kopplung kann der Auftragnehmer-Vorgang nach ISSUE PREPARE noch mittels ISSUE ERROR eventuelle Fehler melden und dem Auftraggeber die Entscheidung überlassen, ob er SYNCPOINT oder SYNCPOINT ROLLBACK durchführen will. Eine UTM-Anwendung kann jedoch den dem ISSUE ERROR entsprechenden Aufruf nicht absetzen, sondern kann nur mit einer PEND-Variante antworten. Somit liegt nach dem ISSUE PREPARE die Entscheidung über Transaktionsende oder Rücksetzen der Transaktion beim UTM-Anwendungsprogramm.

FREE

Das Kommando gibt eine mittels ALLOCATE belegte Session wieder frei.

Das Kommando hat folgende Syntax:

```
EXEC CICS FREE CONVID(name)
```

CONVID(name)
: Bezeichnet die durch ALLOCATE reservierte Session.

WAIT CONVID

Das Kommando sorgt dafür, dass zuvor mittels CONNECT PROCESS oder SEND erstellte Daten tatsächlich abgesendet werden.

Das Kommando hat folgende Syntax:

```
EXEC CICS WAIT CONVID(name)
```

CONVID(name)
: Bezeichnet die durch ALLOCATE reservierte Session.

ISSUE SIGNAL

Das Kommando sendet eine Aufforderung an das UTM-Anwendungsprogramm, die Sendeberechtigung abzugeben. Dieses Kommando ist bei einer Kopplung zu openUTM wirkungslos, da das UTM-Anwendungsprogramm nicht von der Ankunft einer derartigen Aufforderung unterrichtet wird.

4.2.2 CICS-Kommandos für CICS-Auftragnehmer-Vorgänge

EXTRACT PROCESS

Das Kommando dient dazu, vom UTM-Anwendungsprogramm mittels APRO gesetzte Parameter abzufragen.

Das Kommando hat folgende Syntax:

```
EXEC CICS EXTRACT PROCESS [ PROCNAME(data-area) ]
                          [ MAXPROCLENGTH(data-value) ]
                          [ PROCLENGTH(data-value) ]
                          [ SYNCLEVEL(data-area) ]
```

PROCNAME(data-area), MAXPROCLENGTH(data-value) und PROCLENGTH(data-value)
: Erfragt den vom UTM-Anwendungsprogramm bei APRO im Parameter KCRN gesetzten CICS-Transaktionscode.

SYNCLEVEL(data-area)
: Erfragt das von UTM-Anwendungsprogramm bei APRO im Parameter KCOF gesetzte Synchronisations-Level des LU6.2-Protokolls.
0 entspricht KCOF=B (Basisfunktionen).
1 entspricht KCOF=H (Handshake-Funktionen).
2 entspricht KCOF=C (Commit-Funktionen).

RECEIVE

Dieses Kommando wird verwendet, um eine von einem UTM-Auftraggeber-Vorgang gesendete Nachricht zu empfangen.

Syntax und Bedeutung sind gleich wie bei Auftraggeber-Vorgängen. Allerdings muss beim Lesen der Nachricht vom Auftraggeber CONVID nicht angegeben werden.

SEND

Dieses Kommando sendet eine Dialogteilnachricht.

Syntax und Bedeutung sind gleich wie bei Auftraggeber-Vorgängen. Allerdings muss beim Senden an den Auftraggeber CONVID nicht angegeben werden.

CONVERSE

Dieses Kommando bewirkt, dass eine Nachricht an den Partner gesendet und auf die Antwort gewartet wird. Dieses Kommando hat dieselbe Wirkung wie die Kommandofolge SEND INVITE, RECEIVE.

Syntax und Bedeutung sind gleich wie bei Auftraggeber-Vorgängen.

ISSUE ABEND

Mit diesem Kommando wird der Vorgang zum UTM-Anwendungsprogramm abnormal beendet.

Syntax und Bedeutung sind gleich wie bei Auftraggeber-Vorgängen. Allerdings muss beim ISSUE ABEND zum UTM-Auftraggeber-Vorgang CONVID nicht angegeben werden.

ISSUE CONFIRMATION

Mit diesem Kommando wird eine positive Verarbeitungsquittung an das UTM-Anwendungsprogramm gesendet.

Syntax und Bedeutung sind gleich wie bei Auftraggeber-Vorgängen. Allerdings muss beim ISSUE CONFIRMATION zum UTM-Auftraggeber-Vorgang CONVID nicht angegeben werden.

ISSUE ERROR

Mit diesem Kommando wird eine Fehlermeldung oder negative Verarbeitungsquittung an das UTM-Anwendungsprogramm gesendet.

Ruft das CICS-Auftragnehmerprogramm ISSUE ERROR auf, wenn es vom UTM-Anwendungsprogramm zum Einleiten des Transaktionsendes aufgefordert wird, so führt dies zu einem Rücksetzen der Transaktion.

Syntax und Bedeutung sind gleich wie bei Auftraggeber-Vorgängen. Allerdings muss beim ISSUE ERROR zum UTM-Auftraggeber-Vorgang CONVID nicht angegeben werden.

SYNCPOINT

Mit dem Kommando wird auf Anforderung des Auftraggebers das Transaktionsende eingeleitet.

Wichtig ist bei einer Kopplung zu openUTM, dass ein CICS-Programm SYNCPOINT nur dann verwenden darf, wenn das UTM-Anwendungsprogramm es vorher zum Einleiten des Transaktionsendes aufgefordert hat. Das CICS-Programm kann am Parameter STATE oder im Feld EIBSYNC eines jeden CICS-Kommandos erkennen, ob es zur Einleitung des Transaktionsendes aufgefordert wurde. STATE muss dann einen der Werte SYNCFREE, SYNCRECEIVE oder SYNCSEND annehmen. EIBSYNC muss den Wert X'FF' annehmen.

Die Syntax ist gleich wie bei Auftraggeber-Vorgängen.

SYNCPOINT ROLLBACK

Mit diesem Kommando wird die Transaktion zurückgesetzt.

Syntax und Bedeutung sind gleich wie bei Auftraggeber-Vorgängen.

FREE

Das Kommando gibt die Session zum Auftraggeber-Vorgang wieder frei.

Syntax und Bedeutung sind gleich wie bei Auftraggeber-Vorgängen. Allerdings muss beim Auftraggeber-Vorgang CONVID nicht angegeben werden.

WAIT CONVID

Das Kommando sorgt dafür, dass zuvor mittels SEND erstellte Daten tatsächlich abgesendet werden.

Syntax und Bedeutung sind gleich wie bei Auftraggeber-Vorgängen.

ISSUE SIGNAL

Wie beim Auftraggeber-Vorgang beschrieben sollte das Kommando nicht angewendet werden.

ISSUE PREPARE

Das Kommando darf in einem Auftraggeber-Vorgang nicht abgesetzt werden.

RETURN

Das Kommando RETURN impliziert das Senden noch nicht gesendeter Nachrichten mit einem Sicherungsrequest an den Partner und anschließender Sessionfreigabe. Das RETURN-Kommando ist nur zulässig, wenn der openUTM-Auftraggeber-Vorgang bereits CTRL PE oder PEND FI gegeben hat (Indikator EIBFREE im EIB) oder nachdem das CICS-Programm mittels ISSUE-ABEND den Vorgang abnormal beendet hat.

Das Kommando hat folgende Syntax:

```
EXEC CICS RETURN
```

4.2.3 Hinweise zur CICS-Programmierung

Bei einer openUTM-CICS-Kopplung gibt es noch Einschränkungen gegenüber dem bei sonstigen LU6.2-Kopplungen verfügbaren Funktionsumfang.

1. Die LU6.2-Kommunikation zwischen CICS und openUTM kann bei CICS nur mit Hilfe von DTP (Distributed Transaction Processing) programmiert werden. Asynchronous Processing mittels START und RETRIEVE oder Distributed Program Link mittels EXEC CICS LINK ist mit openUTM-LU62 nicht möglich.
2. Ein CICS-Anwendungsprogramm als Auftragnehmer darf nie eigenständig SYNCPOINT oder ISSUE PREPARE verwenden, sondern nur, wenn es vom UTM-Anwendungsprogramm dazu aufgefordert wurde.
3. Basic Conversation ist nicht möglich. Basic Conversation wird bei CICS mit Kommandos programmiert, die mit GDS beginnen, also z.B. EXEC CICS GDS ALLOCATE.
4. Ruft das CICS-Auftragnehmerprogramm ISSUE ERROR auf, wenn es vom UTM-Anwendungsprogramm zum Einleiten des Transaktionsendes aufgefordert wird, so führt dies zu einem Rücksetzen der Transaktion.
5. PIP-Daten können beim CONNECT PROCESS nicht verwendet werden. Die Daten gehen verloren.
6. Es ist nicht möglich, auf unterschiedlichen Verbindungen zum selben Partner unterschiedliche Mode-Namen zu verwenden. Bei CICS/ESA V4.1 wird der Mode-Name bei der Session-Definition eingestellt und kann damit an der Programmschnittstelle implizit über den SYSID-Parameter des ALLOCATE-Aufrufs ausgewählt werden.
7. Wenn eine LU6.2-Conversation zu openUTM-LU62 nicht aufgebaut werden kann, weil openUTM-LU62 die entsprechende Verbindung zur UTM-Anwendung nicht aufbauen kann, so erhält CICS keine detaillierten Ablehnungsgründe. Detaillierte Ablehnungsgründe sind nur in der Protokolldatei von openUTM-LU62 zu finden.
8. Ist der Generierungsparameter UTM-ASYNC=YES gesetzt, so baut openUTM-LU62 LU6.2-Conversations immer mit Sync-Level 1 oder 2 auf, nie mit Sync-Level 0. Das Sync-Level einer ankommenden Conversation kann beim CICS-API mittels EXTRACT PROCESS erfragt werden.

4.2.4 Vergleich mit KDCS-Aufrufen

Im folgenden soll versucht werden, die CICS-Kommandos in ihrer Semantik mit KDCS-Aufrufen zu vergleichen.

Adressierung eines fernen Vorgangs mit Transaktionssicherung

```
ALLOCATE                              APRO DM/AM KCOF=C
CONNECT PROCESS SYNCLEVEL(2)
```

Adressierung eines fernen Vorgangs ohne Transaktionssicherung

```
ALLOCATE                              APRO DM KCOF=H
CONNECT PROCESS SYNCLEVEL(1)
```

bzw.

```
ALLOCATE                              APRO DM/AM KCOF=B
CONNECT PROCESS SYNCLEVEL(0)
```

Nachrichtenaustausch ohne Sicherungspunkt

```
SEND INVITE WAIT                      MPUT NE KCRN=“>...“
RECEIVE                               PEND KP
                                      .
                                      INIT
                                      MGET KCRN=“>...“
```

Senden einer Nachricht und Anforderung zum Transaktionsende

```
SEND                                  MPUT NE KCRN=“>...“
SYNCPOINT                             CTRL PR KCRN=“>...“
                                      PEND KP
                                      .
                                      INIT
                                      MGET NT KCRN=“>...“
```

Senden einer Nachricht und Anforderung Transaktions- und Vorgangsende

```
SEND LAST                             MPUT NE KCRN=“>...“
SYNCPOINT                             CTRL PE KCRN=“>...“
                                      PEND KP
                                      .
                                      INIT
                                      MGET NT KCRN=“>...“
```

Nachrichtenaustausch und Anforderung zum Transaktionsende

```
SEND INVITE                          MPUT NE KCRN=“>...“
SYNCPOINT                            PEND RE
RECEIVE                              .
                                     INIT
                                     MGET NT KCRN=“>...“
```

Rücksetzen der Transaktion

```
SYNCPOINT ROLLBACK                   MPUT RM KCRN=“<...“
                                     PEND RS
                                     .
                                     INIT
                                     MGET NT KCRN=“<...“
```

Rücksetzen der Transaktion und Abbruch des Vorgangs

```
SYNCPOINT ROLLBACK                   PEND ER/FR
ISSUE ABEND
```

Nachrichtenaustausch mit Verarbeitungsquittung

```
SEND INVITE CONFIRM                  MPUT NE KCRN=“>...“
RECEIVE                              MPUT HM KCRN=“>...“
                                     PEND KP
                                     .
                                     INIT
                                     MGET NT KCRN=“>...“
```

Senden einer Antwort mit positiver Verarbeitungsquittung

```
ISSUE CONFIRMATION                   MPUT NE KCRN=“>...“
SEND                                 PEND FI
```

Senden einer Antwort mit negativer Verarbeitungsquittung

```
ISSUE ERROR                          MPUT EM KCRN=“>...“
```

Abbruch des Dialogs zum Partner

```
ISSUE ABEND                          CTRL AB
```

4.2.5 Beispiele für openUTM-CICS-Kommunikation

Start eines openUTM-Dialogvorgangs aus einem CICS-Anwendungsprogramm

1. Einschrittvorgang mit Transaktionssicherung

```
--->
    RECEIVE Client
    ALLOCATE
    CONNECT PROCESS CONVID
        SYNCLEVEL(2)
    SEND INVITE WAIT CONVID
                            -- FMH5+Daten ----->
                                                INIT PU ENDTA=O,SEND=Y
                                                MGET    VGST=O,TAST=O
                                                MPUT NE
                                                PEND KP
                            <- Daten -----------
    RECEIVE CONVID
    SEND LAST CONVID
    SYNCPOINT
                            -- Daten+
                               RequestCommit -->
                                                INIT PU ENDTA=O,SEND=N
                                                MGET    VGST=O,TAST=P
                                                PEND FI
                            <- Committed -------
    =NORMAL
    SEND Client
<---
    RETURN
```

Im obigen Ablaufbeispiel wurde im UTM-Anwendungsprogramm der Aufruf „INIT PU“ verwendet, weil man damit Transaktionsende (KCENDTA) und Senderecht (KCSEND) prüfen kann. Wenn es aufgrund der Ablauflogik nicht notwendig ist, diese Werte abzufragen, kann auch der normale INIT verwendet werden. Beim MGET wurden die Anzeigen KCVGST (Vorgangsstatus) und KCTAST (Transaktionsstatus) mit VGST bzw. TAST abgekürzt. Das SYNCPOINT-Kommando im CICS-Anwendungsprogramm kehrt erst zurück, nachdem das UTM-Anwendungsprogramm PEND FI aufgerufen hat. Dies ist dargestellt durch die Zeile „=NORMAL“.

RequestCommit und Committed sind Bestandteile des LU6.2-Protokolls. Wenn das CICS-Anwendungsprogramm mehr als einen Auftragnehmer hat, so benutzt es teilweise ein zweiphasiges Protokoll mit den Protokollelementen Prepare, RequestCommit, Committed, Forget.

Im CICS-Anwendungsprogramm kann vor dem SYNCPOINT noch das Kommando ISSUE PREPARE aufgerufen werden. Dieses hat bei einem PEND ER/FR im UTM-Anwendungsprogramm gewisse Vorteile. Siehe hierzu weiter unten.

2. Keine Daten vom UTM-Anwendungsprogramm im CICS-Anwendungsprogramm benötigt und mit Transaktionssicherung

Wenn es von der Ablauflogik her nicht erforderlich ist, dass das aufgerufene UTM-Anwendungsprogramm dem CICS-Anwendungsprogramm Daten zurücksendet, so kann die Kommunikation wie im folgenden Beispiel einfacher programmiert werden.

```
--->
    RECEIVE Client
    ALLOCATE
    CONNECT PROCESS CONVID
        SYNCLEVEL(2)
    SEND LAST CONVID
    SYNCPOINT
                            -- FMH5+Daten+
                               RequestCommit -->
                                              INIT PU ENDTA=F,SEND=N
                                              MGET    VGST=O,TAST=P
                                              PEND FI
                            <- Committed -------
    =NORMAL
    SEND Client
<---
    RETURN
```

3. Zwei Transaktionen in einem Vorgang, erste Variante

```
--->
    RECEIVE Client
    ALLOCATE
    CONNECT PROCESS CONVID
        SYNCLEVEL(2)
    SEND INVITE CONVID
    SYNCPOINT
                              -- FMH5+Daten+
                                 RequestCommit -->
                                                  INIT PU ENDTA=C,SEND=Y
                                                  MGET    VGST=O,TAST=P
                                                  MPUT NE
                                                  PEND RE
                              <- Committed+Daten -
    =NORMAL
    RECEIVE CONVID
    SEND Client
<---
--->
    RECEIVE Client
    SEND LAST CONVID
    SYNCPOINT
                              -- Daten+
                                 RequestCommit -->
                                                  INIT PU ENDTA=F,SEND=N
                                                  MGET    VGST=O,TAST=P
                                                  PEND FI
                              <- Committed -------
    =NORMAL
    SEND Client
<---
    RETURN
```

4. Zwei Transaktionen in einem Vorgang, zweite Variante

```
--->
    RECEIVE Client
    ALLOCATE
    CONNECT PROCESS CONVID
        SYNCLEVEL(2)
    SEND INVITE WAIT CONVID
                              -- FMH5+Daten ----->
                                                  INIT PU ENDTA=O,SEND=Y
                                                  MGET    VGST=O,TAST=O
                                                  MPUT NE
                                                  PEND KP
                              <- Daten -----------
    RECEIVE CONVID
    SYNCPOINT
                              -- RequestCommit -->
                                                  INIT PU ENDTA=R,SEND=N
                                                  MGET    VGST=O,TAST=P
                                                  PEND RE
                              <- Committed -------
    =NORMAL
    SEND Client
<---
--->
    RECEIVE Client
    SEND LAST CONVID
    SYNCPOINT
                              -- Daten+
                                 RequestCommit -->
                                                  INIT PU ENDTA=F,SEND=N
                                                  MGET    VGST=O,TAST=P
                                                  PEND FI
                              <- Committed -------
    =NORMAL
    SEND Client
<---
    RETURN
```

Diese Variante unterscheidet sich von der vorhergehenden dadurch, dass die Antwortnachricht des UTM-Anwendungsprogramms noch innerhalb der ersten Transaktion gesendet wird. Bei der ersten Variante wird die Antwortnachricht in der zweiten Transaktion gesendet. Die zweite Variante ist aufwendiger zu programmieren und erzeugt auch mehr Nachrichten als die erste Variante. Sie hat aber den Vorteil, dass eine eventuell mit dem SYNCPOINT gekoppelte Datenbankänderung erst nach Erhalt der Nachricht vom UTM-Anwendungsprogramm festgeschrieben wird.

5. PEND ER/FR im UTM-Anwendungsprogramm

```
--->
    RECEIVE Client
    ALLOCATE
    CONNECT PROCESS CONVID
        SYNCLEVEL(2)
    SEND LAST CONVID
    SYNCPOINT
                              -- FMH5+Daten+
                                 RequestCommit -->
                                                  INIT PU ENDTA=F,SEND=N
                                                  MGET   VGST=O,TAST=P
                                                  PEND ER/FR
                              <- FMH7 ------------
                              -- FMH7-Response -->
    Abend ASP3
<---
```

Im obigen Fall führt ein PEND ER oder PEND FR von UTM dazu, dass das CICS-Auftraggeberprogramm mit Abend ASP3 abgebrochen wird. Dieser Abend kann vom CICS-Anwendungsprogramm nicht abgefangen werden. Soll dies vermieden werden, so muss im CICS-Anwendungsprogramm das Kommando ISSUE PREPARE verwendet werden:

```
--->
    RECEIVE Client
    ALLOCATE
    CONNECT PROCESS CONVID
        SYNCLEVEL(2)
    SEND LAST CONVID
    ISSUE PREPARE CONVID
                              -- FMH5+Daten+
                                 Prepare -------->
                                                  INIT PU ENDTA=F,SEND=N
                                                  MGET    VGST=O,TAST=P
                                                  PEND ER/FR
                              <- FMH7 ------------
                              -- FMH7-Response -->
    =TERMERR
    SEND Client
<---
    RETURN
```

6. PEND RS im UTM-Anwendungsprogramm

```
--->
    RECEIVE Client
    ALLOCATE
    CONNECT PROCESS CONVID
        SYNCLEVEL(2)
    SEND CONVID
    SYNCPOINT
                              -- FMH5+Daten+
                                 RequestCommit -->
                                                    INIT PU ENDTA=F,SEND=N
                                                    MGET   VGST=O,TAST=P
                                                    PEND RE
                              <- Committed -------
    =NORMAL
    SEND Client
<---
--->
    RECEIVE Client
    SEND LAST CONVID
    SYNCPOINT
                              -- Daten+
                                 RequestCommit -->
                                                    INIT PU ENDTA=F,SEND=N
                                                    MGET    VGST=O,TAST=P
                                                    MPUT RM KCRN=<
                                                    PEND RS
                              <- FMH7 ------------
                              -- FMH7-Response -->
    =ROLLEDBACK
    SEND Client
<---
--->
    RECEIVE Client
    SEND LAST CONVID
    SYNCPOINT
                              -- Daten+
                                 RequestCommit -->
                                                    INIT PU ENDTA=F,SEND=N
                                                    MGET NT KCRN=<
                                                    MGET    VGST=O,TAST=P
                                                    PEND FI
                              <- Committed -------
    =NORMAL
    SEND Client
<---
    RETURN
```

Ein PEND RS im UTM-Anwendungsprogramm verhält sich jedoch nur dann so wie im Beispiel beschrieben, wenn der Vorgang schon vorher einen Sicherungspunkt erreicht hat. Ein PEND RS im ersten Dialogschritt hat dieselbe Wirkung wie ein PEND FR.

7. Einschrittvorgang ohne Transaktionssicherung

```
--->
    RECEIVE Client
    ALLOCATE
    CONNECT PROCESS CONVID
        SYNCLEVEL(0)
    SEND INVITE WAIT CONVID
                            -- FMH5+Daten ----->
                                                INIT PU ENDTA=O,SEND=Y
                                                MGET    VGST=O,TAST=U
                                                MPUT NE
                                                PEND FI
                            <- Daten+CEB -------
    RECEIVE CONVID
    SEND Client
<---
    RETURN
```

8. Verarbeitungsquittung ohne Transaktionssicherung

```
--->
    RECEIVE Client
    ALLOCATE
    CONNECT PROCESS CONVID
        SYNCLEVEL(1)
    SEND INVITE CONFIRM
        CONVID
                            -- FMH5+Daten ----->
                                                INIT PU ENDTA=O,SEND=Y
                                                MGET    VGST=O,TAST=U
                                                MGET    KCRMGT=H
                                                MPUT NE
                                                PEND FI
                            <- Daten+CEB -------
      =OK EIBERR=X'OO'
    RECEIVE CONVID
    SEND Client
<---
    RETURN
```

9. Negative Verarbeitungsquittung ohne Transaktionssicherungung

```
--->
    RECEIVE Client
    ALLOCATE
    CONNECT PROCESS CONVID
        SYNCLEVEL(1)
    SEND INVITE CONFIRM
        CONVID
                            -- FMH5+Daten ----->
                                                INIT PU ENDTA=O,SEND=Y
                                                MGET    VGST=O,TAST=U
                                                MGET    KCRMGT=H
                                                MPUT EM
                                                PEND FI
                            <- FMH7 ------------
    =OK EIBERR=X'FF'
    RECEIVE CONVID =EOC
    SEND Client
<---
    RETURN
```

Start eines UTM-Asynchron-Vorgangs aus einem CICS-Anwendungsprogramm

10. Mit Transaktionssicherung

```
--->
    RECEIVE Client
    ALLOCATE
    CONNECT PROCESS CONVID
        SYNCLEVEL(2)
    SEND LAST CONVID
    SYNCPOINT
                            -- FMH5+Daten+
                               RequestCommit -->
                            <- Committed -------
    =NORMAL
    SEND Client
<---
    RETURN
                                                  INIT
                                                  FGET
                                                  PEND FI
```

openUTM sendet die Quittungsnachricht sofort, nachdem es den Asynchron-Auftrag erhalten und gesichert hat. Je nach Verfügbarkeit von Systemressourcen kann es bis zur Ausführung des Asynchronauftrags einige Zeit dauern.

Die Verwendung von EXEC CICS START zum Starten eines Asynchron-Vorgangs ist bei LU6.2 nicht möglich.

11. Ohne Transaktionssicherung

```
--->
    RECEIVE Client
    ALLOCATE
    CONNECT PROCESS CONVID
        SYNCLEVEL(1)
    SEND LAST CONFIRM
        CONVID
                            -- FMH5+Daten ----->
                            <- Quittung --------
    =NORMAL
    SEND Client
<---
    RETURN
                                                  INIT
                                                  FGET
                                                  PEND FI
```

Der Verzicht auf Transaktionssicherung bringt für die Anwendungsprogramme folgende Nachteile:

– Das Absenden der Asynchron-Nachricht im CICS-Programm kann nicht ohne weiteres mit anderen Abläufen im Anwendungsprogramm gekoppelt werden. Es kann also z. B. vorkommen, dass die Asynchron-Nachricht trotz eines gescheiterten lokalen Datenbank-Updates an die UTM-Anwendung gesendet wird.

– Wenn die Quittung aufgrund eines Kommunikationsfehlers verloren geht, wird die Asynchron-Nachricht eventuell mehrmals an die UTM-Anwendung gesendet.

Start eines CICS-Dialogvorgangs aus einem UTM-Anwendungsprogramm

12. Einschrittvorgang mit Transaktionssicherung, erste Variante

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=C
    MPUT NE KCRN=>A
    PEND KP
                              -- FMH5+Daten ----->
                                                  RECEIVE STATE(SEND)
                                                  SEND
                              <- Daten -----------
    INIT
    MGET NT KCRN=>A
        VGST=0,TAST=0
    MPUT NE Client
    PEND FI
                              -- Prepare -------->
                                                  RECEIVE STATE(SYNCFREE)
                                                  SYNCPOINT
                              <- RequestCommit ---
    (UTM-Systemcode)
                              -- Committed ------>
                              <- Forget ----------
                                                  =NORMAL
                                                  RETURN
<---
```

13. Einschrittvorgang mit Transaktionssicherung, zweite Variante

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=C
    MPUT NE KCRN=>A
    PEND KP
                            -- FMH5+Daten ----->
                                                RECEIVE STATE(SEND)
                                                SEND
                            <- Daten -----------
    INIT
    MGET NT KCRN=>A
        VGST=O,TAST=O
    MPUT NE KCRN=>A KCLM=O
    CTRL PE KCRN=>A
    PEND KP
                            -- Prepare -------->
                                                RECEIVE STATE(SYNCFREE)
                                                SYNCPOINT
                            <- RequestCommit ---
    INIT
    MGET NT KCRN=>A
        VGST=C,TAST=P
    MPUT NE Client
    PEND FI
                            -- Committed ------>
                            <- Forget ----------
                                                =NORMAL
                                                RETURN
<---
```

Im Beispiel 13 sieht das 3. Teilprogramm, ob CICS Commit oder Rollback gegeben hat.

Im Beispiel 12 erfährt das Teilprogramm von Commit oder Rollback in CICS nichts mehr, dies erfolgt allein unter Kontrolle vom UTM-Systemcode.

14. Keine Daten vom CICS-Anwendungsprogramm im UTM-Anwendungsprogramm benötigt und mit Transaktionssicherung

Wenn es von der Ablauflogik her nicht erforderlich ist, dass das aufgerufene CICS-Anwendungsprogramm dem UTM-Anwendungsprogramm Daten zurücksendet, so kann die Kommunikation wie im folgenden Beispiel einfacher programmiert werden.

```
--->
    INIT
    MGET    Client
    APRO DM KCRN=>A KCOF=C
    MPUT NE KCRN=>A
    CTRL PE KCRN=>A
    PEND KP
                             -- FMH5+Daten+
                                Prepare -------->
                                                  RECEIVE STATE(SYNCFREE)
                                                  SYNCPOINT
                             <- RequestCommit ---
    INIT
    MGET NT KCRN=>A
        VGST=C,TAST=P
    MPUT NE Client
    PEND FI
                             -- Committed ------>
                             <- Forget ----------
                                                  =NORMAL
                                                  RETURN
<---
```

15. Zwei Transaktionen in einem Vorgang, erste Variante

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=C
    MPUT NE KCRN=>A
    PEND RE
                              -- FMH5+Daten+
                                 Prepare -------->
                                                  RECEIVE STATE(SYNCSEND)
                                                  SYNCPOINT
                              <- RequestCommit ---
                              -- Committed ------>
                              <- Forget ----------
                                                  =NORMAL
                                                  SEND
                              <- Daten -----------
    INIT
    MGET NT KCRN=>A
        VGST=O,TAST=O
    MPUT NE KCRN=>A
    CTRL PE KCRN=>A
    PEND KP
                              -- Daten+Prepare -->
                                                  RECEIVE STATE(SYNCFREE)
                                                  SYNCPOINT
                              <- RequestCommit ---
    INIT
    MGET NT KCRN=>A
        VGST=C,TAST=P
    MPUT NE Client
    PEND FI
                              -- Committed ------>
                              <- Forget ----------
                                                  =NORMAL
                                                  RETURN
<---
```

Wenn beim zweiten `Prepare` keine Daten gesendet werden, können analog zu den Beispielen 12 und 13 die folgenden UTM-Aufrufe entfallen:

```
MPUT NE KCRN=>A
CTRL PE KCRN=>A
PEND KP
INIT
MGET NT KCRN=>A VGST=C,TAST=P
```

16. Zwei Transaktionen in einem Vorgang, zweite Variante

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=C
    MPUT NE KCRN=>A
    PEND KP
                            -- FMH5+Daten ----->
                                                RECEIVE STATE(SEND)
                                                SEND
                            <- Daten -----------
    INIT
    MGET NT KCRN=>A
        VGST=O,TAST=O
    MPUT NE KCRN=>A KCLM=O
    CTRL PR KCRN=>A
    PEND KP
                            -- Prepare -------->
                                                RECEIVE STATE(SYNCRECEIVE)
                                                SYNCPOINT
                            <- RequestCommit ---
    INIT
    MGET NT KCRN=>A
        VGST=O,TAST=P
    MPUT NE Client
    PEND RE
                            -- Committed ------>
                            <- Forget ----------
                                                =NORMAL
<---
--->
    INIT
    MGET    Client
    MPUT NE KCRN=>A
    CTRL PE KCRN=>A
    PEND KP
                            -- Daten+Prepare -->
                                                RECEIVE STATE(SYNCFREE)
                                                SYNCPOINT
                            <- RequestCommit ---
```

Fortsetzung nächste Seite

Fortsetzung

```
    INIT
    MGET NT KCRN=>A
        VGST=C,TAST=P
    MPUT NE Client
    PEND FI
                            -- Committed ------>
                            <- Forget ----------
                                                =NORMAL
                                                RETURN
<---
```

Ähnlich wie in den Beispielen 12 und 13 können einige UTM-Aufrufe entfallen, wenn man im Teilprogramm auf die Rollback-/Commit-Information von CICS verzichtet. Der 2. und 3. Programmabschnitt reduziert sich dann auf die Folge

```
INIT
MGET NT KCRN=>A VGST=O,TAST=O
CTRL PR KCRN=>A
MPUT NE Client
PEND RE
```

Diese Variante unterscheidet sich von der vorhergehenden dadurch, dass die Antwortnachricht des CICS-Anwendungsprogramms noch innerhalb der ersten Transaktion gesendet wird. Bei der ersten Variante wird die Antwortnachricht in der zweiten Transaktion gesendet. Die zweite Variante ist aufwändiger zu programmieren und erzeugt auch mehr Nachrichten als die erste Variante. Sie hat aber den Vorteil, dass eine eventuell mit dem PEND RE gekoppelte Datenbankänderung erst nach Erhalt der Nachricht vom CICS-Anwendungsprogramm festgeschrieben wird.

17. CICS-Anwendungsprogramm setzt Transaktion zurück und bricht Vorgang ab, erste Variante

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=C
    MPUT NE KCRN=>A
    PEND KP
                            -- FMH5+Daten ----->
                                                RECEIVE STATE(SEND)
                                                SYNCPOINT ROLLBACK
                                                ISSUE ABEND
                                                RETURN
                            <- FMH7 ------------
                            -- FMH7-Response -->
    Rücksetzen der Transaktion
    evtl. mit Meldung KO34 auf Client
    Falls zuvor schon ein Sicherungspunkt
    erreicht war, dann Start des
    Folgeprogramms zu diesem Sicherungspunkt:
    INIT    KCKNZVG=R
    MGET NT KCRN=>A
        VGST=O,TAST=R
    ...
```

18. CICS-Anwendungsprogramm setzt Transaktion zurück und bricht Vorgang ab, zweite Variante

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=C
    MPUT NE KCRN=>A
    PGWT KP
                            -- FMH5+Daten ----->
                                                RECEIVE STATE(SEND)
                                                SYNCPOINT ROLLBACK
                                                ISSUE ABEND
                                                RETURN
                            <- FMH7 ------------
                            -- FMH7-Response -->
        KCTARB=Y
    MGET NT KCRN=>A
        VGST=T,TAST=R
    MPUT    Client
    PEND KCOM=FR
...
```

Im Gegensatz zu Beispiel 17 erhält das Teilprogramm im Beispiel 18 nach dem Rücksetzen durch CICS eine Rückmeldung zur Kontrolle.

19. CICS-Anwendungsprogramm setzt Transaktion zurück, Vorgang läuft weiter

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=C
    MPUT NE KCRN=>A
    PEND KP
                            -- FMH5+Daten ----->
                                                RECEIVE STATE(SEND)
                                                SEND
                            <- Daten -----------
    INIT
    MGET NT KCRN=>A
        VGST=O,TAST=O
    MPUT NE KCRN=>A KCLM=O
    CTRL PR KCRN=>A
    PEND KP
                            -- Prepare -------->
                                                RECEIVE STATE(SYNCRECEIVE)
                                                SYNCPOINT
                            <- RequestCommit ---
    INIT
    MGET NT KCRN=>A
        VGST=O,TAST=P
    MPUT NE Client
    PEND RE
                            -- Committed ------>
                            <- Forget ----------
                                                =NORMAL
<---
--->
    INIT
    MGET    Client
    MPUT NE KCRN=>A
    CTRL PE KCRN=>A
    PEND KP
                            -- Prepare -------->
                                                RECEIVE STATE(SYNCFREE)
                                                SYNCPOINT ROLLBACK
                            <- FMH7 ------------
                            -- FMH7-Response -->
                                                =NORMAL
<--- K034-Meldung
--->
```

Fortsetzung nächste Seite

Fortsetzung

```
    INIT    KCKNZVG=R
    MGET    Client
    MPUT NE KCRN=>A
    CTRL PE KCRN=>A
    PEND KP
                            -- Prepare -------->
                                                RECEIVE STATE(SYNCFREE)
                                                SYNCPOINT
                            <- RequestCommit ---
    INIT
    MGET NT KCRN=>A
        VGST=C,TAST=P
    MPUT NE Client
    PEND FI
                            -- Committed ------>
                            <- Forget ----------
                                                =NORMAL
                                                RETURN
<---
```

Es ist zu beachten, dass das UTM-Anwendungsprogramm in diesem Fall nach dem SYNCPOINT ROLLBACK keine Statusnachricht vom Auftragnehmer erhält. openUTM stellt dem Anwendungsprogramm nur dann eine Statusnachricht vom Autragnehmer zu, wenn der Auftragnehmer durch das Rücksetzen der Transaktion beendet wurde.

Neben der K034-Meldung wird die Nachricht an den Client vom letzten Sicherungspunkt nochmal an den Client gesendet.

Ähnlich wie in den Beispielen 12 und 13 können einige UTM-Aufrufe entfallen, wenn man im Teilprogramm auf die Rollback-/Commit-Information von CICS verzichtet. Der 2. und 3. Programmabschnitt reduziert sich dann auf die Folge

```
INIT
MGET NT KCRN=>A VGST=O,TAST=O
CTRL PR KCRN=>A
MPUT NE Client
PEND KCOM=RE
```

20. Einschrittvorgang ohne Transaktionssicherung

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=B
    MPUT NE KCRN=>A
    PEND KP
                            -- FMH5+Daten ----->
                                                RECEIVE STATE(SEND)
                                                SEND LAST WAIT
                                                RETURN
                            <- Daten+CEB -------
    INIT
    MGET NT KCRN=>A
        VGST=C,TAST=U
    MPUT NE Client
    PEND FI
<---
```

21. Verarbeitungsquittung ohne Transaktionssicherung

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=H
    MPUT NE KCRN=>A
    MPUT HM KCRN=>A
    PEND KP
                            -- FMH5+Daten ----->
                                                RECEIVE STATE(CONFSEND)
                                                ISSUE CONFIRMATION
                                                SEND LAST WAIT
                                                RETURN
                            <- Daten+CEB -------
    INIT
    MGET NT KCRN=>A
        KCRMGT=C
    MGET NT KCRN=>A
        VGST=C,TAST=U
    MPUT NE Client
    PEND FI
<---
```

22. Negative Verarbeitungsquittung ohne Transaktionssicherung

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=H
    MPUT NE KCRN=>A
    MPUT HM KCRN=>A
    PEND KP
                            -- FMH5+Daten ----->
                                                RECEIVE STATE(CONFSEND)
                                                ISSUE ERROR
                                                SEND LAST WAIT
                                                RETURN
                            <- FMH7 ------------
    INIT
    MGET NT KCRN=>A
        KCRMGT=E
    MGET NT KCRN=>A
        VGST=C,TAST=U
    MPUT NE Client
    PEND FI
<---
```

Start eines CICS-Asynchron-Vorgangs aus einem UTM-Anwendungsprogramm

23. Start eines CICS-Asynchron-Vorgangs aus einem UTM-Anwendungsprogramm ohne Transaktionssicherung

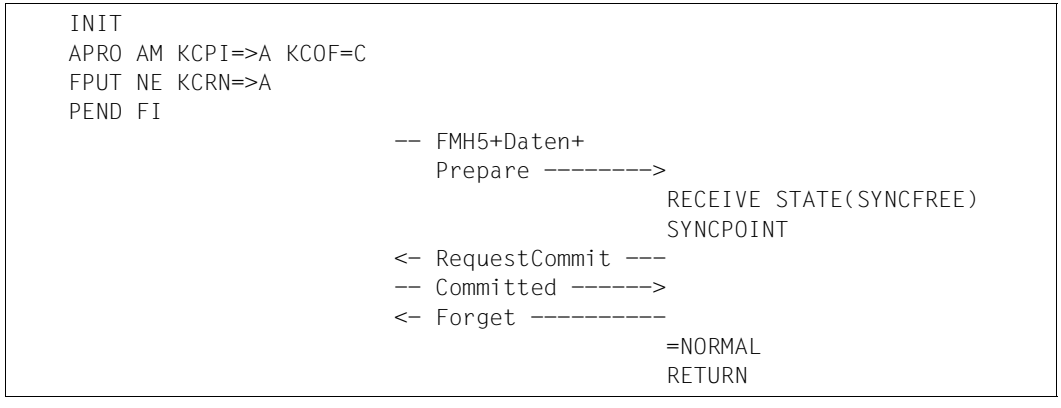
```
    INIT
    APRO AM KCPI=>A KCOF=B
    FPUT NE KCRN=>A
    PEND FI
                            -- FMH5+Daten ----->
                                                RECEIVE STATE(RECEIVE)
                                                RECEIVE STATE(CONFFREE)
                                                ISSUE CONFIRMATION
                                                RETURN
                            <- Daten+CEB -------
```

Für diese Art der Kommunikation muss eine openUTM-LU62-Instanz verwendet werden, die mit dem Parameter UTM-ASYNC=YES generiert ist.

Der in Beispiel 20 aufgeführte Fall ist vom Ablauf her ähnlich. Der wesentliche Unterschied besteht darin, dass das CICS-Programm in Beispiel 20 noch Daten zurücksenden kann. Auf UTM-Seite hat der Dialog-Fall den Vorteil, dass die Anwendung über Erfolg oder Misserfolg des Aufrufs informiert wird.

Sendet das CICS-Programm eine negative Verarbeitungsquittung mittels ISSUE ERROR, so löscht openUTM den Auftrag. Das UTM-Anwendungsprogramm kann jedoch selbstverständlich nicht mehr benachrichtigt werden.

24. Start eines CICS-Asynchron-Vorgangs aus einem UTM-Anwendungsprogramm mit Transaktionssicherung



Setzt das CICS-Programm die Transaktion zurück mittels SYNCPOINT ROLLBACK, so löscht openUTM den Auftrag. Das UTM-Anwendungsprogramm kann jedoch selbstverständlich nicht mehr benachrichtigt werden.

Die Transaktionssicherung im Beispiel 24 bringt für die Anwendungsprogramme gegenüber dem Ablauf ohne Transaktionssicherung (Beispiel 23) folgende Vorteile:

– Das Absenden der Asynchron-Nachricht im UTM-Anwendungsprogramm wird automatisch mit anderen Abläufen im Anwendungsprogramm, z. B. mit einem Datenbank-Update gekoppelt.

– Wenn der PEND FI erfolgreich ist, wird der CICS-Asynchron-Vorgang garantiert genau einmal gestartet.

Dieser Fall ist ähnlich zu Beispiel 14. Die CICS-Programmierung ist bei beiden Beispielen identisch. Auf UTM-Seite hat der Dialog-Fall (Beispiel 14) den Vorteil, dass die Anwendung über Erfolg oder Misserfolg des Aufrufs informiert wird. Der asynchrone Fall (Beispiel 24) eignet sich vor allem für zeitgesteuerte Aufträge.

4.2.6 Distributed Program Link

In CICS gibt es die Möglichkeit, Programme auf verschiedenen Systemen mittels Distributed Program Link (DPL) zu verbinden. Mit dieser Technik kann ein CICS-Programm auf einem System ein CICS-Programm auf einem anderen System als Unterprogramm aufrufen. Es verwendet dazu den CICS-Aufruf EXEC CICS LINK. Distributed Program Link ist zwischen CICS und openUTM nicht möglich. In einer mehrstufigen Hierarchie können jedoch auch UTM-Anwendungsprogramm eingesetzt werden. Die folgenden beiden Beispiele zeigen eine derartige Kommunikation. Dabei zeigt das erste Beispiel die Verwendung von EXEC CICS LINK zwischen 2 CICS-Systemen. Im zweiten Beispiel wird der Distributed Program Link von einem System mit CICS-Client aufgerufen. Bei CICS-Client wird die Programmschnittstelle „External Call Interface“ (ECI) benutzt.

1. Distributed Program Link zwischen zwei CICS-Systemen

```
    CICS-System 1         CICS-System 2         openUTM
--->
    RECEIVE Client
    LINK SYSID(...)
        TRANSID(...)
              ---------->
                         ALLOCATE
                         CONNECT PROCESS CONVID
                             SYNCLEVEL(2)
                         SEND INVITE WAIT
                             CONVID
                                   ---------->
                                              INIT PU ENDTA=O,SEND=Y
                                              MGET    VGST=O,TAST=O
                                              MPUT NE
                                              PEND KP
                                   <----------
                         =NORMAL
                         RECEIVE CONVID
                         RETURN
              <----------
    =NORMAL
    SYNCPOINT
              ---------->
                                   ---------->
                                              INIT PU ENDTA=R,SEND=N
                                              MGET    KCRLM=O
                                                      VGST=O,TAST=P
                                              PEND RE
                                   <----------
              <----------
              ---------->
                                   ---------->
                                              INIT PU ENDTA=F,SEND=N
                                              MGET    KCRLM=O
                                                      VGST=O,TAST=P
                                              PEND FI
                                   <----------
              <----------
    =NORMAL
    SEND Client
<---
    RETURN
```

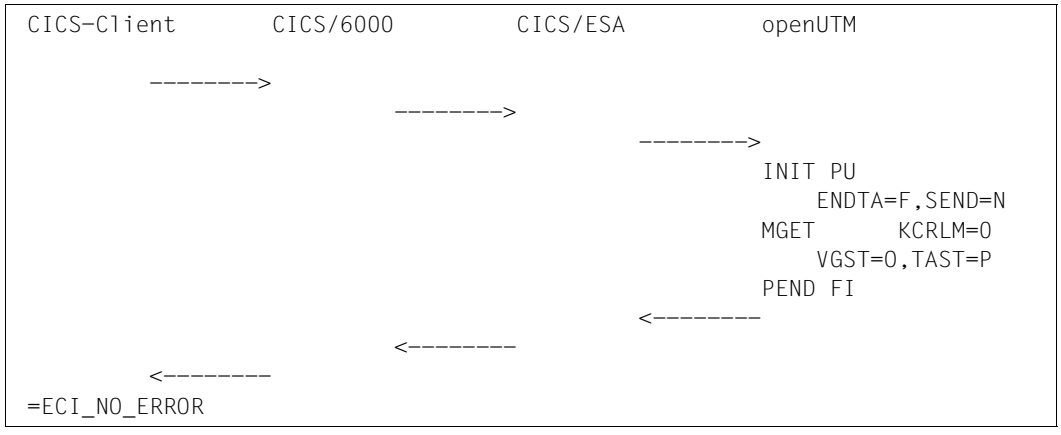
2. Distributed Program Link von CICS-Client

```
CICS-Client       CICS/6000         CICS/ESA          openUTM

CICS_ExternalCall
    (program)
         -------->
                  LINK SYSID(...)
                      TRANSID(...)
                           -------->
                                    ALLOCATE
                                     CONNECT PROCESS
                                        CONVID
                                        SYNCLEVEL(2)
                                    SEND INVITE WAIT
                                        CONVID
                                             -------->
                                                      INIT PU
                                                          ENDTA=O,SEND=Y
                                                      MGET
                                                          VGST=O,TAST=O
                                                      MPUT NE
                                                      PEND KP
                                             <--------
                                    =NORMAL
                                    RECEIVE CONVID
                                    RETURN
                           <--------
                  =NORMAL
         <--------
=ECI_NO_ERROR
CICS_ExternalCall
    (ECI_COMMIT)
         -------->
                           -------->
                                             -------->
                                                      INIT PU
                                                          ENDTA=R,SEND=N
                                                      MGET      KCRLM=O
                                                          VGST=O,TAST=P
                                                      PEND RE
                                             <--------
                           <--------
         <--------
```

Fortsetzung nächste Seite

Fortsetzung



4.2.7 Hinweise zur openUTM-Programmierung

Folgende Punkte sind bei der Programmierung von UTM-Anwendungen zu berücksichtigen, wenn diese Anwendungen mit LU6.2-Partnern kommunizieren wollen:

1. Fordert ein UTM-Anwendungsprogramm als Auftraggeber den LU6.2-Partner mittels CTRL PR oder CTRL PE auf, das Transaktionsende einzuleiten, so kann es danach keine Daten mehr vom Partner empfangen, und zwar unabhängig davon, ob es vor dem CTRL einen MPUT aufgerufen hat oder nicht. Diese Einschränkung erfordert in vielen Fällen einen zusätzlichen Dialogschritt.
2. Benutzerdaten können nur im UDT-Format transportiert werden. Dies bedeutet, dass bei MPUT, FPUT, DPUT an einen LU6.2-Partner das Feld KCMF immer mit Leerzeichen versorgt sein muss.
3. Benutzerdaten können bei Verwendung von TRANSIT nur bis zu einer Länge von 32763 Byte von UTM an CICS gesendet werden. KCLM bei MPUT; FPUT; DPUT an einen LU6.2-Partner darf also maximal den Wert 32763 annehmen.
4. Wird beim APRO KCSECTYP=P angegeben, d.h. will das UTM-Anwendungsprogramm dem Partner eine Benutzerkennung senden, so muss auch ein Passwort angegeben werden, d.h. KCPWDLTH muss größer als 0 sein. Benutzerkennung und Passwort dürfen jeweils maximal 10 byte lang sein.
5. Wenn openUTM-LU62 eine von einem UTM-Anwendungsprogramm gewünschte Verbindung zu einem CICS-Transaktionscode nicht aufbauen kann, so erhält die UTM-Anwendung keine detaillierten Ablehnungsgründe. Die Information in der SYSLOG-Datei von openUTM reicht also meist nicht zur Diagnose aus. Detaillierte Ablehnungsgründe sind nur in der Protokolldatei von openUTM-LU62 zu finden.
6. openUTM bietet die Möglichkeit, bei der LTAC-Generierung eine maximale Wartezeit für das Belegen einer Session zum Partner (zeit1) und eine maximale Wartezeit für das Eintreffen einer Antwort vom Auftragnehmer (zeit2) zu generieren. Standardmäßig ist zeit1 auf 30 Sekunden, zeit2 auf unendlich gestellt. Bei einer openUTM-CICS-Kopplung wird mit zeit1 nur die Wartezeit für die Belegung einer Verbindung zwischen openUTM und openUTM-LU62 überwacht. Ist kein Verbindungsaufbau von openUTM-LU62 zu CICS möglich, so muss die Wartezeit auf das Belegen dieser Verbindung entweder in openUTM-LU62 (mittels ALLOC-TIME) oder über zeit2 begrenzt werden.

4.3 Verwendung der Programmschnittstelle CPI-C

In CICS kann für die LU6.2-Kommunikation auch die Programmschnittstelle CPI-C benutzt werden anstelle der CICS-Kommandos. Da bei Kopplungen zwischen openUTM und LU6.2-Partnern ungleich CICS der LU6.2-Partner oft mit der Programmschnittstelle CPI-C realisiert ist, sollen im folgenden die obigen Beispiele der openUTM-CICS-Kommunikation auf die Programmschnittstelle CPI-C übertragen werden.
Beispiele für derartige Partner sind APPC/MVS, IMS, System i5 (früher AS/400).

Auch in openUTM kann CPI-C anstelle von KDCS für die OSI-TP-Kommunikation benutzt werden. Wenn beide Partner CPI-C benutzen, so ist dies sehr ähnlich einer openUTM-openUTM-Kopplung mit CPI-C auf beiden Seiten oder einer openUTM-OpenCPIC-Kopplung. Beides ist ausführlich in dem openUTM-Handbuch „Anwendungen erstellen mit X/Open-Schnittstellen“ beschrieben.

Soll eine transaktionsgesicherte Kommunikation über OSI-TP oder LU6.2 mit CPI-C programmiert werden, so benötigt man zusätzlich zu CPI-C noch eine Programmschnittstelle zum Anfordern und Bestätigen eines Sicherungspunkts sowie zum Zurücksetzen einer Transaktion. In openUTM und bei X/Open wird hierzu die Programmschnittstelle TX verwendet. Bei IBM-Produkten ist dagegen meist die Programmschnittstelle CPI-RR (Common Programming Interface for Resource Recovery) mit den Aufrufen SRRCMIT und SRRBACK anzutreffen. Deshalb sind die obigen openUTM-CICS-Beispiele im folgenden nochmals mit CPI-C mit CPI-RR formuliert.

4.3.1 Vergleich mit KDCS-Aufrufen

Im folgenden soll versucht werden, die CICS-Kommandos in ihrer Semantik mit KDCS-Aufrufen zu vergleichen.

Adressierung eines fernen Vorgangs mit Transaktionssicherung

```
CMINIT                                  APRO DM/AM KCOF=C
CMSSL (CM_SYNC_POINT)
CMALLC
```

Adressierung eines fernen Vorgangs ohne Transaktionssicherung

```
CMINIT                              APRO DM KCOF=H
CMSSL (CM_CONFIRM)
CMALLC
```

bzw.

```
CMINIT                              APRO DM/AM KCOF=B
CMSSL (CM_NONE)
CMALLC
```

Entgegennahme einer Nachricht beim Auftragnehmer

```
CMACCP                              INIT oder INIT PU
CMRCV                               MGET
```

Nachrichtenaustausch ohne Sicherungspunkt

```
CMSEND                              MPUT NE
CMRCV                               PEND KP
                                    .
                                    INIT
                                    MGET
```

Senden einer Nachricht und Anforderung zum Transaktionsende

```
CMSEND                              MPUT NE KCRN=“>...“
SRRCMIT                             CTRL PR KCRN=“>...“
                                    PEND KP
                                    .
                                    INIT
                                    MGET NT KCRN=“>...“
```

Senden einer Nachricht und Anforderung Transaktions- und Vorgangsende

```
CMSEND                              MPUT NE KCRN=“>...“
CMDEAL                              CTRL PE KCRN=“>...“
SRRCMIT                             PEND KP
                                    .
                                    INIT
                                    MGET NT KCRN=“>...“
```

Nachrichtenaustausch und Anforderung zum Transaktionsende

```
CMSST (CM_SEND_AND_PREP_TO_RECEIVE)  MPUT NE KCRN=“>...“
CMSEND                               PEND RE
SRRCMIT                              .
CMRCV                                INIT
                                     MGET NT KCRN=“>...“
```

Rücksetzen der Transaktion

```
SRRBACK                              MPUT RM KCRN=“<...“
                                     PEND RS
                                     .
                                     INIT
                                     MGET NT KCRN=“<...“
```

Rücksetzen der Transaktion und Abbruch des Vorgangs

```
SRRBACK                              PEND ER/FR
CMSDT (CM_DEALLOCATE_ABEND)
CMDEAL
```

Nachrichtenaustausch mit Verarbeitungsquittung

```
CMSPTR(PREP_TO_RECEIVE_CONFIRM)      MPUT NE KCRN=“>...“
CMSST (CM_SEND_AND_PREP_TO_RECEIVE)  MPUT HM KCRN=“>...“
CMSEND                               PEND KP
CMRCV                                .
                                     INIT
                                     MGET NT KCRN=“>...“
```

Senden einer Antwort mit positiver Verarbeitunsquittung

```
CMCFMD                               MPUT NE
CMSEND                               PEND FI
```

Senden einer Antwort mit negativer Verarbeitunsquittung

```
CMSERR                               MPUT EM
```

Abbruch des Dialogs zum Partner

```
CMSDT (CM_DEALLOCATE_ABEND)          CTRL AB
CMDEAL
```

4.3.2 Beispiele für openUTM-CPI-C-Kommunikation

Start eines openUTM-Dialogvorgangs aus einem CPI-C-Anwendungsprogramm

1. Einschrittvorgang mit Transaktionssicherung

```
CMINIT
CMSSL (CM_SYNC_POINT)
CMALLC
CMSEND
CMRCV
                        -- FMH5+Daten ----->
                                          INIT PU ENDTA=O,SEND=Y
                                          MGET    VGST=O,TAST=O
                                          MPUT NE
                                          PEND KP
                        <- Daten -----------
=CM_OK
CMDEAL
SRRCMIT
                        -- Daten+
                           RequestCommit -->
                                          INIT PU ENDTA=F,SEND=N
                                          MGET    VGST=O,TAST=P
                                          PEND FI
                        <- Committed -------
=RR_OK
```

2. Keine Daten vom UTM-Anwendungsprogramm im CPI-C-Anwendungsprogramm benötigt und mit Transaktionssicherung.

```
CMINIT
CMSSL (CM_SYNC_POINT)
CMALLC
CMSEND
CMDEAL
SRRCMIT
                        -- FMH5+Daten+
                           RequestCommit -->
                                          INIT PU ENDTA=F,SEND=N
                                          MGET    VGST=O,TAST=P
                                          PEND FI
                        <- Committed -------
=RR_OK
```

3. Zwei Transaktionen in einem Vorgang, erste Variante

```
CMINIT
CMSSL (CM_SYNC_POINT)
CMALLC
CMSST (CM_SEND_AND_PREP_TO_RECEIVE)
CMSEND
SRRCMIT
                          -- FMH5+Daten+
                             RequestCommit -->
                                              INIT PU ENDTA=C,SEND=Y
                                              MGET    VGST=O,TAST=P
                                              MPUT NE
                                              PEND RE
                          <- Committed+Daten -
=RR_OK
CMDEAL
SRRCMIT
                          -- Daten+
                             RequestCommit -->
                                              INIT PU ENDTA=F,SEND=N
                                              MGET    VGST=O,TAST=P
                                              PEND FI
                          <- Committed -------
=RR_OK
```

4. Zwei Transaktionen in einem Vorgang, zweite Variante

```
    CMINIT
    CMSSL (CM_SYNC_POINT)
    CMALLC
    CMSEND
    CMRCV
                              -- FMH5+Daten ----->
                                                  INIT PU ENDTA=O,SEND=Y
                                                  MGET    VGST=O,TAST=O
                                                  MPUT NE
                                                  PEND KP
                              <- Daten -----------
    =CM_OK
    SRRCMIT
                              -- RequestCommit -->
                                                  INIT PU ENDTA=R,SEND=N
                                                  MGET    VGST=O,TAST=P
                                                  PEND RE
                              <- Committed -------
    =RR_OK
    CMSEND
    CMDEAL
    SRRCMIT
                              -- Daten+
                                 RequestCommit -->
                                                  INIT PU ENDTA=F,SEND=N
                                                  MGET    VGST=O,TAST=P
                                                  PEND FI
                              <- Committed -------
    =RR_OK
```

5. PEND ER/FR im UTM-Anwendungsprogramm

```
    CMINIT
    CMSSL (CM_SYNC_POINT)
    CMALLC
    CMSEND
    CMDEAL
    SRRCMIT
                              -- FMH5+Daten+
                                 RequestCommit -->
                                                    INIT PU ENDTA=F,SEND=N
                                                    MGET   VGST=O,TAST=P
                                                    PEND ER/FR
                              <- FMH7 ------------
                              -- FMH7-Response -->
    =RR_BACKED_OUT
```

6. PEND RS im UTM-Anwendungsprogramm

```
    CMINIT
    CMSSL (CM_SYNC_POINT)
    CMALLC
    CMSEND
    SRRCMIT
                              -- FMH5+Daten+
                                 RequestCommit -->
                                                    INIT PU ENDTA=R,SEND=N
                                                    MGET   VGST=O,TAST=P
                                                    PEND RE
                              <- Committed -------
    =RR_OK
    CMSEND
    CMDEAL
    SRRCMIT
                              -- Daten+
                                 RequestCommit -->
                                                    INIT PU ENDTA=F,SEND=N
                                                    MGET    VGST=O,TAST=P
                                                    MPUT RM KCRN=<
                                                    PEND RS
                              <- FMH7 ------------
                              -- FMH7-Response -->
    =RR_BACKED_OUT
    CMSEND
    CMDEAL
    SRRCMIT
                              -- Daten+
                                 RequestCommit -->
                                                    INIT PU ENDTA=F,SEND=N
                                                    MGET NT KCRN=<
                                                    MGET    VGST=O,TAST=P
                                                    PEND FI
                              <- Committed -------
    =RR_OK
```

7. Einschrittvorgang ohne Transaktionssicherung

```
CMINIT
CMALLC
CMSEND
CMRCV
                          -- FMH5+Daten ----->
                                              INIT PU ENDTA=O,SEND=Y
                                              MGET    VGST=O,TAST=U
                                              MPUT NE
                                              PEND FI
                          <- Daten+CEB -------
=CM_DEALLOCATED_NORMAL
```

8. Verarbeitungsquittung ohne Transaktionssicherung

```
CMINIT
CMSSL (CM_CONFIRM)
CMALLC
CMSST (CM_SEND_AND_PREP_TO_RECEIVE)
CMSEND
                          -- FMH5+Daten ----->
                                              INIT PU ENDTA=O,SEND=Y
                                              MGET    VGST=O,TAST=U
                                              MGET    KCRMGT=H
                                              MPUT NE
                                              PEND FI
                          <- Daten+CEB -------
=CM_OK
CMRCV
=CM_DEALLOCATED_NORMAL
```

9. Negative Verarbeitungsquittung ohne Transaktionssicherung

```
CMINIT
CMSSL (CM_CONFIRM)
CMALLC
CMSST (CM_SEND_AND_PREP_TO_RECEIVE)
CMSEND
                          -- FMH5+Daten ----->
                                               INIT PU ENDTA=O,SEND=Y
                                               MGET    VGST=O,TAST=U
                                               MGET    KCRMGT=H
                                               MPUT EM
                                               PEND FI
                          <- FMH7 ------------
=CM_PROGRAM_ERROR_PURGING
CMRCV
=CM_DEALLOCATED_NORMAL
```

Start eines UTM-Asynchron-Vorgangs aus einem CPI-C-Anwendungsprogramm

10. Mit Transaktionssicherung

```
CMINIT
CMSSL (CM_SYNC_POINT)
CMALLC
CMSEND
CMDEAL
SRRCMIT
                          -- FMH5+Daten+
                             RequestCommit -->
                          <- Committed -------
=RR_OK
                                               INIT
                                               FGET
                                               PEND FI
```

openUTM sendet die Quittungsnachricht sofort, nachdem es den Asynchron-Auftrag erhalten und gesichert hat. Je nach Verfügbarkeit von Systemressourcen kann es bis zur Ausführung des Asynchronauftrags einige Zeit dauern.

11. Ohne Transaktionssicherung

```
    CMINIT
    CMSSL (CM_CONFIRM)
    CMALLC
    CMSEND
    CMDEAL
                            -- FMH5+Daten ----->
                            <- Quittung --------
    =CM_OK
                                                  INIT
                                                  FGET
                                                  PEND FI
```

openUTM sendet die Quittungsnachricht sofort, nachdem es den Asynchron-Auftrag erhalten und gesichert hat. Je nach Verfügbarkeit von Systemressourcen kann es bis zur Ausführung des Asynchronauftrags einige Zeit dauern.

Zu den Vorteilen der Transaktionssicherung bei Asynchron-Vorgängen siehe Seite 132.

Start eines CPI-C-Dialogvorgangs aus einem UTM-Anwendungsprogramm

12. Einschrittvorgang mit Transaktionssicherung, erste Variante

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=C
    MPUT NE KCRN=>A
    PEND KP
                            -- FMH5+Daten ----->
                                                CMACCP
                                                CMRCV
                                                -> CM_SEND_RECEIVED
                                                CMSEND
                            <- Daten -----------
    INIT
    MGET NT KCRN=>A
        VGST=O,TAST=O
    MPUT NE Client
    PEND FI
                            -- Prepare -------->
                                                CMRCV
                                                -> CM_TAKE_COMMIT_
                                                      DEALLOCATE
                                                SRRCMIT
                            <- RequestCommit ---
    (UTM-Systemcode)
                            -- Committed ------>
                            <- Forget ----------
                                                =RR_OK
<---
```

13. Einschrittvorgang mit Transaktionssicherung, zweite Variante

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=C
    MPUT NE KCRN=>A
    PEND KP
                            -- FMH5+Daten ----->
                                                CMACCP
                                                CMRCV
                                                -> CM_SEND_RECEIVED
                                                CMSEND
                            <- Daten -----------
    INIT
    MGET NT KCRN=>A
        VGST=O,TAST=O
    MPUT NE KCRN=>A KCLM=O
    CTRL PE KCRN=>A
    PEND KP
                            -- Prepare -------->
                                                CMRCV
                                                -> CM_TAKE_COMMIT_
                                                      DEALLOCATE
                                                SRRCMIT
                            <- RequestCommit ---
    INIT
    MGET NT KCRN=>A
        VGST=C,TAST=P
    MPUT NE Client
    PEND FI
                            -- Committed ------>
                            <- Forget ----------
                                                =RR_OK
<---
```

Im Beispiel 13 sieht das 3. Teilprogramm, ob CPI-C Commit oder Rollback gegeben hat.

Im Beispiel 12 erfährt das Teilprogramm von Commit oder Rollback in CPI-C nichts mehr, dies erfolgt allein unter Kontrolle vom UTM-Systemcode.

14. Keine Daten vom CPI-C-Anwendungsprogramm im UTM-Anwendungsprogramm benötigt und mit Transaktionssicherung.

```
--->
    INIT
    MGET    Client
    APRO DM KCRN=>A KCOF=C
    MPUT NE KCRN=>A
    CTRL PE KCRN=>A
    PEND KP
                              -- FMH5+Daten+
                                 Prepare -------->
                                                  CMACCP
                                                  CMRCV
                                                  -> CM_TAKE_COMMIT_
                                                        DEALLOCATE
                                                  SRRCMIT
                              <- RequestCommit ---
    INIT
    MGET NT KCRN=>A
        VGST=C,TAST=P
    MPUT NE Client
    PEND FI
                              -- Committed ------>
                              <- Forget ----------
                                                  =RR_OK
<---
```

15. Zwei Transaktionen in einem Vorgang, erste Variante

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=C
    MPUT NE KCRN=>A
    PEND RE
                              -- FMH5+Daten+
                                 Prepare -------->
                                                  CMACCP
                                                  CMRCV
                                                  -> CM_TAKE_COMMIT_SEND
                                                  SRRCMIT
                              <- RequestCommit ---
                              -- Committed ------>
                              <- Forget ----------
                                                  =RR_OK
                                                  CMSEND
                              <- Daten -----------
    INIT
    MGET NT KCRN=>A
        VGST=O,TAST=O
    MPUT NE KCRN=>A
    CTRL PE KCRN=>A
    PEND KP
                              -- Daten+Prepare -->
                                                  CMRCV
                                                  -> CM_TAKE_COMMIT_
                                                        DEALLOCATE
                                                  SRRCMIT
                              <- RequestCommit ---
    INIT
    MGET NT KCRN=>A
        VGST=C,TAST=P
    MPUT NE Client
    PEND FI
                              -- Committed ------>
                              <- Forget ----------
                                                  =RR_OK
<---
```

Wenn beim zweiten `Prepare` keine Daten gesendet werden, können analog zu den Beispielen 12 und 13 die folgenden UTM-Aufrufe entfallen:

```
MPUT NE KCRN=>A
CTRL PE KCRN=>A
PEND KP
INIT
MGET NT KCRN=>A VGST=C,TAST=P
```

16. Zwei Transaktionen in einem Vorgang, zweite Variante

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=C
    MPUT NE KCRN=>A
    PEND KP
                            -- FMH5+Daten ----->
                                                CMACCP
                                                CMRCV
                                                -> CM_SEND_RECEIVED
                                                CMSEND
                            <- Daten -----------
    INIT
    MGET NT KCRN=>A
        VGST=O,TAST=O
    MPUT NE KCRN=>A KCLM=O
    CTRL PR KCRN=>A
    PEND KP
                            -- Prepare -------->
                                                CMRCV
                                                -> CM_TAKE_COMMIT
                                                SRRCMIT
                            <- RequestCommit ---
    INIT
    MGET NT KCRN=>A
        VGST=O,TAST=P
    MPUT NE Client
    PEND RE
                            -- Committed ------>
                            <- Forget ----------
                                                =RR_OK
<---
```

Fortsetzung nächste Seite

Fortsetzung

```
--->
    INIT
    MGET    Client
    MPUT NE KCRN=>A
    CTRL PE KCRN=>A
    PEND KP
                            -- Daten+Prepare -->
                                                CMRCV
                                                -> CM_TAKE_COMMIT_
                                                      DEALLOCATE
                                                SRRCMIT
                            <- RequestCommit ---
    INIT
    MGET NT KCRN=>A
        VGST=C,TAST=P
    MPUT NE Client
    PEND FI
                            -- Committed ------>
                            <- Forget ----------
                                                =RR_OK
<---
```

Ähnlich wie in den Beispielen 12 und 13 können einige UTM-Aufrufe entfallen, wenn man im Teilprogramm auf die Rollback-/Commit-Information von CPI-C verzichtet. Der 2. und 3. Programmabschnitt reduziert sich dann auf die Folge

```
INIT
MGET NT KCRN=>A VGST=O,TAST=O
CTRL PR KCRN=>A
MPUT NE Client
PEND RE
```

17. CPI-C-Anwendungsprogramm setzt Transaktion zurück und bricht Vorgang ab, erste Variante

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=C
    MPUT NE KCRN=>A
    PEND KP
                            -- FMH5+Daten ----->
                                                CMACCP
                                                CMRCV
                                                -> CM_SEND_RECEIVED
                                                SRRBACK
                                                CMSDT
                                                    (CM_DEALLOCATE_ABEND)
                                                CMDEAL
                             <- FMH7 ------------
                            -- FMH7-Response -->
    Rücksetzen der Transaktion
    evtl. mit Meldung KO34 auf Client
    Falls zuvor schon ein Sicherungspunkt
    erreicht war, dann Start des
    Folgeprogramms zu diesem Sicherungspunkt:
    INIT    KCKNZVG=R
    MGET NT KCRN=>A
        VGST=O,TAST=R
    ...
```

18. CPI-C-Anwendungsprogramm setzt Transaktion zurück und bricht Vorgang ab, zweite Variante

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=C
    MPUT NE KCRN=>A
    PGWT KP
                            -- FMH5+Daten ----->
                                                CMACCP
                                                CMRCV
                                                -> CM_SEND_RECEIVED
                                                SRRBACK
                                                CMSDT
                                                    (CM_DEALLOCATE_ABEND)
                                                CMDEAL
                            <- FMH7 ------------
                            -- FMH7-Response -->
        KCTARB=Y
    MGET NT KCRN=>A
        VGST=T,TAST=R
    MPUT    Client
    PEND KCOM=FR
...
```

Im Gegensatz zu Beispiel 17 erhält das Teilprogramm im Beispiel 18 nach dem Rücksetzen durch CPI-C eine Rückmeldung zur Kontrolle.

19. CPI-C-Anwendungsprogramm setzt Transaktion zurück, Vorgang läuft weiter

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=C
    MPUT NE KCRN=>A
    PEND KP
                            -- FMH5+Daten ----->
                                                CMACCP
                                                CMRCV
                                                -> CM_SEND_RECEIVED
                                                CMSEND
                            <- Daten -----------
    INIT
    MGET NT KCRN=>A
        VGST=O,TAST=O
    MPUT NE KCRN=>A
    CTRL PR KCRN=>A
    PEND KP
                            -- Prepare -------->
                                                CMRCV
                                                -> CM_TAKE_COMMIT
                                                SRRCMIT
                            <- RequestCommit ---
    INIT
    MGET NT KCRN=>A
        VGST=O,TAST=P
    MPUT NE Client
    PEND RE
                            -- Committed ------>
                            <- Forget ----------
                                                =RR_OK
<---
--->
    INIT
    MGET    Client
    MPUT NE KCRN=>A
    CTRL PE KCRN=>A
    PEND KP
                            -- Prepare -------->
                                                CMRCV
                                                -> CM_TAKE_COMMIT_
                                                      DEALLOCATE
                                                SRRBACK
```

Fortsetzung nächste Seite

Fortsetzung

```
                            <- FMH7 ------------
                            -- FMH7-Response -->
                                                =RR_OK
<--- K034
--->
    INIT    KCKNZVG=R
    MGET    Client
    MPUT NE KCRN=>A
    CTRL PE KCRN=>A
    PEND KP
                            -- Prepare -------->
                                                CMRCV
                                                -> CM_TAKE_COMMIT_
                                                      DEALLOCATE
                                                SRRCMIT
                            <- RequestCommit ---
    INIT
    MGET NT KCRN=>A
        VGST=C,TAST=P
    MPUT NE Client
    PEND FI
                            -- Committed ------>
                            <- Forget ----------
                                                =RR_OK
<---
```

Es ist zu beachten, dass das UTM-Anwendungsprogramm in diesem Fall nach dem SRRBACK keine Statusnachricht vom Auftragnehmer erhält. openUTM stellt dem Anwendungsprogramm nur dann eine Statusnachricht vom Autragnehmer zu, wenn der Auftragnehmer durch das Rücksetzen der Transaktion beendet wurde.

Neben der K034-Meldung wird die Nachricht an den Client vom letzten Sicherungspunkt nochmal an den Client gesendet.

Ähnlich wie in den Beispielen 12 und 13 können einige UTM-Aufrufe entfallen, wenn man im Teilprogramm auf die Rollback-/Commit-Information von CICS verzichtet. Der 2. und 3. Programmabschnitt reduziert sich dann auf die Folge

```
INIT
MGET NT KCRN=>A VGST=O,TAST=O
CTRL PR KCRN=>A
MPUT NE Client
PEND KCOM=RE
```

20. Einschrittvorgang ohne Transaktionssicherung

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=B
    MPUT NE KCRN=>A
    PEND KP
                              -- FMH5+Daten ----->
                                                  CMACCP
                                                  CMRCV
                                                  -> CM_SEND_RECEIVED
                                                  CMSEND
                                                  CMDEAL
                              <- Daten+CEB -------
    INIT
    MGET NT KCRN=>A
        VGST=C,TAST=U
    MPUT NE Client
    PEND FI
<---
```

21. Verarbeitungsquittung ohne Transaktionssicherung

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=H
    MPUT NE KCRN=>A
    MPUT HM KCRN=>A
    PEND KP
                            -- FMH5+Daten ----->
                                                CMACCP
                                                CMRCV
                                                -> CM_CONFIRM_SEND_
                                                      RECEIVED
                                                CMCFMD
                                                CMSEND
                                                CMSDT(CM_DEALLOCATE_FLUSH)
                                                CMDEAL
                            <- Daten+CEB -------
    INIT
    MGET NT KCRN=>A
        KCRMGT=C
    MGET NT KCRN=>A
        VGST=C,TAST=U
    MPUT NE Client
    PEND FI
<---
```

22. Negative Verarbeitungsquittung ohne Transaktionssicherung

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=H
    MPUT NE KCRN=>A
    MPUT HM KCRN=>A
    PEND KP
                            -- FMH5+Daten ----->
                                                CMACCP
                                                CMRCV
                                                -> CM_CONFIRM_SEND_
                                                      RECEIVED
                                                CMSERR
                                                CMSEND
                                                CMSDT(CM_DEALLOCATE_FLUSH)
                                                CMDEAL
                            <- FMH7 ------------
    INIT
    MGET NT KCRN=>A
        KCRMGT=E
    MGET NT KCRN=>A
        VGST=C,TAST=U
    MPUT NE Client
    PEND FI
<---
```

Start eines CPIC-Asynchron-Vorgangs aus einem UTM-Anwendungsprogramm

23. Start eines CPI-C-Asynchron-Vorgangs aus einem UTM-Anwendungsprogramm ohne Transaktionssicherung

```
INIT
APRO AM KCPI=>A KCOF=B
FPUT NE KCRN=>A
PEND FI
                        -- FMH5+Daten ----->
                                            CMACCP
                                            CMRCV
                                            -> CM_DATA
                                            CMRCV
                                            -> CM_CONFIRM_DEALLOC_
                                                  RECEIVED
                                            CMCFMD
                        <- Daten+CEB -------
```

Für diese Art der Kommunikation muss eine openUTM-LU62-Instanz verwendet werden, die mit dem Parameter UTM-ASYNC=YES generiert ist.

Der in Beispiel 20 aufgeführte Fall ist vom Ablauf her ähnlich. Der wesentliche Unterschied besteht darin, dass das CPI-C-Programm in Beispiel 20 noch Daten zurücksenden kann. Auf UTM-Seite hat der Dialog-Fall den Vorteil, dass die Anwendung über Erfolg oder Misserfolg des Aufrufs informiert wird.

Sendet das CPI-C-Programm eine negative Verarbeitungsquittung mittels CMSERR, so löscht openUTM den Auftrag. Das UTM-Anwendungsprogramm kann jedoch selbstverständlich nicht mehr benachrichtigt werden.

24. Start eines CPI-C-Asynchron-Vorgangs aus einem UTM-Anwendungsprogramm mit Transaktionssicherung

```
INIT
APRO AM KCPI=>A KCOF=C
FPUT NE KCRN=>A
PEND FI
                        -- FMH5+Daten+
                           Prepare -------->
                                            CMACCP
                                            CMRCV
                                            -> CM_TAKE_COMMIT_
                                                  DEALLOCATE
                                            SRRCMIT
                        <- RequestCommit ---
                        -- Committed ------>
                        <- Forget ----------
                                            =RR_OK
                                            RETURN
```

Setzt das CPI-C-Programm die Transaktion zurück mittels SRRBACK, so löscht openUTM den Auftrag. Das UTM-Anwendungsprogramm kann jedoch selbstverständlich nicht mehr benachrichtigt werden.

Die Transaktionssicherung im Beispiel 24 bringt für die Anwendungsprogramme gegenüber dem Ablauf ohne Transaktionssicherung (Beispiel 23) folgende Vorteile:

– Das Absenden der Asynchron-Nachricht im UTM-Anwendungsprogramm wird automatisch mit anderen Abläufen im Anwendungsprogramm, z. B. mit einem Datenbank-Update gekoppelt.

– Wenn der PEND FI erfolgreich ist, wird der CICS-Asynchron-Vorgang garantiert genau einmal gestartet.

Dieser Fall ist ähnlich zu Beispiel 14. Die CPI-C-Programmierung ist bei beiden Beispielen identisch. Auf UTM-Seite hat der Dialog-Fall (Beispiel 14) den Vorteil, dass die Anwendung über Erfolg oder Misserfolg des Aufrufs informiert wird. Der asynchrone Fall (Beispiel 24) eignet sich vor allem für zeitgesteuerte Aufträge.

5 openUTM-IMS-Kopplung über LU6.2

In diesem Kapitel finden Sie die Aspekte der openUTM-IMS-Kopplung über LU6.2 bezüglich Generierung und Programmierung. IMS ist ein Transaktionsmonitor und eine Datenbank der Firma IBM. Seit der Version 6 hat der IMS-Transaktionsmonitor (IMS TM) die Fähigkeit, eine transaktionsgesicherte LU6.2-Kommunikation zu Partner-Anwendungen zu betreiben. Diese kann genutzt werden bei einer Kopplung zu openUTM.

5.1 openUTM-IMS-Kopplung generieren

5.1.1 IMS-Startup-Parameter

Soll eine IMS-Anwendung LU6.2 benutzen, so muss hierfür die Komponente APPC/IMS verwendet werden. APPC/IMS ist Teil von IMS TM. Es benutzt 2 Komponenten des Betriebssystems z/OS: APPC/MVS und RRS/MVS. Soll eine IMS-Anwendung LU6.2-Protokolle nutzen, so muss bei den IMS-Startup-Parametern oder im Member DFSPBxxx von IMS.PROCLIB der Wert APPC=Y angegeben werden. Bei transaktionsgesicherter Kommunikation muss zusätzlich RRS=Y angegeben werden.

Beispiel für IMS-Startup-Parameter in der sogenannten DCC-Prozedur:

```
//         PROC RGN=2000K,SOUT=A,DPTY='(14,15)',
//           SYS=,SYS1=,SYS2=,
//           RGSUF=IV1,PARM1=APPC=Y,PARM2=RRS=Y,APPLID1=IMS81CR1,AOIS=R
//IEFPROC EXEC PGM=DFSMVRC0,DPRTY=&DPTY,
//           REGION=&RGN,
//           PARM='CTL,&RGSUF,&PARM1,&PARM2,&APPLID1,&AOIS'
```

APPC/MVS ist ein Subsystem im Betriebssystem z/OS, das das LU6.2-Protokoll realisiert. RRS/MVS (Resource Recovery Services) ist eine Komponente im Betriebssystem, die als Sync-Point-Manager das 2-Phase-Commit zwischen Datenbanken und LU6.2-Partnern koordiniert.

5.1.2 Definition des LU-Namens von IMS

Für die LU6.2-Kommunikation benötigt IMS eine APPLID, die nicht mit der APPLID für sonstige SNA-Kommunikation identisch sein darf. Die APPLID fungiert als LU-Name. Sie muss in VTAM definiert sein. Beispiel für eine derartige VTAM-Definition:

```
IMS     VBUILD TYPE=APPL           APPLICATION MAJOR NODE
IMSAPPL1 APPL ACBNAME=IMSAPPL1,    ACBNAME FOR APPC
        APPC=YES,
        ATNLOSS=ALL,
        DLOGMOD=APPCMODE,
        DMINWNL=5,
        DSESLIM=10,
        MODETAB=LOGMODES,
        PARSESS=YES,
        SECACPT=NONE,
        SRBEXIT=YES,
        SYNCLVL=SYNCPT
```

Dabei ist IMSAPPL1 die APPLID von IMS. APPCMODE ist in diesem Beispiel der MODE-Name. Mit DSESLIM gibt man die Maximalzahl von LU6.2-Sessions an, mit DMINWNL die Minimalzahl von Contention-Winner-Sessions. Mit SECACPT wird gesteuert, ob die LU6.2-Partnerprogramme beim Aufbau einer Conversation eine Userid und eventuell ein Passwort mitgeben müssen. SYNCLVL=SYNCPT wird benötigt für 2-Phase-Commit-Kommunikation.

Zudem muss die APPLID auch bei APPC/MVS bekannt gegeben werden. Dies geschieht mit einer LUADD-Anweisung in der SYS1.PARMLIB(APPCPMxx):

```
LUADD ACBNAME(IMSAPPL1) BASE SCHED(IMSREG) TPDATA(SYS1.APPCTP)
```

Dabei ist IMSAPPL1 wiederum die APPLID von IMS. Mit dem Parameter SCHED wird die IMS-Region angegeben. Der bei TPDATA angegebene Wert ist der Name der Datei, in der die TP_Profile-Definitionen enthalten sind.

Die IMS-Systemdefinition wird mit einer Reihe von Assembler-Makros festgelegt. Änderungen der Systemdefinition zur Laufzeit sind möglich über das Zusatzprodukt ETO (Extended Terminal Option).

5.1.3 Definition von IMS-Transaktionen

DL/I muss als ein API definiert werden. IMS-Transaktionscodes, die mit DL/I realisiert sind, müssen in der IMS-Systemdefinition über ein TRANSACT-Makro definiert werden. Zusätzlich muss das zugehörige Anwendungsprogramm mit einem APPLCTN-Makro definiert werden. IMS-Transaktionscodes, die mit CPI-C realisiert sind, müssen dagegen im TP_Profile von APPC/MVS definiert werden.

Beispiel für APPLCTN- und TRANSACT-Makro

```
APPLCTN GPSB=IMS1PG1,PGMTYPE=TP
TRANSACT CODE=IMS1TR1,MSGTYPE=(,RESPONSE),INQUIRY=NO,MODE=SNGL
```

Dabei bezeichnet IMS1PG1 den Namen des Programms und IMS1TR1 den Namen des Transaktionscodes.

Bei Mehrschritt-Transaktionen muss im TRANSACT-Makro der Parameter SPA angegeben werden. Der Wert dieses Parameters beschreibt die Größe einer sogenannten Scratchpad-Area. Sie dient dazu, Daten über mehrere Dialogschritte hinweg verfügbar zu halten.

Beispiel für eine TP_Profile-Definition

```
TPADD TPSCHED_EXIT(DFSTPPE0)
           TPNAME(IMS1TPN)
           SYSTEM
           ACTIVE(YES)
           TPSCHED_DELIMITER(##)
              TRANCODE=IMS1SRVR
              CLASS=15
              MAXRGN=20
              RACF=NONE
              CPUTIME=100
              ##
```

Dabei bezeichnet IMS1TPN den LU6.2-TP-Namen, also den Namen, der von openUTM aus angesprochen werden muss. Der beim Parameter TRANCODE verwendete Name IMS1SVR kann unterschiedlich von diesem Namen gewählt werden und wird dann in IMS als Transaktionscode verwendet.

5.1.4 Definition von Partner-LU und openUTM-Transaktionen

Agiert IMS nur als Auftragnehmer, so sind keine weiteren Definitionen in IMS erforderlich. Agiert IMS auch als Auftraggeber, so muss die Adressinformation für das Partner-Programm generiert werden.

IBM empfiehlt, Partner-Transaktionen als Side-Information in APPC/MVS zu definieren. Der Begriff Side-Information kommt aus dem Umfeld von CPI-C. In der Side-Information werden einem symbolischen Namen (symbolic destination name) Partner-LU-Name, Partner-TP-Name und MODE-Name zugeordnet. Der symbolische Name wird dann in den Anwendungsprogrammen zur Adressierung der Partner-Transaktion verwendet. Eine Side-Information in APPC/MVS sieht z.B. wie folgt aus:

```
SIADD
    DESTNAME(DESTUTM1)
    TPNAME(UTMTAC01)
    MODENAME(APPCMODE)
    PARTNER_LU(APPCLUX)
```

Der Name der Side-Information-Datei muss IMS bekannt gegeben werden. Dies geschieht mit einer SIDEINFO-Anweisung in der SYS1.PARMLIB(APPCPMxx):

```
SIDEINFO DATASET(SYS1.APPCSI)
```

Wird die Standard-DL/I-Programmschnittstelle verwendet ohne die LU6.2-Erweiterungen des CHNG-Aufrufs, so kann die openUTM-Transaktion nicht über Side-Information adressiert werden. Stattdessen muss ein LTERM-Name verwendet werden. Ein LTERM in IMS ist ähnlich wie bei openUTM ein symbolischer Name für einen Kommunikationspartner, der an der DL/I-Programmschnittstelle verwendet wird. LTERM-Namen müssen mittels IMS-Generierungsparameter einem realen Kommunikationspartner zugeordnet werden.

Bei dieser Art von Adressierung muss zunächst ein Alternate PCB definiert werden. Zur Definition eines Alternate PCB sind Generierungsanweisungen an drei unterschiedlichen Stellen erforderlich:

1. PCB-Anweisung in IMS PSBGEN

Beispiel:

```
PCB TYPE=TP,MODIFY=YES
PSBGEN PSBNAME=IMSTOJMS,CMPAT=YES,LANG=COBOL
```

2. Definition von ACBGEN

Beispiel:

```
//ACBGEN   EXEC ACBGEN,COMP='POSTCOMP'
//G.SYSIN  DD *
  BUILD PSB=IMSTOJMS
```

3. APPLCTN-Makro mit Parameter PSB und TRANSACT-Makro

Beispiel:

```
APPLCTN  PSB=IMSOJMS,PGMTYPE=TP
TRANSACT CODE=IMSD,MSGTYPE=(,RESPONSE),INQUIRY=NO,MODE=SNGL
```

Zusätzlich muss mit einem sogenannten LU6.2-Descriptor in IMS.PROCLIB(DFS62DTx) ein LTERM-Name definiert und diesem LU-Name, TP-Name und MODE-Name zugewiesen werden. Dies sieht z.B. wie folgt aus:

```
A UTMIMS01 LUNAME=APPCLUX TPNAME=UTMTAC01 MODE=APPCMODE SYNCLEVEL=N
```

Der LTERM-Name heißt in diesem Beispiel UTMIMS01.

Man kann auch beide Typen der Partner-Definition kombinieren, indem man beim LU6.2-Descriptor anstelle der Parameter LUNAME, TPNAME, MODE den Parameter SIDE verwendet und damit auf eine Side-Information verweist.

Im Gegensatz zu CICS gibt es bei IMS keine Festlegung der Maximalzahl von Sessions zu einer Partner-LU. Diese Maximalzahl wird in der APPL-Definition von VTAM festgelegt.

5.1.5 VTAM-Generierung

Ein erster Teil der benötigten VTAM-Generierung wurde oben beim APPL-Makro beschrieben.

Der für die LU6.2-Kommunikation verwendete MODE-Name muss in VTAM definiert werden. Beispiel für die Definition eines MODE-Namens:

```
APPCMODE MODEENT LOGMODE=APPCMODE,                                    *
               RUSIZES=X'8989',                                       *
               PSNDPAC=X'00',                                         *
               SRCVPAC=X'00',                                         *
               SSNDPAC=X'01'
```

Mit RUSIZES wird die maximale RU-Size in Sende- und Empfangsrichtung festgelegt. Die Codierung erfolgt in 2 Hex-Ziffern X'abab'. Die tatsächliche RUSIZE berechnet sich dann mit der Formel n = a* 2b. Die Werte bei PSNDPAC, SRCVPAC, SSNDPAC beeinflussen den Pacing-Count.

Beispiel für die VTAM-Definition von Partner-PU und Partner-LU

```
PO2CG3   PU    ADDR=C1,                                          *
               DISCNT=NO,                                        *
               DLOGMOD=SNX32702,   WITH QUERY                    *
               IDBLK=017,          IDBLK FOR IBM-3174            *
               IDNUM=20003,                                      *
               ISTATUS=ACTIVE,                                   *
               MAXDATA=1033,                                     *
               MAXOUT=7,                                         *
               MAXPATH=2,                                        *
               MODETAB=MOD3270,    LOGMODE TABLE FOR REMOTE 3270 *
               PACING=0,                                         *
               PUTYPE=2,                                         *
               SSCPFM=USSSCS,                                    *
               USSTAB=USSSCS,                                    *
               VPACING=0
*
APPCLUX LU     LOCADDR=0,           INDEPENDENT LU               *
               USSTAB=USSSCS,       ACF/VTAM - USS-TABLE         *
               DLOGMOD=APPCMODE,    ACF/VTAM - DEFAULT LOGMODE   *
               MODETAB=LOGMODES,    LOGMODE TABLE                *
               RESSCB=4,            RESERVE 4 SCB'S FOR THIS LU  *
               PACING=3,                                         *
               VPACING=2,                                        *
               SSCPFM=FSS
```

Beispiel für die Definition eines mit Enterprise Extender angebundenen Partnersystems ohne Definition der LUs

```
SMNEEA2  VBUILD TYPE=SWNET
*
PUEEA1   PU    IDBLK=003,                                          *
               IDNUM=00002,                                        *
               MAXPATH=5,                                          *
               MAXDATA=256,                                        *
               ADDR=01,                                            *
               CPNAME=MCHOOXYC,                                    *
               CPCP=YES,                                           *
               HPR=YES,                                            *
               PUTYPE=2
PATH2A   PATH  SAPADDR=8,                                          *
               IPADDR=111.22.333.144,
               GRPNM=GPP390,                                       *
```

5.1.6 LU6.2-Security

Unter LU6.2-Security versteht man das Schützen von Transaktionen gegen unerlaubte Zugriffe. Beim Aufbau einer Conversation gibt es 3 Security-Varianten:

- NONE: Es wird keine Userid mitgegeben.
- ALREADY_VERIFIED: Es wird eine Userid mitgegeben und ein Already-Verified-Indikator, der dem Auftragnehmer mitteilt, dass der User beim Auftraggeber bereits verifiziert wurde.
- PROGRAM: Es wird eine Userid und ein Passwort mitgegeben.

IMS verwendet das Produkt RACF zur Zugriffskontrolle. Userids, die von Partner-Anwendungen gesendet werden, müssen somit immer in RACF eingetragen sein. Welcher User welchen Transaktionscode aufrufen darf, ist in RACF eingetragen. Auch die Passworte zu den Userids sind in RACF eingetragen.

Ob beim Aufbau einer LU6.2-Conversation zu IMS eine Userid und eventuell ein Passwort mitgegeben werden müssen, ist im Parameter SECACPT des VTAM-APPL-Makros festgelegt. Details zur RACF-Prüfung von eingehenden Conversations sind im Parameter RACF der TP_Profile-Definition festgelegt.

5.1.7 Vollständiges Generierungsbeispiel

Das folgende Beispiel enthält die Generierungsanweisungen einer openUTM-IMS-Kopplung mit folgenden Randbedingungen:

- openUTM-LU62 läuft auf einem Windows-System
- Kopplung zwischen IBM-Host und Windows-System mittels Enterprise Extender

Die Dateinamen der IMS- und VTAM-Definitionen sind als Beispiel anzusehen und können in echten Einsatzfällen anders gewählt sein.

IMS-Definitionen

IMS-Systemdefinitionen in IMS.CNTL(STAGE1):

```
//STAGE1   EXEC PGM=ASMA90,PARM='NOOBJ,DECK',REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSLIB   DD DISP=SHR,DSN=IMS810.ADFSMAC
//SYSPUNCH DD DISP=SHR,DSN=USER.IMS.CNTL(STAGE2)
//SYSUT1   DD UNIT=3390,SPACE=(CYL,(05,05)),DCB=OPTCD=C
//SYSUT2   DD UNIT=3390,SPACE=(CYL,(05,05)),DCB=OPTCD=C
//SYSUT3   DD UNIT=3390,SPACE=(CYL,(05,05)),DCB=OPTCD=C
//SYSIN    DD *
         APPLCTN  GPSB=TESTIMS1,PGMTYPE=TP
         TRANSACT CODE=IMS1,MSGTYPE=(,NONRESPONSE),INQUIRY=NO,MODE=SNGL
         APPLCTN  GPSB=IMSTAC0,PGMTYPE=TP
         TRANSACT CODE=IMS0,MSGTYPE=(,RESPONSE),INQUIRY=NO,MODE=SNGL
        END ,
```

TP_Profile-Definition in IMS.CNTL(APPCTP):

```
//         EXEC PGM=ATBSDFMU
//SYSPRINT DD   SYSOUT=*
//SYSSDLIB DD   DSN=SYS1.APPCTP,DISP=SHR
//SYSSDOUT DD   SYSOUT=*
//SYSIN    DD   DATA,DLM=XX
     TPADD TPSCHED_EXIT(DFSTPPE0)
          TPNAME(IMSX)
          SYSTEM
          ACTIVE(YES)
          TPSCHED_DELIMITER(##)
             TRANCODE=TESTIMSX
             CLASS=1
             MAXRGN=1
             RACF=NONE
             CPUTIME=0
XX
```

Side-Information in IMS.CNTL(APPCSI):

```
//         EXEC PGM=ATBSDFMU
//SYSPRINT DD   SYSOUT=*
//SYSSDLIB DD   DSN=SYS1.APPCSI,DISP=SHR
//SYSSDOUT DD   SYSOUT=*
//SYSIN    DD   DATA,DLM=XX
     SIDELETE
          DESTNAME(DESTJW1)
     SIADD
          DESTNAME(DESTJW1)
          TPNAME(DATAECHO)
          MODENAME(APPCHOST)
          PARTNER_LU(P390.IMSJ)
XX
```

LTERM-Definition in PROCLIB(DFS62DTI):

```
A LTERMJW1 SIDE=DESTJW1  SYNCLEVEL=N CONVTYPE=M
```

Definition der IMS-LU in PARMLIB(APPCPM1A):

```
LUADD ACBNAME(MVSLU01) BASE TPDATA(SYS1.APPCTP)
LUADD ACBNAME(IMSA)    BASE SCHED(IVP1) TPDATA(SYS1.APPCTP)
SIDEINFO DATASET(SYS1.APPCSI)
```

VTAM-Generierung

Start Options:

```
SSCPNAME=P390SSCP,
NETID=P390
```

Switched-Major-Node-Definition in VTAMLST(SWXCAEE):

```
SWXCA1A VBUILD TYPE=SWNET,MAXNO=256,MAXGRP=256
SW1A13A     PU IDBLK=05D,IDNUM=25129,MAXPATH=5,MAXDATA=256,ADDR=01,    -
               CPNAME=MHPB02GC,                                        -
               CPCP=YES,HPR=YES,                                       -
               PUTYPE=2
PATH13A   PATH SAPADDR=4,                                              -
               IPADDR=111.22.33.233,                                   -
               GRPNM=GPP390
```

Major-Node-Definition für Enterprise Extender in VTAMLST(XCAEE1):

```
XCAEEE1 VBUILD TYPE=XCA
PORT1A    PORT MEDIUM=HPRIP,IPPORT=12000,                              -
               IPTOS=(20,40,80,C0,C0),LIVTIME=10,                      -
               SRQTIME=15,SRQRETRY=3,SAPADDR=4
GPP390   GROUP DIAL=YES,ANSWER=ON,ISTATUS=ACTIVE,CALL=INOUT,           -
               DYNPU=YES,IPADDR=111.22.33.244
LNEE01    LINE
PUEE01      PU
LNEE02    LINE
PUEE02      PU
```

APPL-Definition in VTAMLST(A0IMS):

```
A0IMS    VBUILD TYPE=APPL              APPLICATION MAJOR NODE
IMSA     APPL  ACBNAME=IMSA,           ACBNAME MUST BE EQ LABEL        C
               APPC=YES,                                              C
               ATNLOSS=ALL,                                           C
               AUTOSES=0,                                             C
               DDRAINL=NALLOW,                                        C
               DLOGMOD=APPCHOST,                                      C
               DMINWNL=5,                                             C
               DMINWNR=5,                                             C
               DRESPL=NALLOW,                                         C
               DSESLIM=10,                                            C
               LMDENT=19,                                             C
               PARSESS=YES,                                           C
               SECACPT=NONE,                                          C
               SRBEXIT=YES,                                           C
               SYNCLVL=SYNCPT,                                        C
               VPACING=1
```

MODE-Definition in VTAM.SOURCE(ISTINCLM):

```
ISTINCLM MODETAB
APPCHOST MODEENT LOGMODE=APPCHOST,                                    *
               RUSIZES=X'8989',   4096                                *
               SRCVPAC=X'00',                                         *
               SSNDPAC=X'01'
```

IBM Communications Server

```
NODE=(
          ANYNET_SUPPORT=NONE
          CP_ALIAS=MHPB02GC
          DEFAULT_PREFERENCE=NATIVE
          DISCOVERY_SUPPORT=DISCOVERY_SERVER
          DLUR_SUPPORT=NORMAL
          FQ_CP_NAME=P390.MHPB02GC
          MAX_LOCATES=150
          MAX_LS_EXCEPTION_EVENTS=200
          NODE_ID=05D25129
          NODE_TYPE=NETWORK_NODE
          REGISTER_WITH_CDS=1
          REGISTER_WITH_NN=NONE
          SEND_TERM_SELF=0
          TP_SECURITY_BEHAVIOR=VERIFY_EVEN_IF_NOT_DEFINED
)
PORT=(
          ACTIVATION_DELAY_TIMER=30
          ALLOW_ABM_XID_MISMATCH=0
          DELAY_APPLICATION_RETRIES=1
          DLC_NAME=IBMEEDLC
          IMPLICIT_BRANCH_EXTENDER_LINK=0
          IMPLICIT_CP_CP_SESS_SUPPORT=1
          IMPLICIT_DEACT_TIMER=600
          IMPLICIT_DSPU_SERVICES=NONE
          IMPLICIT_HPR_SUPPORT=1
          IMPLICIT_LIMITED_RESOURCE=NO
          IMPLICIT_LINK_LVL_ERROR=0
          LINK_STATION_ROLE=NEGOTIABLE
          MAX_ACTIVATION_ATTEMPTS=0
          MAX_IFRM_RCVD=8
          MAX_RCV_BTU_SIZE=1500
          PORT_NAME=IBMEEDLC
          PORT_TYPE=SATF
          RETRY_LINK_ON_DISCONNECT=1
          RETRY_LINK_ON_FAILED_START=1
          RETRY_LINK_ON_FAILURE=1
```

```
DEFAULT_TG_CHARS=(
          COST_PER_BYTE=0
          COST_PER_CONNECT_TIME=0
          EFFECTIVE_CAPACITY=133
          PROPAGATION_DELAY=LAN
          SECURITY=NONSECURE
          USER_DEFINED_1=0
          USER_DEFINED_2=0
          USER_DEFINED_3=0
)
PORT_OEM_SPECIFIC_DATA=(
OEM_LINK_DATA=(
          OEM_DATA=010000000400000004000000030000000F00000000000000
          OEM_DATA=0A0000000000000000
)
OEM_PORT_DEFAULTS=(
          COST_PER_CONNECT_TIME=0
          EFFECTIVE_CAPACITY=133
          INB_LINK_ACT_LIM=128
          OUT_LINK_ACT_LIM=127
          PROPAGATION_DELAY=LAN
          SECURITY=NONSECURE
          TOT_LINK_ACT_LIM=255
)
)
)
LINK_STATION=(
          ACTIVATE_AT_STARTUP=1
          ACTIVATION_DELAY_TIMER=-1
          ADJACENT_BRANCH_EXTENDER_NODE=PROHIBITED
          ADJACENT_NODE_TYPE=LEARN
          AUTO_ACTIVATE_SUPPORT=0
          BRANCH_EXTENDER_LINK=1
          CP_CP_SESS_SUPPORT=1
          DEFAULT_NN_SERVER=0
          DELAY_APPLICATION_RETRIES=0
          DEPENDENT_LU_COMPRESSION=0
          DEPENDENT_LU_ENCRYPTION=OPTIONAL
          DEST_ADDRESS=F45519AC
          DISABLE_REMOTE_ACT=0
          DSPU_SERVICES=NONE
          FQ_ADJACENT_CP_NAME=P390.P390SSCP
          HPR_LINK_LVL_ERROR=0
          HPR_SUPPORT=1
          INHERIT_PORT_RETRY_PARMS=1
          LIMITED_RESOURCE=NO
          LINK_DEACT_TIMER=600
          LINK_STATION_ROLE=NEGOTIABLE
```

```
          LS_NAME=P390SSCP
          MAX_ACTIVATION_ATTEMPTS=-1
          MAX_IFRM_RCVD=7
          MAX_SEND_BTU_SIZE=1500
          NODE_ID=05D25129
          NULL_ADDRESS_MEANING=USE_WILDCARD
          PORT_NAME=IBMEEDLC
          PU_NAME=P390SSCP
          RETRY_LINK_ON_DISCONNECT=0
          RETRY_LINK_ON_FAILED_START=0
          RETRY_LINK_ON_FAILURE=0
          REVERSE_ADDRESS_BYTES=0
          SOLICIT_SSCP_SESSION=0
          TG_NUMBER=0
          USE_DEFAULT_TG_CHARS=1
          USE_PU_NAME_IN_XID=0
TG_CHARS=(
          COST_PER_BYTE=0
          COST_PER_CONNECT_TIME=0
          EFFECTIVE_CAPACITY=0
          PROPAGATION_DELAY=MINIMUM
          USER_DEFINED_1=0
          USER_DEFINED_2=0
          USER_DEFINED_3=0
)
LINK_STATION_OEM_SPECIFIC_DATA=(
OEM_LINK_DATA=(
          OEM_DATA=0100000004000000040000003000000F00000000000000
          OEM_DATA=0A000000F45519AC
)
)
)
DLUR_DEFAULTS=(
          DEFAULT_PU_NAME=MHPB02GC
          DLUS_RETRY_LIMIT=3
          DLUS_RETRY_TIMEOUT=5
)
LOCAL_LU=(
          LU_NAME=IMSJ
          DEFAULT_POOL=0
          LU_ALIAS=IJWIMSJ
          LU_SESSION_LIMIT=0
          NAU_ADDRESS=0
          ROUTE_TO_CLIENT=0
          SYNCPT_SUPPORT=1
)
```

```
MODE=(
          MODE_NAME=APPCHOST
          AUTO_ACT=6
          COMPRESSION=PROHIBITED
          COS_NAME=#CONNECT
          DEFAULT_RU_SIZE=0
          ENCRYPTION_SUPPORT=NONE
          MAX_INCOMING_COMPRESSION_LEVEL=NONE
          MAX_NEGOTIABLE_SESSION_LIMIT=128
          MAX_OUTGOING_COMPRESSION_LEVEL=NONE
          MAX_RU_SIZE_UPPER_BOUND=4096
          MIN_CONWINNERS_SOURCE=6
          PLU_MODE_SESSION_LIMIT=12
          RECEIVE_PACING_WINDOW=1
)
PARTNER_LU=(
          FQ_PLU_NAME=P390.IMSA
          ADJACENT_CP_NAME=
          CONV_SECURITY_VERIFICATION=1
          MAX_MC_LL_SEND_SIZE=32767
          PARALLEL_SESSION_SUPPORT=1
          PARTNER_LU_ALIAS=PA713236
          PREFERENCE=USE_DEFAULT_PREFERENCE
)
SPLIT_STACK=(
          STARTUP=1
          POOL_NAME=
)
SHARED_FOLDERS=(
          CACHE_SIZE=256
EXTENSION_LIST=(
)
)
LOAD_BALANCING=(
          ADVERTISE_FREQUENCY=1
          APPC_LU_LOAD_FACTOR=0
          DEFAULT_MAX_LU62_SESSIONS=512
          ENABLE_LOAD_BALANCING=0
          HOST_LU_LOAD_FACTOR=0
          LOAD_VARIANCE=3
)

VERIFY=(
          CFG_MODIFICATION_LEVEL=12
          CFG_VERSION_LEVEL=1
          CFG_LAST_SCENARIO=6
)
```

openUTM-LU62-Generierung ohne Verwendung von TNSX

```
INSTANCE
     LOC-LU-ALIAS=IJWIMSJ,
     LOC-TSEL=T'IJWOSICL',
     LOC-LISTENER-PORT=29664,
     LOC-APT=(1,2,29664),
     LOC-AEQ=712614,
     REM-LU-ALIAS=PA713236,
     MODENAME=APPCHOST,
     ALLOC-TIME=0,
     LU62-CODE=*NO,
     REM-NSEL=utmhost,
     REM-LISTENER-PORT=23000,
     REM-TSEL=T'SMP23000',
     REM-APT=(1,2,29660),
     REM-AEQ=1,
     APPL-CONTEXT=UDTSEC,
     ASSOCIATIONS=12,
     CONNECT=12,
     CONTWIN=6,
     OSITP-CODE=*NO
```

openUTM-LU62-Generierung bei Verwendung von TNSX

```
INSTANCE
     LOC-LU-ALIAS=IJWIMSJ,
     LOC-AE=IJWOSICL,
     LOC-APT=(1,2,29664),
     LOC-AEQ=712614,
     REM-LU-ALIAS=PA713236,
     MODENAME=APPCHOST,
     ALLOC-TIME=0,
     LU62-CODE=*NO,
     REM-AE=BCOSIO.SMP23000,
     REM-APT=(1,2,29660),
     REM-AEQ=1,
     APPL-CONTEXT=UDTSEC,
     ASSOCIATIONS=12,
     CONNECT=12,
     CONTWIN=6,
     OSITP-CODE=*NO
```

TNSX-Generierung

```
BCOSIO.SMP23000\
          TA        RFC1006 111.33.55.111 PORT 23000 T'SMP23000'
          PSEL      V''
          SSEL      V''
IJWOSICL\
          TSEL      RFC1006 T'IJWOSICL'
          TSEL      LANINET A'29664'
          PSEL      V''
          SSEL      V''
```

UTM-Generierung

```
UTMD    APT=(1,2,29660),                        -
        MAXJR=200,                              -
        CONCTIME=(180,60),                      -
        PTCTIME=0
ACCESS-POINT BCOSIO,                            -
        P-SEL=*NONE,                            -
        S-SEL=*NONE,                            -
        T-SEL=C'SMP23000',                       -
        AEQ=1,                                  -
        T-PROT=RFC1006,                         -
        LISTENER-PORT=23000,                    -
        TSEL-FORMAT=T
OSI-CON IJWOSICL,                               -
        ACTIVE=YES,                             -
        LISTENER-PORT=29664,                    -
        LOCAL-ACCESS-POINT=BCOSIO,              -
        MAP=USER,                               -
        N-SEL=C'MHPBO2GC',                      -
        OSI-LPAP=IJWOSICL,                      -
        P-SEL=*NONE,                            -
        S-SEL=*NONE,                            -
        T-PROT=RFC1006,                         -
        T-SEL=C'IJWOSICL',                      -
        TSEL-FORMAT=T
```

```
OSI-LPAP IJWOSICL,                                -
        APPLICATION-CONTEXT=UDTSEC,               -
        APPLICATION-ENTITY-QUALIFIER=712614,      -
        APPLICATION-PROCESS-TITLE=(1,2,29664),    -
        ASSOCIATION-NAMES=IJW,                    -
        ASSOCIATIONS=12,                          -
        CONNECT=12,                               -
        CONTWIN=6,                                -
        IDLETIME=0,                               -
        PERMIT=ADMIN,                             -
        STATUS=ON
TAC DATAECHO,                                     -
        PROGRAM=comims,                          -
        TYPE=D
LTAC    LIMS0,                                    -
        RTAC=IMS0,                                -
        LPAP=IJWOSICL,                            -
        TYPE=D
LTAC    LIMS1,                                    -
        RTAC=IMS1,                                -
        LPAP=IJWOSICL,                            -
        TYPE=A
LTAC    ICPC,                                     -
        RTAC=CSVRCPIC,                            -
        LPAP=IJWOSICL,                            -
        TYPE=D
```

Die wesentlichen Parameter in diesem Beispiel sind:

Parameter	**IMS-Host**	**openUTM-LU62 Windows**	**openUTM Endsystem**
IP-Adresse	111.22.33.244 (F45519AC)	111.22.33.233	111.33.55.111
LU-Name	P390.IMSA	P390.IMSJ	
LU-Alias-Name	PA713236	IJWIMSJ	
CP-Name	P390.P390SSCP	P390.MHPB02GC	
IDBLK/IDNUM		05D 25129	
MODE-Name	APPCHOST		
T-Selektor		IJWOSICL	SMP23000
Port-Nummer		29664	23000
AP-Title		(1,2,29664)	(1,2,29660)
AE-Qualifier		712614	1
APPL-Context		UDTSEC	

5.2 openUTM-IMS-Kopplung programmieren

IMS-Anwendungsprogramme, die LU6.2 nutzen, können mit den beiden Programmschnittstellen DL/I oder CPI-C realisiert sein.

In der IMS-Literatur unterscheidet man zwischen 3 Varianten von IMS-Anwendungsprogrammen bei LU6.2-Kommunikation:

- Standard DL/I: Anwendungsprogramm verwendet nur DL/I-Aufrufe.
- Modified DL/I: Zum Lesen der Nachricht vom Auftraggeber und zum Senden der Antwort an den Auftraggeber wird DL/I verwendet. Zusätzlich wird CPI-C verwendet, um eine Kommunikation zu einem Auftragnehmer aufzubauen.
- CPI-C: Anwendungsprogramm verwendet ausschließlich CPI-C-Aufrufe.

5.2.1 Programmschnittstelle DL/I

DL/I ist die herkömmliche IMS-Programmschnittstelle. Bestehende DL/I-Programme, die Nachrichten von Kommunikationspartnern entgegennehmen und an denselben Partner wieder eine Antwort zurücksenden, können oft ohne Änderung auch mit LU6.2-Partnern kommunizieren. Auch 2-Phase-Commit-Transaktionen können auf diese Weise bearbeitet werden. Zusätzlich ist es mit DL/I möglich, einem LU6.2-Partner eine asynchrone, nicht transaktionsgesicherte Nachricht zu senden, vergleichbar dem FPUT bei openUTM. DL/I bietet jedoch keine Möglichkeit, eine 2-Phase-Commit-Transaktion beim Partner aufzurufen. Auch das Setzen einer Userid beim Senden einer asynchronen Nachricht ist bei DL/I nicht möglich.

In DL/I stehen für die Datenkommunikation folgende Aufrufe zur Verfügung:

Name	Parameter	Bedeutung	Verwendung
AUTH	I/O-PCB, I/O-Area	authorization	Prüfen, ob der angemeldete Benutzer bestimmte Funktionen aufrufen und Ressourcen nutzen darf
CHNG	Alt-PCB, dest name, opt list	change	Adressieren eines Partners
CMD	I/O-PCB, I/O-Area	command	Absetzen eines Administrations-Kommandos
GCMD	I/O-PCB, I/O-Area	get command	Lesen der Rückmeldung zum Administrations-Kommando
GU	I/O-PCB, I/O-Area	get unique	Lesen der ersten Nachricht
GN	I/O-PCB, I/O-Area	get next	Lesen einer Folgenachricht
ISRT	I/O-PCB oder Alt-PCB, I/O-Area [, mod name]	insert	Senden einer Nachricht

PURG	I/O-PCB oder Alt-PCB	purge	Sofortiges Senden einer Nachricht (vgl. Flush bei CPI-C)
SETO	I/O-PCB oder Alt-PCB, I/O-Area, opt list	set options	Senden von Send_Error oder Deallocate_Abend

Zusätzlich können im LU6.2-Umfeld folgende DL/I-Aufrufe für System Services genutzt werden:

Name	Parameter	Bedeutung	Verwendung
INQY	aib, I/O-Area	inquiry	Abfragen von Eigenschaften des Output-PCB
ROLB	I/O-PCB, I/O-Area	rollback	Senden von Deallocate_Abend und Rollback der Transaktion, normale Fortsetzung des Anwendungsprogramms
ROLL	keiner	roll	Senden von Deallocate_Abend und Rollback der Transaktion, abnormales Programmende, Löschen der Eingabe-Nachricht
ROLS	I/O-PCB, I/O-Area, token	rollback to sets/setu	Senden von Deallocate_Abend und Rollback der Transaktion, Programmende, Eingabe-Nachricht bleibt in Queue erhalten, Rücksetzpunkt kann spezifiziert werden
SETS	I/O-PCB, I/O-Area, token	set backout point	Setzen eines Rücksetzpunktes für einen nachfolgenden ROLS
SETU	I/O-PCB, I/O-Area, token	set unconditional	Wie SETS, jedoch mit etwas anderem Verhalten in Fehlersituationen
SYNC	I/O-PCB	sycnpoint	Setzen eines Sicherungspunktes

Bedeutung der Parameter

I/O-PCB und Alt-PCB:

Beim Empfangen und Senden einer Nachricht wird ein PCB (program communication block) übergeben, in dem Informationen wie Status, Partnername (LTERM) usw. abgespeichert werden. Wird eine Nachricht an den Auftraggeber zurückgesendet, wird der I/O-PCB benutzt. Zum Senden einer Nachricht an einen Auftragnehmer muss ein Alternate-PCB benutzt werden, bei dem vorher mit dem Aufruf CHNG eine neue Zieladresse gesetzt wird. Bestandteil des PCB ist der Partner-Typ. Folgende Partner-Typen sind möglich:

- APPC (LU6.2-Partner)
- OTMA (TCP/IP-Partner)
- TERMINAL (Terminal oder LU6.1-Partner)
- TRANSACTION (Transaktionscode).

I/O-Area

ist der Nachrichtenbereich.

dest name (Destination Name):
: LTERM- oder Transaktionscode-Name des Partners.

opt list (Options list):
: Bei CHNG können hier Parameter des LU6.2-Partners angegeben werden, nämlich LU-Name, MODE-Name, TP-Name und Sync-Level (none oder confirm). Alternativ kann auch ein Symbolic Destination Name angegeben werden. Bei LU6.2-Kommunikation erfolgt die Adressierung entweder über opt list oder über einen LTERM-Namen im Feld dest name.
: Bei SETO wird hier angegeben, ob Send_Error oder Deallocate_Abend gesendet werden soll.

mod name (Message output descriptor name):
: Formatname.

aib (Application Interface Control Block):
: Dieser enthält die PCB-Adresse und einige Zusatzinformationen.

token: Name eines Rücksetzpunktes.

Es ist möglich, auch längere Dialoge abwechselnd mit GU/GN und ISRT zu programmieren.

Beispielhafter Auszug aus einem DL/I-COBOL-Programm (GN-Aufruf)

```
ID DIVISION.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
  77 GN-FUNC      PIC X(4) VALUE 'GN  '.
  01 IOAREA       PIC X(256).
LINKAGE SECTION.
  01 D1PC.
    02 D1PCDBN  PIC X(8).
    02 D1PCLEVL PIC 9(2).
    02 D1PCSTAT PIC X(2).
    02 D1PCPROC PIC X(4).
    02 D1PCRESV PIC S9(5) COMP.
    02 D1PCSEGN PIC X(8).
    02 D1PCKFBL PIC S9(5) COMP.
    02 D1PCNSSG PIC S9(5) COMP.
    02 D1PCKFBA PIC X(20).
PROCEDURE DIVISION.
  ENTRY 'DLITCBL' USING D1PC.
  ...
  CALL 'CBLTDLI' USING GN-FUNC, D1PC, IOAREA.
  IF D1PCSTAT NOT = '  '
    ...
  GOBACK.
```

5.2.2 Programmschnittstelle CPI-C

CPI-C ermöglicht die Nutzung des kompletten LU6.2-Umfangs. Nur mit dieser Programmschnittstelle ist es möglich, von IMS-Programmen auch Synclevel-Syncpoint-Conversations zu Partnern aufzubauen. Zur Steuerung des 2-Phase-Commit muss in diesem Fall CPI-RR verwendet werden. Für weitere Informationen siehe auch Abschnitt „Verwendung der Programmschnittstelle CPI-C“ auf Seite 151.

5.2.3 LU6.2-Edit-Exit-Routine

Die LU6.2-Edit-Exit-Routine (DFSLUEE0) wird von IMS immer durchlaufen, wenn ein DL/I-Programm eine LU6.2-Nachricht sendet oder empfängt. IBM liefert eine fertige Exit-Routine aus, die aber vom Kunden abgeändert werden kann. Die Exit-Routine kann zu folgenden Zwecken eingesetzt werden:

- Änderung des Synchronization-Level einer asynchronen LU6.2-Conversation
- Änderung von Nachrichteninhalten
- Löschen von Nachrichten-Segmenten
- Abbruch (DEALLOCATE_ABEND) einer LU6.2-Conversation

5.2.4 Benutzung von Formatnamen

MFS (Message Format Service) ist die IMS-Komponente zum Behandeln formatierter Ein- und Ausgaben auf zeichenorientierten Bildschirmen (Typ 3270).

DL/I-Anwendungsprogramme erhalten beim Einlesen einer 3270-Nachricht im Aufruf GU einen Formatnamen, den sogenannten MOD-Name. Dabei steht MOD für Message Output Descriptor. Beim ISRT können sie einen MOD-Name mitgeben. Sollen derartige Programme unverändert für LU6.2-Kommunikation verwendet werden, so muss die LU6.2-Edit-Exit-Routine (DFSLUEE0) für diese Zwecke verwendet werden. Die LU6.2-Partnerprogramme müssen dann am Anfang der Nachricht den Formatnamen senden und erhalten ebenso am Anfang der Rücknachricht einen Formatnamen.

5.2.5 Ablauf-Beispiele mit DL/I-Programmen

In den folgenden Beispielen werden Abläufe mit openUTM- und IMS-Programmen dargestellt. Auf beiden Seiten sind nur die Programmaufrufe für die LU6.2-Kommunikation aufgeführt. Für die Kommunikation mit dem openUTM-Client oder dem IMS-Client sind weitere Programmaufrufe notwendig.

Die Rückgabewerte im UTM-Anwendungsprogramm sind *kursiv* dargestellt.

1. Synchroner Aufruf einer IMS-Transaktion von openUTM ohne Transaktionssicherung

IMS-Anwendung		UTM-Anwendung
		APRO DM KCOF=B, KCRN=A
		MPUT NE, KCRN=A
		PEND KP
	<-- FMH5 + Daten	
GU IOPCB		
ISRT IOPCB		
Exit		
	Daten + CEB -->	
		MGET NT *VGST=C,* KCRN=A

2. Asynchroner Aufruf einer IMS-Transaktion von openUTM ohne Transaktionssicherung

IMS-Anwendung		UTM-Anwendung
		APRO AM KCOF=B, KCRN=A
		FPUT NE, KCRN=A
		PEND FI
	<-- FMH5 + Daten + CEB	
	Quittung -->	
GU IOPCB		
ISRT IOPCB		
Exit		
	FMH5 + Daten + CEB -->	
	<-- Quittung	
		TAC DFSASYNC
		FGET
		PEND FI

Für diese Art der Kommunikation muss eine openUTM-LU62-Instanz verwendet werden, die mit dem Parameter UTM-ASYNC=YES generiert ist.

Bemerkung:
Der TAC DFSASYNC muss in der UTM-Anwendung als ASYNTAC generiert sein. Der Name DFSASYNC ist nicht frei wählbar, sondern wird von IMS vorgegeben.

3. Asynchroner Aufruf einer IMS-Transaktion von openUTM ohne Transaktionssicherung, IMS-Programm sendet keine Nachricht

IMS-Anwendung		**UTM-Anwendung**
		APRO AM KCOF=B, KCRN=A
		FPUT NE, KCRN=A
		PEND FI
	<-- FMH5 + Daten + CEB	
	Quittung -->	
GU IOPCB		
Exit		

Für diese Art der Kommunikation muss eine openUTM-LU62-Instanz verwendet werden, die mit dem Parameter UTM-ASYNC=YES generiert ist.

4. Synchroner Aufruf einer IMS-Transaktion von openUTM ohne Transaktionssicherung, Mehrschritt-Transaktion

IMS-Anwendung		**UTM-Anwendung**
		APRO DM KCOF=B, KCRN=A
		MPUT NE, KCRN=A
		PEND KP
	<-- FMH5 + Daten	
GU IOPCB		
ISRT IOPCB		
	Daten -->	
		MGET NT *VGST=O,* KCRN=A
		MPUT NE, KCRN=A
		PEND KP
	<-- Daten	
GU IOPCB		
ISRT IOPCB		
Exit		
	Daten + CEB -->	
		MGET NT *VGST=C,* KCRN=A

In diesem Beispiel sind nur die IMS-DL/I-Aufrufe zum Senden und Empfangen von Nachrichten aufgeführt. Um Daten über mehrere Dialgoschritte hinweg verfügbar zu halten, arbeiten IMS-DL/I-Programme typischerweise mit einer Scratchpad-Area. Zum Lesen und Schreiben der Scratchpad-Area sind weitere GU-, GN- oder ISRT-Aufrufe erforderlich.

5. Synchroner Aufruf einer IMS-Transaktion von openUTM mit Transaktionssicherung, erste Variante

IMS-Anwendung		**UTM-Anwendung**
		APRO DM KCOF=C, KCPI=>A
		MPUT NE, KCRN=A
		PEND KP
	<--FMH5 + Daten	
GU IOPCB		
ISRT IOPCB		
	Daten + RQD2-->	
		(Systemcode)
	<- +DR2	
		MGET NT *VGST=O,* KCRN=A
		PEND FI
	<-- Prepare	
	Req Commit-->	
	<-- Committed	
	Forget + CEB-->	

6. Synchroner Aufruf einer IMS-Transaktion von openUTM mit Transaktionssicherung, zweite Variante

IMS-Anwendung		UTM-Anwendung
		APRO DM KCOF=C, KCRN=A
		MPUT NE, KCRN=A
		PEND KP
	<--FMH5 + Daten	
GU IOPCB		
ISRT IOPCB		
	Daten + RQD2-->	
		(Systemcode)
	<- +DR2	
		MGET NT *VGST=O,* KCRN=A
		MPUT NE KCLM=0, KCRN=A
		CTRL PE
		PEND KP
	<-- Prepare	
	Req Commit-->	
		MGET NT *VGST=C,* KCRN=A
	<-- Committed	
	Forget + CEB-->	
		PEND FI

Im Beispiel 6 sieht das 3. Teilprogramm, ob IMS Commit oder Rollback gegeben hat.

Im Beispiel 5 erfährt das Teilprogramm von Commit oder Rollback in IMS nichts mehr, dies erfolgt allein unter Kontrolle vom UTM-Systemcode.

7. Synchroner Aufruf einer IMS-Transaktion von openUTM mit Transaktionssicherung, Rollback von openUTM

IMS-Anwendung		**UTM-Anwendung**
		APRO DM KCOF=C, KCRN=A
		MPUT NE, KCRN=A
		PEND KP
	<--FMH5 + Daten	
GU IOPCB		
ISRT IOPCB		
	Daten -->	
		MGET NT *VGST=O,* KCRN=A
		PEND FR
	<-- Backout	
	Quittung -->	

8. Synchroner Aufruf einer IMS-Transaktion von openUTM mit Transaktionssicherung, Rollback vom IMS-Programm

IMS-Anwendung		**UTM-Anwendung**
		APRO DM KCOF=C, KCRN=A
		MPUT NE, KCRN=A
		PEND KP
	<--FMH5 + Daten	
GU IOPCB		
ISRT IOPCB		
ROLB		
	FMH7 -->	
	Fehlerdaten -->	
	FMH7 + CEB -->	
		Rollback der UTM-Transaktion

5.2.6 Ablauf-Beispiele mit CPI-C-Programmen

1. Synchroner Aufruf eines IMS-CPI-C-Programms von openUTM ohne Transaktionssicherung

IMS-Anwendung		UTM-Anwendung
		APRO DM KCOF=B, KCRN=A
		MPUT NE, KCRN=A
		PEND KP
	<-- FMH5 + Daten	
Accept		
Receive_Data		
Receive_SEND		
Send_Data		
SRRCMIT (lokaler Syncpoint)		
Deallocate		
	Daten + CEB -->	
		MGET NT *VGST=C,* KCRN=A

Dieses Beispiel entspricht dem Beispiel 20 im Abschnitt „Beispiele für openUTM-CPI-C-Kommunikation“ auf Seite 154.

2. Synchroner Aufruf eines IMS-CPI-C-Programms von *open*UTM mit Transaktionssicherung

IMS-Anwendung		UTM-Anwendung
		APRO DM KCOF=C, KCPI=A
		MPUT NE KCRN=A
		PEND KP
	<--FMH5 + Daten	
Accept		
Receive DATA		
Receive SEND		
Send_Data		
Prep_To_Rcv		
	Daten -->	
		MGET NT KCRN=A *VGST=O,*

IMS-Anwendung		UTM-Anwendung
		PEND FI
	<-- Prepare	
Receive SYNCP-DEAL		
SRRCMIT		
	Req Commit-->	
		(UTM-Systemcode)
	<-- Committed	
	Forget ------->	
		PEND FI

Dieses Beispiel entspricht dem Beispiel 12 im Abschnitt „Beispiele für openUTM-CPI-C-Kommunikation“ auf Seite 154. Dort finden Sie weitere Beispiele.

5.2.7 Ablauf-Beispiele mit Standard-IMS-Transaktionen

1. Synchroner Aufruf eines IMS-Kommandos von openUTM (z.B. /DISPLAY APPC)

IMS-Command		UTM-Anwendung
		APRO DM KCOF=B "/DISPLAY", KCRN=A
		MPUT NE "APPC", KCRN=A
		PEND KP
	<-- FMH5 + Daten	
/DISPLAY APPC		
Ausgabe		
	Daten + CEB -->	
		MGET NT *VGST=C,* KCRN=A

Bemerkung: Mit dem Kommando "/DISPLAY APPC" kann man sich den Zustand aller APPC-Ressourcen ausgeben lassen.

2. Asynchroner Aufruf eines IMS-Kommandos von openUTM (z.B. /DISPLAY APPC)

IMS-Command		UTM-Anwendung
		APRO AM KCOF=B "/DISPLAY", KCRN=A
		FPUT NE "APPC", KCRN=A
		PEND FI
	<-- FMH5 + Daten + CEB	
	Quittung -->	
/DISPLAY APPC		
Ausgabe		
	FMH5 + Daten + CEB -->	
	<-- Quittung	
		TAC DFSCMD
		FGET
		PEND FI

Für diese Art der Kommunikation muss eine openUTM-LU62-Instanz verwendet werden, die mit dem Parameter UTM-ASYNC=YES generiert ist.

Bemerkung: Der TAC DFSCMD muss in der UTM-Anwendung als ASYNTAC generiert sein. Der Name DFSCMD ist nicht frei wählbar sondern wird von IMS vorgegeben.

3. Synchroner Aufruf des IMS Message Switch von openUTM

DFSAPPC		UTM-Anwendung
		APRO DM KCOF=B "DFSAPPC", KCRN=A
		MPUT NE, KCRN=A
		PEND KP
	<-- FMH5 + Daten	
Aufruf Message Switch		
	CEB -->	
		MGET NT *VGST=C,* KCRN=A

Bemerkung: Der IMS Message Switch erlaubt das Senden von asynchronen Nachrichten an beliebige LU6.2-Partner der IMS-Anwendung. Beispielsweise wird mit der Nachricht "DFSAPPC (TPN=REPORT LU=FRED) REP1" IMS veranlasst, die Nachricht "REP1" an den TP-Namen "REPORT" der Partner-LU "FRED" zu senden.

5.2.8 Ablauf-Beispiele mit IMS als Auftraggeber

1. Aufruf eines UTM-Asynchron-TACS von IMS ohne Transaktionssicherung

IMS-Anwendung		UTM-Anwendung
GU IOPCB		
CHNG ALTPCB		
ISRT ALTPCB		
PURG ALTPCB		
	FMH5 + Daten + CEB -->	
	<-- Quittung	
		INIT
		FGET
		PEND FI

2. Aufruf eines UTM-Dialog-TACS von IMS mit Transaktionssicherung

IMS-Anwendung		UTM-Anwendung
Allocate synclev 2		
Send_Data		
Prep_To_Rcv		
	FMH5 + Daten -->	
		INIT
		MGET *VGST=O*
		MPUT NE
		PEND KP
	<-- Daten	
Receive DATA		
Deallocate		
SRRCMIT		
	Prepare -->	
		INIT
		MGET *TAST=P*
		PEND FI

IMS-Anwendung		UTM-Anwendung
	<-- Req Commit	
	Committed -->	
	<-- Forget + CEB	

Transaktionsgesicherte Kommunikation mit IMS als Auftraggeber ist nur möglich, wenn die IMS-Anwendungsprogramme die Programmschnittstelle CPI-C verwenden.

5.3 IMS-Administration

Administrationskommandos werden im wesentlichen für folgende Aufgaben benötigt:

- Anzeige der Verbindungszustände
- Aufbau von LU6.2-Sessions zum Partner
- Auflösen von Exchange-Log-Name-Problemen

Zum Anzeigen der Verbindungszustände gibt es folgende IMS-Kommandos

/DISPLAY APPC,LU,ALL	Anzeige aller LUs
/DISPLAY APPC,TP,ALL	Anzeige der verwendeten LU6.2-TP-Namen
/DISPLAY APPC,CONV	Anzeige der laufenden Conversations
/DISPLAY APPC,UOR,ALL	Anzeige der laufenden 2-Phase-Commit-Transaktionen

Mit dem IMS-Kommando /ALLOCATE kann man Sessions zu LU6.2-Partnern aufbauen.

Das LU6.2-Protokoll enthält beim Sessionaufbau ein Exchange Log Name. Hierbei tauschen die beiden Partner ihre Log-Namen aus. Damit kann festgestellt werden, ob seit dem letzten Verbindungsabbau einer der beiden Partner mit einem Kaltstart seine Log-Records gelöscht hat. Hat ein Partner einen Kaltstart durchgeführt, der andere Partner aber nicht, so entsteht ein Warm-Cold-Mismatch. Dieser kann nur durch manuellen Eingriff aufgelöst werden. Bei openUTM-LU62 wird ein Warm-Cold-Mismatch als XLN-Zustand "err" angezeigt. Hat IMS einen Kaltstart durchgeführt, openUTM-LU62 jedoch nicht, so kann man dies durch Neustart der openUTM-LU62-Instanz mit Kaltstartoption beheben. Im umgekehrten Fall, wenn also openUTM-LU62 mit Kaltstart hochgefahren wurde, IMS aber mit Warmstart, ist ein Eingriff auf Seite von IMS notwendig.

6 openUTM-CICS-Kopplung über LU6.1

In diesem Kapitel werden die Aspekte der openUTM-CICS-Kopplung über LU6.1 sowie die zugehörige Generierung und Programmierung behandelt.

Die Generierung und Programmierung für eine openUTM-CICS-Kopplung ist für den CICS-Anwender weitgehend identisch mit einer IMS-CICS-Kopplung über LU6.1 (siehe „CICS for MVS/ESA Intercommunication Guide“ und „CICS for MVS/ESA Distributed Transaction Programming Guide“).

6.1 CICS-Definitionen für openUTM-CICS-Sessions

In diesem Abschnitt soll das Generieren von Verbindungen zwischen CICS und openUTM dargestellt werden.

Generierungen für CICS können über Makros oder online (RDO - Resource Definition Online) erfolgen. Bei neueren CICS-Versionen wird zunehmend nur noch die RDO Definition beschrieben. Deshalb wird diese hier zur Beschreibung benutzt.

Für eine Verbindung mit openUTM müssen über RDO definiert werden:

```
CONNECTION
SESSIONS
```

In der System Initialization Table (SIT) wird außerdem der Name der CICS-Anwendung angegeben.

DFHSIT APPLID=cics_netname

1-8 Zeichen langer Name, mit dem der SNA-Netzname für die CICS-Anwendung festgelegt wird. Dieser Name muss mit dem RNETNAME-Operanden in der korrespondierenden LPAP-Anweisung der UTM-Generierung übereinstimmen.

DEFINE CONNECTION

Mit dieser Definition wird das ferne System, in diesem Fall die UTM-Anwendung, beschrieben. Sie entspricht damit der LPAP-Anweisung in der UTM-Generierung.

```
DEFINE CONNECTION   connection_name
       GROUP        group_name
       NETNAME      netname
       ACCESSMETHOD VTAM
       PROTOCOL     LU61
       DATASTREAM   USER
       RECORDFORMAT U | VB
              ...
```

Hierbei haben die Operanden folgende Bedeutungen:

CONNECTION connection_name
: 1-4 Zeichen langer Name zur Bezeichnung der UTM-Anwendung. Dieser Name existiert nur innerhalb von CICS. Er muss in den CICS-Programmen beim ALLOCATE im Parameter SYSID angegeben werden, wenn das CICS-Programm einen UTM-Transaktionscode aufrufen will. Er muss auch in einer Session-Definition vorkommen.

GROUP group_name
: 1-8 Zeichen langer Name. Der CICS-Systemverwalter muss diesen Namen vergeben. Er kommt nur in den CICS-Definitionen vor.

NETNAME netname
: 1-8 Zeichen langer Name, unter dem die UTM-Anwendung im SNA-Netz bekannt ist. Dieser Name muss mit dem LNETNAME-Operanden in der korrespondierenden LPAP-Anweisung der UTM-Generierung übereinstimmen.

ACCESSMETHOD VTAM
: Zugriffsmethode VTAM.

PROTOCOL LU61
: Es wird eine LU6.1-Verbindung definiert.

DATASTREAM USER
: Die Benutzerdaten werden vollständig vom Anwenderprogramm definiert.

RECORDFORMAT U | VB
: Das Format muss mit der Satzform, wie sie von CICS geschickt wird, korrespondieren. Der Standardwert U (=unformatted) entspricht Sätzen ohne Längenfeld. Wird VB (=variable blocked) angegeben, dann müssen die von CICS bereitgestellten Sätze dem von LU6.1 definierten VLVB-Format entsprechen, d.h. sie müssen Längenfelder enthalten.

DEFINE SESSIONS

Hiermit wird **eine** LU6.1-Session definiert. Dies entspricht der LSES-Anweisung in der UTM-Generierung.

```
DEFINE SESSIONS     session_name
       GROUP        group_name
       CONNECTION   connection_name
       SESSNAME     loc_ses_name
       NETNAMEQ     rem_ses_name
       PROTOCOL     LU61
       RECEIVECOUNT 01 | leerzeichen
       SENDCOUNT    01 | leerzeichen
       SENDSIZE     ru_size
       RECEIVESIZE  ru_size
       AUTOCONNECT  NO | YES
       BUILDCHAIN   YES
       IOAREALEN    value1, value2
              ...
```

Die Operanden haben folgende Bedeutungen:

SESSIONS session_name
: Name der Sessions-Definition. Dieser Name kommt nur in den CICS-Definitionen vor.

GROUP group_name
: 1-8 Zeichen langer Name. Der CICS-Systemverwalter muss diesen Namen vergeben. Er kommt nur in den CICS-Definitionen vor. Es sollte derselbe Name wie in der zugehörigen Connection-Definition gewählt werden.

CONNECTION connection_name
: Name der zugehörigen CONNECTION Definition.

SESSNAME loc_ses_name
: 1-4 Zeichen langer Name mit dem der local half session qualifier festgelegt wird. Dieser Name muss mit dem RSES-Operanden in der korrespondierenden LSES-Anweisung der UTM-Generierung übereinstimmen.

NETNAMEQ rem_ses_name
: 1-8 Zeichen langer Name, mit dem der remote half session qualifier festgelegt wird. Dieser Name muss mit dem LSES-Namen in der korrespondierenden LSES-Anweisung der UTM-Generierung übereinstimmen.

PROTOCOL LU61
: Es wird eine LU6.1-Session definiert.

RECEIVECOUNT/SENDCOUNT

Hiermit wird bestimmt welcher Partner PLU und wer SLU ist. In der UTM-Anwendung werden die entsprechenden Angaben in der SESCHA-Anweisung gemacht.

RECEIVECOUNT 01
SENDCOUNT leerzeichen
Für diese Session ist die CICS-Anwendung PLU (primary logical unit) und contention loser. In der korrespondierenden SESCHA-Anweisung der UTM-Generierung muss PLU=Y,CONTWIN=N angegeben werden.

RECEIVECOUNT leerzeichen
SENDCOUNT 01
Für diese Session ist die CICS-Anwendung SLU (secondary logical unit) und contention winner. In der korrespondierenden SESCHA-Anweisung der UTM-Generierung muss PLU=N,CONTWIN=Y angegeben werden. Bei einer Kopplung über TRANSIT-CLIENT ist dies nicht zulässig.

SENDSIZE / RECEIVESIZE ru_size

Damit wird die maximale Größe der request units (RU) für diese Session festgelegt. Der Wert für RU-Size wird bestimmt durch die maximale Datenlänge einschließlich der Function-Management-Header (FMH). Die Function-Management-Header haben eine Größe zwischen 7 und 50 Bytes. Bei einer Kopplung über TRANSIT-CLIENT sollte hier jeweils der Wert 1024 gewählt werden.

AUTOCONNECT NO | YES

Die Verbindungen werden automatisch beim Start von CICS aktiviert (YES) oder nicht (NO).

BUILDCHAIN YES

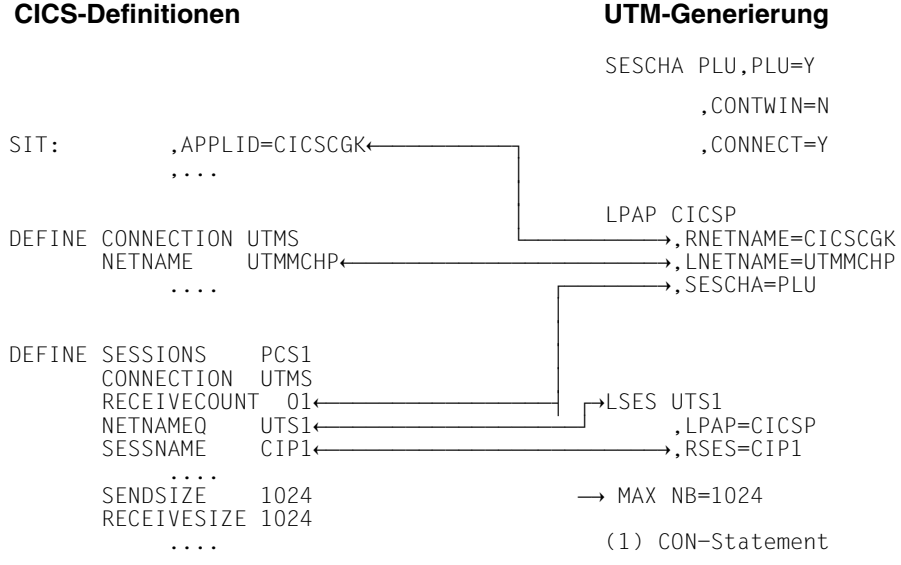
Dieser Operand muss für alle openUTM-CICS-Sessions angegeben werden.

IOAREALEN value1, value2

Mit *value1* wird die Mindestgröße der Terminal-Input-Output-Area von CICS festgelegt. Dieser Wert sollte mindestens so groß gewählt werden wie die mit maximale Nachrichtenlänge der Anwendungsprogramme.

Bei einer Kopplung über TRANSIT-CLIENT muss CICS immer PLU sein, d.h. RECEIVECOUNT = 1. Die maximale RU-size darf 1024 Bytes nicht überschreiten. Außerdem wird pro CONNECTION nur eine SESSION zugelassen (keine parallelen Sessions !). Wenn der Datentransfer über mehr als eine Session verteilt werden soll, so müssen mehrere CONNECTION definiert werden und die Auswahl der verschiedenen Sessions muss über das UTM- bzw. CICS-Anwendungsprogramm gesteuert werden.

Beispiel zu Abhängigkeiten von CICS- und UTM-Generierung



6.2 TRANSIT-Generierung für openUTM-CICS-Sessions

Das openUTM-Gateway 1 (UTMGW1) benutzt eine interne Schnittstelle von TRANSIT-SERVER. Dabei werden die LU6.1-Protokolle von openUTM transparent durchgereicht. Zusätzlich wird der Sessionauf- und Abbau und die Umwandlung des Transmission-Header (FID1 ← → FID2) behandelt.

Die Kommunikation mit dem UTM-System erfolgt über CMX. Die Kommunikation mit der IBM-Applikation erfolgt über TRANSIT-SERVER und - je nach Kopplungsart - über CMX oder direkt über den UNIX-Kern. Eine ausführliche Beschreibung des openUTM-Gateway finden Sie im Handbuch TRANSIT-CLIENT.

Eine ausführliche Beschreibung der verschiedenen Kopplungsarten zwischen TRANSIT-SERVER und dem IBM-System finden Sie im Handbuch TRANSIT-SERVER.

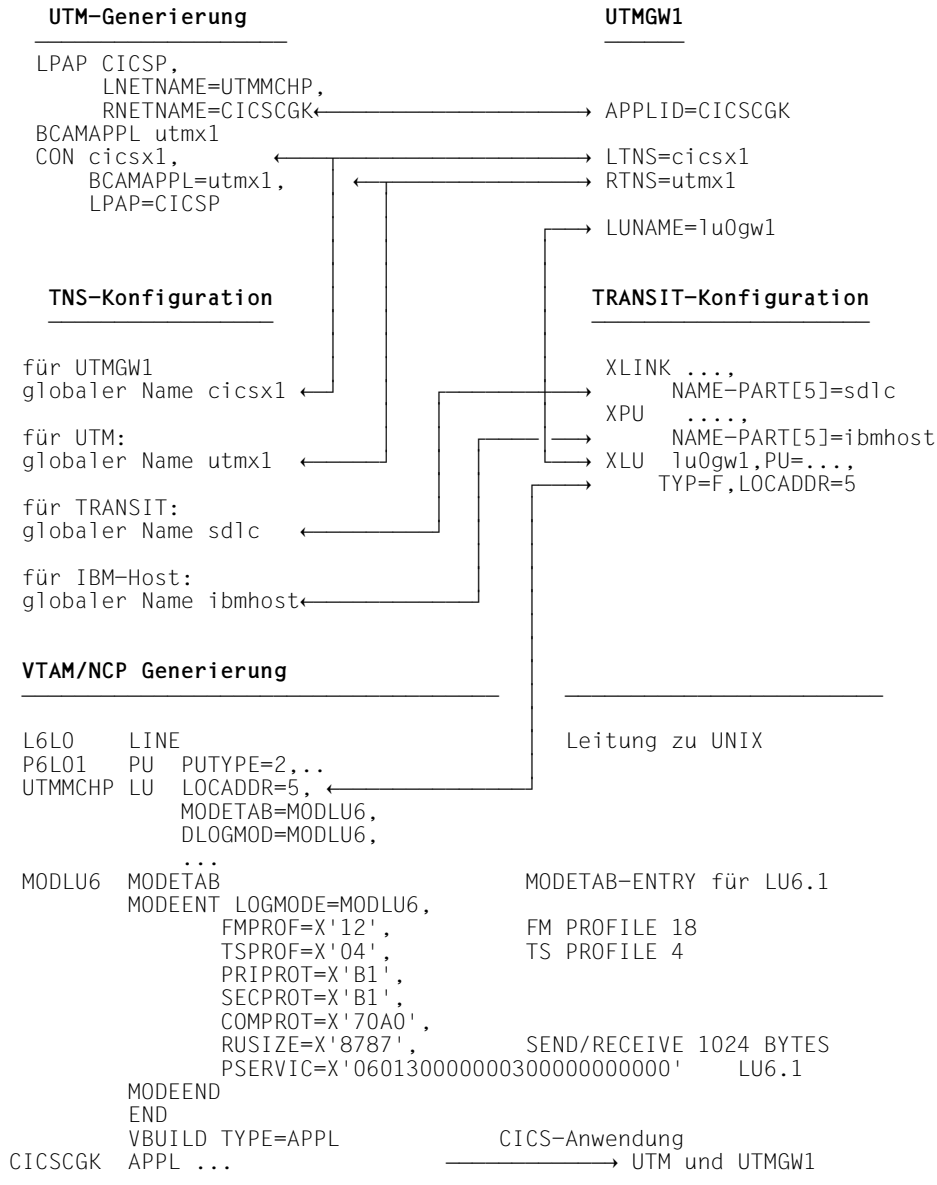
Bei einer Kopplung openUTM mit CICS werden folgende Komponenten durchlaufen.

```
openUTM → CMX → UTMGW1 → TRANSIT-SERVER → CMX → CCP → NCP/VTAM → CICS
```

Bei manchen Kopplungsarten entfallen CMX und CCP zwischen TRANSIT-SERVER und NCP/VTAM. Für alle diese Komponenten müssen Definitionen gemacht werden.

Die Abhängigkeit zwischen openUTM und CICS wurde im vorigen Kapitel beschrieben. Im folgenden Beispiel wird grob auf die Abhängigkeiten bei den UNIX-Definitionen eingegangen.

Das Beispiel gilt für eine Session, wobei openUTM und UTMGW1 in einem Rechner laufen.


UTM-Generierung
UTMGW1
LPAP CICSP,
LNETNAME=UTMMCHP,
RNETNAME=CICSCGK
APPLID=CICSCGK
BCAMAPPL utmx1
CON cicsx1,
LTNS=cicsx1
BCAMAPPL=utmx1,
RTNS=utmx1
LPAP=CICSP
LUNAME=luOgw1
TNS-Konfiguration
TRANSIT-Konfiguration
für UTMGW1
globaler Name cicsx1
XLINK ...,
NAME-PART[5]=sdlc
XPU ....,
für UTM:
globaler Name utmx1
NAME-PART[5]=ibmhost
XLU luOgw1,PU=...,
für TRANSIT:
globaler Name sdlc
TYP=F,LOCADDR=5
für IBM-Host:
globaler Name ibmhost
VTAM/NCP Generierung
L6LO LINE
Leitung zu UNIX
P6LO1 PU PUTYPE=2,..
UTMMCHP LU LOCADDR=5,
MODETAB=MODLU6,
DLOGMOD=MODLU6,
...
MODLU6 MODETAB
MODETAB-ENTRY für LU6.1
MODEENT LOGMODE=MODLU6,
FMPROF=X'12',
FM PROFILE 18
TSPROF=X'04',
TS PROFILE 4
PRIPROT=X'B1',
SECPROT=X'B1',
COMPROT=X'70A0',
RUSIZE=X'8787',
SEND/RECEIVE 1024 BYTES
PSERVIC=X'060130000000300000000000'
LU6.1
MODEEND
END
VBUILD TYPE=APPL
CICS-Anwendung
CICSCGK APPL ...
UTM und UTMGW1

6.3 Definieren von CICS-Transaktionen

6.3.1 Lokale Transaktionen

CICS-Transaktionen werden über RDO definiert.

```
DEFINE TRANSACTION <transaction code>
       PROGRAM     <program name>
       INDOUBT     WAIT
        .........
```

Die Operanden haben folgende Bedeutung.

TRANSACTION <transaction code>
: 1-4 Zeichen langer Transaktionscode für die CICS-Transaktion. Dieser Name muss mit dem RTAC-Operanden in der korrespondierenden LTAC-Anweisung der UTM-Generierung übereinstimmen.

PROGRAM <program name>
: Name eines CICS-Programms.

INDOUBT WAIT
: Diese Angabe ist nötig, um nach Sessionfehlern eine ordnungsgemäße Synchronisation mit openUTM zu erreichen.

6.3.2 Ferne Transaktionen

Die UTM-Transaktionen werden in CICS normalerweise nicht definiert. Der Transaktionscode erscheint nur in den Programmen. Ferne Dialog-Transaktionen werden durch den Aufruf BUILD ATTACH und Asynchron-Programme mit START festgelegt. Beim START-Aufruf darf der Transaktionscode jedoch nur 4 BYTES lang sein. Dies kann man in der UTM-Generierung berücksichtigen oder es wird diese Transaktion über RDO definiert.

```
DEFINE TRANSACTION  <local transaction code>
       REMOTESYSTEM <UTM-Systemid>
       REMOTENAME   <remote transaction code>
      .........
```

Die Operanden haben folgende Bedeutung.

TRANSACTION <local transaction code>
: 1-4 Zeichen langer lokaler Transaktionscode für einen UTM-Vorgang. Dieser Name wird im START-Aufruf verwendet.

REMOTESYSTEM <UTM-Systemid>
: Name der UTM-Anwendung in der CICS-Anwendung (vgl. Seite 210).

REMOTENAME <remote transaction code>
: 1-8 Zeichen langer Transaktionscode für einen Vorgang in der UTM-Anwendung. Dieser Name muss mit dem TAC-Namen in der korrespondierenden TAC-Anweisung der UTM-Generierung übereinstimmen.

Beispiel:

6.4 CICS-Programmierung bei Kopplung mit openUTM

Die Kommunikation zwischen CICS und openUTM kann auf zwei Arten erfolgen.

– Distributed Transaction Processing (DTP) LU6.1

– Asynchronous Processing LU6.1 mit START und RETRIEVE

Alle anderen Kopplungsmögichkeiten, die von CICS angeboten werden, können nicht verwendet werden (z.B. Distributed Program Link, Multiregion Operation).

Im folgenden soll ein grober Überblick über die Besonderheiten einer openUTM-CICS-Kommunikation vermittelt werden.

6.4.1 Regeln und Restriktionen bei CICS-Programmierung

Bei Erstellung von CICS-Programmen sind folgende Gesichtspunkte zu berücksichtigen:

1. Für openUTM sind eine ferne CICS- und eine ferne UTM-Anwendung nicht unterscheidbar. Mit anderen Worten: openUTM weiß nicht, ob es sich beim fernen Vorgang um einen CICS- oder einen UTM-Vorgang handelt.
2. Kommandos oder Kommandofolgen in CICS-Programmen müssen einem Aufruf oder einer Folge von Aufrufen an der KDCS-Programmschnittstelle entsprechen, damit sie mit UTM-Vorgängen kommunizieren können.
3. Am Ende einer CICS-Transaktion (Logical Unit of Work) werden nicht alle, sondern einzelne, ausgewählte Betriebsmittel gesichert. Am Ende einer UTM-Transaktion werden dagegen alle geänderten Betriebsmittel gesichert. Mit anderen Worten: openUTM arbeitet voll transaktionsorientiert und eine UTM-Sicherung ist "unteilbar". CICS-Vorgänge, die eine übergreifende Transaktionssicherung voraussetzen, müssen deshalb die Dialog- oder Asynchron-Nachricht als gesichert (PROTECTED) behandeln.
4. Eine Session zwischen einem UTM- und einem CICS-Vorgang ist an einer Sicherung beteiligt und darf nicht ohne Sicherungspunkt freigegeben werden.
5. In Bezug auf Beendigung einer Sessionfreigabe (Bracket) sind bei openUTM der Auftragnehmer- und der Auftraggeber-Vorgang nicht gleichberechtigt: Der Auftraggeber-Vorgang darf vor seinem Auftragnehmer-Vorgang nicht als erster (mit PEND FI) die Freigabe der Session einleiten. Deshalb darf ein CICS-Vorgang bei Zusammenarbeit mit einem UTM-Vorgang nur dann als erster die Sessionfreigabe einleiten, wenn er Auftragnehmer ist.

6. Indikatoren für Sicherungspunkte, Sessionbeendigung oder Wechsel in SEND- bzw. RECEIVE-Zustand werden bei openUTM in der Nachricht gesendet. CICS-Kommandos und ihre Kombinationen, die das Senden von isolierten Indikatoren, d.h. ohne Nachrichteninhalt an openUTM veranlassen können, dürfen nicht verwendet werden (z.B. WAIT-Option bei SEND in „CICS for MVS/ESA Application Programming Reference“).
7. Das Eintreffen des Senderechts (change direction) oder von "end bracket" wird von openUTM als Ende der Nachricht interpretiert. Erst danach wird sie als eine vollständige Nachricht angesehen und das Zielprogramm wird aktiviert. Das Hinzufügen des change direction Indikators zur Nachricht kann mit den CICS-Kommandos CONVERSE oder SEND INVITE erreicht werden. Das Hinzufügen des end bracket Indikators zur Nachricht kann mit den CICS-Kommandos SEND LAST, RETURN oder SEND LAST, SYNCPOINT erreicht werden.
8. Bei openUTM ist ein Sicherungspunkt immer mit dem Senden einer vollständigen Dialog-Nachricht verbunden. Ein CICS-Vorgang muss deshalb dafür Sorge tragen, dass beim SYNCPOINT-Kommando das Senderecht (change direction) an den UTM-Partner-Vorgang übertragen wird.
9. openUTM erwartet immer den Funktion Management Header ATTACH für die Adressierung von Vorgängen. Im CICS-Auftraggeber-Vorgang muss das CICS-Kommando BUILD ATTACH verwendet werden. Der UTM-Transaktionscode kann nicht an den Beginn der Nachricht gestellt werden.

6.5 CICS-Kommandos für CICS-Auftraggeber-Vorgänge

Im folgenden wird beschrieben, wie aus einer CICS-Anwendung ein UTM-Vorgang gestartet wird.

ALLOCATE

Mit diesem Kommando versucht ein CICS-Auftraggeber-Vorgang eine Session für einen Auftragnehmer-Vorgang zu belegen.

Das Kommando hat folgende Syntax:

```
EXEC CICS ALLOCATE SYSID(name)
                   [PROFILE(name)]
                   [NOQUEUE]
```

SYSID(name)
: spezifiziert den Namen für die ferne UTM-Anwendung, wie er in einer CONNECTION-Definition angegeben wurde. Unmittelbar nach der Belegung der Session ist der Name der belegten Session im Feld EIBRSRCE des exec interface blocks (EIB) zu finden. Dieser Name muss bei allen folgenden Auftraggeber-Kommandos (SEND/RECEIVE/CONVERSE), die sich auf diese Session beziehen, angegeben werden.

PROFILE(name)
: spezifiziert ein bestimmtes Profil für die Kommunikation mit dem Partner. PROFILEs werden mittels CICS-Definitionen (siehe „CICS for MVS/ESA Resource Definition“) festgelegt.

NOQUEUE
: bedeutet, dass nicht gewartet werden soll, falls nicht sofort eine Session verfügbar ist. Eine Session ist "nicht sofort verfügbar", wenn entweder

  1. alle Sessions zu der fernen UTM-Anwendung belegt sind, oder
  2. vorhandene Sessions noch nicht aktiv (gebunden) sind, oder
  3. die noch verfügbaren Sessions alle contention loser sessions (RECEIVECOUNT 1) sind.

  Wird NOQUEUE nicht verwendet, so hängt CICS den Request in eine Warteschlange ein und das Programm wartet bis eine Session verfügbar ist.

  Bei openUTM unter UNIX-Systemen darf NOQUEUE nicht verwendet werden, da CICS immer contention loser ist und somit keine Session sofort zur Verfügung steht.

BUILD ATTACH

Mit diesem Kommando wird vom CICS-Auftraggeber-Vorgang ein ATTACH Header (FMH5) erstellt, in den der TAC für den fernen UTM-Vorgang einzutragen ist. Dieser FMH ist für Initiierung von UTM-Vorgängen notwendig, und wird mit SEND oder CONVERSE an den Partner gesendet.

Das Kommando hat folgende Syntax:

```
EXEC CICS BUILD ATTACH  ATTACHID(name)
                        PROCESS(name)
                        RECFM(data-area)
```

ATTACHID(name)
: definiert den Namen des ATTACH Headers. Dieser Name wird beim SEND- oder CONVERSE-Kommando angegeben.

PROCESS(name)
: spezifiziert den UTM-Transaktionscode des fernen Vorgangs.

RECFM(data-area)
: data-area ist ein zwei Byte langes Feld, das das Satzformat der von CICS gesendeten Sätze enthält.

```
x'01'  Daten im VLVB-Format
x'04'  Daten im Format „Chain of RUs“
```

Andere Optionen (siehe „CICS for MVS/ESA Application Programming Reference“) dieses Kommandos sind bei der Kopplung mit openUTM ohne Bedeutung.

SEND

Dieses Kommando erstellt eine Dialogteilnachricht.

Das Kommando hat folgende Syntax:

```
EXEC CICS SEND [SESSION(name)]
               [ATTACHID(name)]
               FROM(bereich)
               LENGTH(bereich)
               INVITE
```

SESSION(name)
: spezifiziert den Namen der Session. Dieser ist als Rückgabewert aus dem ALLOCATE-Kommando bekannt (EIBRSRCE). Die Nachricht ist in diesem Fall für den Auftragnehmer (alternate facility) bestimmt. Wird die Option weggelassen, so ist die Nachricht für den Auftraggeber (principal facility) bestimmt.

ATTACHID(name)
: spezifiziert den Namen der ATTACH-ID, wie er im BUILD ATTACH-Kommando definiert wurde.

FROM(bereich) und LENGTH(bereich)
: spezifizieren die Adresse und die Länge der Nachricht. Das Format muss der Angabe bei RECFM beim BUILD ATTACH entsprechen. Wenn beim BUILD ATTACH VLVB angegeben wurde, muss ein 2 Byte langes Längenfeld am Beginn der Nachricht stehen. Mehrere Teilnachrichten können mit einem SEND gesendet werden, indem nach der ersten Teilnachricht erneut ein Längenfeld vor die zweite Teilnachricht positioniert wird. Wenn beim BUILD ATTACH „Chain of RUs“ angegeben wurde (Standardwert), dürfen keine Längenfelder am Beginn der Nachricht stehen.

INVITE
: veranlasst explizit, dass mit dem Senden der Nachricht das Senderecht (change direction) an den Partner übertragen wird. Das Eintreffen des Senderechts wird von openUTM als das Ende der Dialog-Nachricht des Partners interpretiert.

RECEIVE

Dieses Kommando wird verwendet, um eine von einem openUTM Vorgang gesendete Nachricht zu empfangen.

Das Kommando hat folgende Syntax:

```
EXEC CICS RECEIVE [SESSION(name)]
                  [INTO(bereich) | SET(pntr)]
                  LENGTH(bereich)
                  [MAXLENGTH(bereich)]
                  [NOTRUNCATE]
```

SESSION(name)
: spezifiziert den Namen der Session. Dieser ist als Rückgabewert aus dem ALLOCATE-Kommando bekannt (EIBRSRCE). Die Nachricht kommt in diesem Fall vom Auftragnehmer (alternate facility), also vom UTM-Anwendungsprogramm. Wird die Option weggelassen, so soll eine Nachricht vom Auftraggeber (principal facility) empfangen werden.

INTO(bereich)
: spezifiziert die Adresse des Speicherbereichs, in den CICS die Nachricht schreiben soll. Von UTM erhaltene Nachrichten enthalten normalerweise in den ersten beiden Bytes ein Längenfeld, d.h. openUTM verwendet normalerweise das VLVB-Format. Wenn das UTM-Anwendungsprogramm mehrere Teilnachrichten mit MPUT NT gesendet hat, werden diese in einem RECEIVE-Kommando empfangen. Jeder Teilnachricht ist dann ein Längenfeld vorangestellt. Wenn das UTM-Anwendungsprogramm beim MPUT KCDF ungleich 0 oder KCMF ungleich Leerzeichen gesetzt hat, so sendet openUTM jedoch vor der Nachricht einen Function-Management-Header 4 (FMH4), und die Daten ohne führendes Längenfeld. Mittels EXTRACT ATTACH kann das CICS-Programm erfragen, ob die Daten mit oder ohne Längenfeld ankommen. Genaueres zum EXTRACT ATTACH siehe Seite 225. Ob die angekommenen Daten einen FMH4 enthalten, kann das CICS-Programm an dem Indikator INBFMH nach Ausführung des RECEIVE-Kommandos erkennen. Genaueres zum FMH4 siehe ebenfalls Seite 225.

NOTRUNCATE
: Ist NOTRUNCATE nicht angegeben, so wird eine Nachricht bis zur Länge MAXLENGTH zugestellt. Der Rest wird verworfen. Ist NOTRUNCATE angegeben, so können weitere Teile mit RECEIVE angefordert werden.

Wegen der übrigen Parameter siehe „CICS for MVS/ESA Application Programming Reference“.

CONVERSE

Dieses Kommando bewirkt, dass eine Nachricht an den Partner gesendet und auf die Antwort des Partners gewartet wird. Dieses Kommando hat dieselbe Wirkung wie die Kommandofolge SEND INVITE, RECEIVE.

Das Kommando hat folgende Syntax:

```
EXEC CICS CONVERSE [SESSION(name)]
                   [ATTACHID(name)]
                   [FROM(bereich)]
                   FROMLENGTH(bereich)
                   [INTO(bereich) | SET(pntr)]
                   TOLENGTH(bereich)
                   [MAXLENGTH (bereich)]
                   [NOTRUNCATE]
```

Die Parameter haben die gleiche Bedeutung wie bei dem SEND- bzw. RECEIVE-Kommando beschrieben.

EXTRACT ATTACH

Dieses Kommando wird verwendet, um Informationen aus dem empfangenen ATTACH Header zu lesen (Indikator EIBATT im EIB nach RECEIVE).

Das Kommando hat folgende Syntax:

```
EXEC CICS EXTRACT ATTACH RECFM(data-area)
```

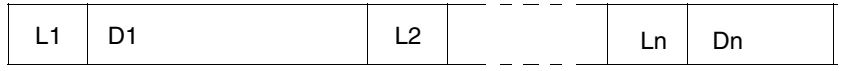
Ein von openUTM gesendeter FMH5 enthält nur Information über das verwendete Nachrichtenformat. In anderen Feldern (siehe „CICS for MVS/ESA Application Programming Reference“) werden nach dem Aufruf keine Informationen geliefert.

RECFM(data-area)

Dieser Parameter bezeichnet ein zwei Byte langes Feld, das in dem niederwertigen Byte Information für den Entblockungsalgorithmus im CICS-Anwenderprogramm enthält.

– Der Wert X’01’ bedeutet, dass die Daten im VLVB-Format (variable length, variable blocked) vorliegen. openUTM schickt die Daten im VLVB-Format, wenn KCDF=0 und KCMF=Leerzeichen beim MPUT gesetzt wurde.

  Jeder mit je einem MPUT NT/NE erzeugte Nachrichtenteil enthält ein 2 Byte langes Längenfeld.:



  MPUT NT MPUT NT MPUT NE

  Die Entblockung der Teilnachrichten muss im CICS-Programm vorgenommen werden.

– Der Wert X’04’ bedeutet, dass die Daten im Format „Chain of RUs“ vorliegen. openUTM schickt die Daten in diesem Format, wenn beim MPUT KCDF ungleich 0 oder KCMF ungleich Leerzeichen angegeben wurde.

  In diesem Fall setzt openUTM vor die Nachricht einen Function-Management-Header 4 (FMH4). Dieser FMH4 enthält die mit KCDF und KCMF gesendeten Informationen in folgender Weise:

```
Byte  0     X’12’ (Länge des FMH4)
Byte  1     X’04’ (Kennzeichen für FMH4)
Byte  2- 5  ohne Bedeutung
Byte  6     X’08’ (Länge des FMH4BN)
Byte  7-14  FMH4BN entspricht KCMF
Byte 15     X’02’ (Länge des FMH4BDT)
Byte 16-17  FMH4BDT entspricht KCDF
```

  Jeder mit MPUT NT/NE erzeugte Nachrichtenteil muss im CICS-Programm mit einem eigenen RECEIVE-Kommando abgeholt werden.

SYNCPOINT

Das Kommando wird

1. zum Anfordern eines Sicherungspunktes und
2. zum Bestätigen eines Sicherungspunktes

verwendet.

Die Kommandofolge SEND, SYNCPOINT fordert einen Sicherungspunkt an. Die Transaktion ist dann im Zustand PET (preliminary end of transaction).

Die Kommandofolge RECEIVE, SYNCPOINT bestätigt eine empfangene Aufforderung zum Setzen eines Sicherungspunkts (Indikator EIBSYNC im EIB). Die Transaktion ist dann im Zustand ET (end of transaction).

Das Kommando hat folgende Syntax:

```
EXEC CICS SYNCPOINT
```

RETURN

Das Kommando RETURN impliziert das Senden noch nicht gesendeter Nachrichten mit einem Sicherungsrequest an den Partner und anschließender Sessionfreigabe. Das RETURN-Kommando ist nur zulässig, wenn der UTM-Auftragnehmer-Vorgang bereits PEND FI gegeben hat (Indikator EIBFREE im EIB).

Das Kommando hat folgende Syntax:

```
EXEC CICS RETURN
```

6.6 CICS-Kommandos für CICS-Auftragnehmer-Vorgänge

RECEIVE

Dieses Kommando wird verwendet, um eine von einem UTM-Auftraggeber-Vorgang gesendete Nachricht zu empfangen. Unmittelbar nach der Initialisierung durch einen fernen UTM-Vorgang ist ein CICS-Vorgang im RECEIVE-Zustand und muss die Nachricht mit diesem Kommando einlesen.

Das Kommando hat folgende Syntax:

```
EXEC CICS RECEIVE {INTO(bereich) | SET(pntr)}
                  LENGTH(bereich)
                  [MAXLENGTH(bereich)]
                  [NOTRUNCATE]
```

Die Option NOTRUNCATE bewirkt, dass der Rest der Nachricht einem folgenden RECEIVE Kommando zur Verfügung gestellt wird, falls die Nachricht länger als MAXLENGTH ist. Zu den übrigen Optionen siehe Seite 223.

EXTRACT ATTACH

Dieses Kommando wird verwendet, um Informationen aus dem empfangenen ATTACH Header zu lesen (Indikator EIBATT im EIB).

Das Kommando hat folgende Syntax:

```
EXEC CICS EXTRACT ATTACH
          PROCESS(data-area)
          RECFM(data-area)
```

Ein von openUTM gesendeter FMH5 enthält den CICS-Transaktionscode (PROCESS-Option) und Information über das verwendete Nachrichtenformat. In anderen Feldern (siehe „CICS for MVS/ESA Application Programming Reference“) werden nach dem Aufruf keine Informationen geliefert.

PROCESS-Option
: enthält nach dem Aufruf den Transaktionscode wie er von openUTM an CICS im FMH5 gesendet wurde; d.h. den RTAC der UTM-Generierung.

RECFM-Option
: siehe Seite 225.

BUILD ATTACH

Mit diesem Kommando wird vom CICS-Auftragnehmer-Vorgang ein ATTACH Header (FMH5) erstellt, in dem openUTM das Nachrichtenformat für nachfolgende Nachrichten bekannt gemacht werden.

Das Kommando hat folgende Syntax:

```
EXEC CICS BUILD ATTACH  ATTACHID(name)
                        RECFM(data-area)
```

ATTACHID(name)
: definiert den Namen des ATTACH Headers. Dieser Name wird beim SEND- oder CONVERSE-Kommando angegeben.

RECFM(data-area)
: data-area ist ein zwei Byte langes Feld, das das Satzformat der von CICS gesendeten Sätze enthält.

```
x'01'  Daten im VLVB-Format
x'04'  Daten im Format „Chain of RUs“
```

Andere Optionen (siehe „CICS for MVS/ESA Application Programming Reference“) dieses Kommandos sind bei der Kopplung mit openUTM ohne Bedeutung.

SEND

Mit dem SEND-Kommando wird eine Dialogteilnachricht erstellt.

Das Kommando hat folgende Syntax:

```
EXEC CICS SEND  [ATTACHID(name)]
                FROM (bereich)
                LENGTH(bereich)
                [INVITE | LAST]
```

Die Nachricht ist für den Auftraggeber (principal facility) bestimmt.

Der Operand INVITE veranlasst explizit, dass mit dem Senden der Nachricht das Senderecht (change direction) an den Partner übertragen wird. Der Dialog wird dabei weitergeführt. openUTM muss darauf mit MGET-MPUT reagieren (kein PEND FI).

Werden vom CICS-Anwendungsprogramm Daten gesendet, die länger als die generierte RU-Size sind, so kommen diese beim UTM-Anwendungsprogramm in mehreren Teilnachrichten an. Soll das vermieden werden, so muss beim SEND der Parameter ATTACHID angegeben werden, der sich auf ein vorheriges BUILD ATTACH bezieht. In diesem BUILD ATTACH kann dann das Nachrichtenformat definiert werden.

CONVERSE

Das CONVERSE-Kommando wird verwendet, um eine Nachricht an den Partner zu senden mit gleichzeitigem Warten auf die Antwort. Dieses Kommando hat dieselbe Wirkung wie die Kommandofolge SEND INVITE,RECEIVE.

Das Kommando hat folgende Syntax:

```
EXEC CICS CONVERSE FROM(bereich)
                   FROMLENGTH(bereich)
                   [ATTACHID(name)]
                   {INTO(bereich) | SET(pntr)}
                   TOLENGTH(bereich)
                   [MAXLENGTH (bereich)]
                   [NOTRUNCATE]
```

SYNCPOINT

Siehe Seite 226.

RETURN

Das Kommando RETURN in einem CICS-Auftragnehmer-Vorgang impliziert das Senden noch nicht gesendeter Nachrichten mit einem Sicherungsrequest (impliziter Syncpoint für protected resources) an den Partner, und die anschließende Freigabe der Session.

Das Kommando hat folgende Syntax:

```
EXEC CICS RETURN
```

6.7 Vergleich mit KDCS-Aufrufen

Im folgenden soll versucht werden, die CICS-Kommandos in ihrer Semantik mit KDCS-Aufrufen zu vergleichen.

Adressierung eines fernen Dialogvorgangs

```
ALLOCATE                          APRO DM
BUILD ATTACH
```

Nachrichtenaustausch ohne Sicherungspunkt:

```
SEND                              MPUT
RECEIVE                           PEND KP
                                  .
                                  INIT
                                  MGET
```

Nachrichtenaustausch und Anforderung Transaktionsende:

```
SEND                              MPUT
SYNCPOINT                         PEND RE
RECEIVE                           .
                                  INIT
                                  MGET
```

Nachrichtenaustausch und Anforderung Transaktions- und Vorgangsende:

```
SEND                              MPUT
RETURN                            PEND FI
```

Bestätigung des Transaktionsendes nach Nachrichtenempfang:

```
                                  INIT
RECEIVE (EIBSYNC=X'FF')           MGET (KCTAST='P')
SEND                              MPUT
SYNCPOINT/RETURN                  PEND RE/FI
```

Wenn vom Vorgänger das Transaktionsende (ein Sicherungspunkt) angefordert wird, setzt CICS EIBSYNC=X'FF' und openUTM den Transaktionsstatus KCTAST='P'. Nach Empfang dieser Werte muss ein Befehl zum Schreiben eines Sicherungspunktes abgesetzt werden (SYNCPOINT/RETURN bzw. PEND RE/FI).

6.8 Programmbeispiele für CICS-openUTM-Kommunikation

1. Start eines UTM-Vorgangs aus einem CICS-Anwendungsprogramm

```
--->
    RECEIVE Client
    ALLOCATE
    BUILD ATTACH
    SEND INVITE SESSION(...)
    RECEIVE SESSION(...)
                              -- FMH5+Daten ---->
                                                    INIT
                                                    MGET
                                                    MPUT NE
                                                    PEND FI
                              <- FMH5+Daten+SRQ -
    Rückkehr des RECEIVE
    SEND Client
    RETURN
<---                          -- SRSP ---------->
```

2. Start eines CICS-Vorgangs aus einem UTM-Anwendungsprogramm

```
--->
    INIT
    MGET Client
    APRO DM KCPI=“>A“
    MPUT NE KCRN=“>A“
    PEND KP
                              -- FMH5+Daten ---->
                                                    RECEIVE
                                                    SEND LAST
                                                    RETURN
                              <- Daten+SRQ ------
    INIT
    MGET NT KCRN=“>A“
    MPUT NE Client
    PEND FI
<---                          -- SRSP ---------->
```

SRQ: Syncpoint Request
SRSP: Syncpoint Response

6.9 CICS-Kommandos für Asynchron-Aufträge

Das Senden und Empfangen von Asynchron-Aufträgen erfolgt bei CICS mit den Kommandos START und RETRIEVE.

START

Mit diesem Kommando wird ein Asynchron-Auftrag erstellt.

Das Kommando hat folgende Syntax:

```
EXEC CICS START
          [SYSID(name)]
          TRANSID(name)
          FROM(bereich)
          LENGTH(bereich)
          NOCHECK
          PROTECT
```

Dabei haben die Parameter folgende Bedeutungen:

SYSID(name)
: spezifiziert den Namen der UTM-Anwendung.

TRANSID(name)
: spezifiziert den Transaktionscode für das Asynchron-Progamm bei openUTM. Der Name darf bei CICS nicht länger als 4 Zeichen sein.

  Es kann aber auch ein lokaler TAC-Name für eine "remote transaction" bei CICS sein (siehe Seite 217). In diesem Fall wird die Option SYSID nicht verwendet, da die Partner-Anwendung durch die Generierung bereits festgelegt wurde.

NOCHECK
: Es wird keine Antwort erwartet. Wenn keine Session verfügbar ist, wird die Nachricht bei CICS in eine Warteschlange eingehängt. Falls wieder eine Session frei ist, wird die Nachricht gesendet.

PROTECT
: Der ferne Asynchron-Vorgang soll nicht gestartet werden, bevor der CICS-Vorgang einen SYNCPOINT oder ein RETURN-Kommando abgesetzt hat.

Die Unterstützung von RTRANSID und RTERMID ist ab Seite 249 beschrieben. Weitere Optionen wie TERMID und FMH (siehe „CICS for MVS/ESA Application Programming Reference“) werden von openUTM nicht unterstützt.

Innerhalb einer CICS-Transaktion (LUW = logical unit of work) darf nur ein Start-Kommando für eine UTM-Anwendung vorkommen.

RETRIEVE

Mit dem RETRIEVE-Kommando wird eine Asynchron-Nachricht gelesen.

Das Kommando hat folgende Syntax:

```
EXEC CICS RETRIEVE
          {INTO(bereich) | SET(pntr)}
```

Die Unterstützung von RTRANSID und RTERMID (siehe „CICS for MVS/ESA Application Programming Reference“) ist ab Seite 249 beschrieben.

Beispiele für den Austausch von Asynchron-Aufträgen

1. Start eines UTM-Asynchron-Auftrags aus einem CICS-Anwendungsprogramm

```
    START
    SYNCPOINT
                              -- FMH6+Daten+SRQ ->
                              <- SRSP ------------
                                                  INIT
                                                  FGET
                                                  PEND FI
```

2. Start eines CICS-Asynchron-Auftrags aus einem UTM-Anwendungsprogramm

```
    INIT
    APRO AM KCPI=“>A“
    FPUT NE KCRN=“>A“
    PEND FI
                              -- FMH6+Daten+SRQ ->
                              <- Daten+SRQ -------
                                                  RETRIEVE
                                                  RETURN
```

SRQ: Syncpoint Request
SRSP: Syncpoint Response

6.10 Hinweise zur openUTM-CICS-Programmierung

Für UTM-Vorgänge, die mit CICS-Vorgängen kommunizieren wollen, sind bezüglich der Programmierung folgende Hinweise zu beachten.

1. Angabe von KCMF (Formatname) und KCDF (Device-Features)

   Werden in UTM-Anwendungsprogrammen Formatname und/oder KCDF angegeben, so werden diese per FMH4 übertragen. Die FMH4-Behandlung muss im CICS-Anwenderprogramm vorgesehen werden.

2. Nachrichtenteile (MPUT NT/FPUT NT)

   Nachrichtenteile werden von openUTM normalerweise in VLVB-Records übertragen. Dabei werden alle Teile mit zugehörigem Längenfeld in einer oder mehreren RUs gesendet (je nach NB-Größe). CICS-Anwendungsprogramme müssen selbst für die Entblockung sorgen.

   Wenn im UTM-Anwendungsprogramm jedoch Formatname und/oder KCDF angegeben wird, so sendet openUTM die Nachrichtenteile im Format „Chain of RUs“.

3. Vorgangswiederanlauf

   openUTM geht davon aus, dass nach Störung der Kommunikation (z.B. Verbindungsverlust) die Verarbeitung am letzten Sicherungspunkt fortgesetzt wird.

   CICS kann diese Funktion nicht erbringen. Dies führt zum Vorgangsstatus ’Z’ im UTM-Anwendungsprogramm nach dem Wiederanlauf.

4. Konvertierung von Benutzerdaten

   Bei einer Kopplung zwischen openUTM unter UNIX- oder Windows-Systemen und CICS müssen Benutzerdaten von ASCII nach EBCDIC und umgekehrt konvertiert werden. Wenn die Benutzerdaten nur aus abdruckbaren Zeichen bestehen, kann man dies durch Verwendung des Parameters MAP=SYSTEM in der SESCHA-Anweisung von openUTM erreichen. Man beachte jedoch, dass dies eine Konvertierung von BS2000-EBCDIC nach ASCII und umgekehrt bewirkt. Auf IBM-Systemen wird jedoch meist ein anderes EBCDIC verwendet. Somit werden Umlaute und einige Sonderzeichen falsch umgesetzt. Es empfiehlt sich deshalb, die ASCII-EBCDIC-Konvertierung im UTM- oder CICS-Anwendungsprogramm durchzuführen. Aus den genannten Gründen sollte auch bei einer Kopplung zwischen UTM unter BS2000/OSD und CICS eine Konvertierung im Anwendungsprogramm vorgenommen werden.

Vereinbarungen bei der openUTM-CICS-Programmierung

Zwischen den Partnern openUTM und CICS müssen für die Programmierung folgende Punkte vereinbart werden.

- Namen der Transaktionen
- Datenformat, Satzformat (VLVB oder Chain of RUs)
- synchron(LU6.1) / asynchron
- Kommandofolge
- Werte im FMH6 (asynchron)

7 openUTM-IMS-Kopplung über LU6.1

In diesem Kapitel werden die Aspekte der openUTM-IMS-Kopplung über LU6.1 sowie die zugehörige Generierung und Programmierung behandelt.

Die Generierung und Programmierung für eine openUTM-IMS-Kopplung ist für den IMS-Anwender weitgehend identisch mit einer IMS-CICS-Kopplung (siehe „IMS/ESA Installation Volume 2: System Definition and Tailoring“ und „IMS/ESA Application Programming: Transaction Manager“).

7.1 IMS-Generierung für openUTM-IMS-Sessions

In diesem Abschnitt werden die wichtigsten IMS-Makros beschrieben, die zur Generierung einer openUTM-IMS-Kopplung erforderlich sind.

Makro COMM

Dieser Makro beschreibt die grundsätzlichen Anforderungen bezüglich der Kommunikation unabhängig vom Terminaltyp. Er muss genau einmal angegeben werden.

```
COMM    RECANY=(number,size),
        APPLID=username,
        EDTNAME=name,
        .........,
        andere Parameter
```

Hierbei haben die Operanden folgende Bedeutungen:

RECANY=(number,size)
: number spezifiziert die Anzahl der VTAM RECEIVE ANY Puffer, die im IMS-System zur Verfügung stehen. size spezifiziert die Größe dieser Puffer. Der Wert von size muss größer oder gleich dem Wert von TRMSGLTH der MAX-Anweisung der UTM-Generierung sein. Zulässige Werte für size sind dem Handbuch "IMS/ESA Installation Volume 2: System Definition and Tailoring" zu entnehmen.

APPLID=username
: 1-8 Zeichen langer Name, unter dem die IMS-Anwendung im SNA-Netz bekannt ist (SNA-netname). Dieser Name muss mit dem RNETNAME-Operanden in der korrespondierenden LPAP-Anweisung der UTM-Generierung übereinstimmen.

EDTNAME=name
: 1-8 Zeichen langer Name, unter dem der ISC-Edit-Prozess im IMS-System bekannt ist. Dieser Name muss mit dem DPN-Operanden in der korrespondierenden SESCHA-Anweisung der UTM-Generierung übereinstimmen.

Makro TYPE

Der Makro dient dazu, den Beginn der Beschreibung einer Gruppe von VTAM-Terminals des gleichen Typs einzuleiten. Ihm folgen die TERMINAL- und NAME-Makros für die Terminals der Gruppe.

```
TYPE    UNITYPE=LUTYPE6,
        ...,
        andere Parameter
```

Der Operand hat folgende Bedeutung:

UNITYPE=LUTYPE6
: Es wird ein LU6.1-Gerätetyp definiert.

Makro TERMINAL

Mit diesem Makro werden physikalische und logische Eigenschaften eines VTAM-Nodes des Typs definiert, der in dem vorausgehenden TYPE-Makro angegeben wurde.

```
TERMINAL NAME=nodename,
         MSGDEL=SYSINFO,
         OPTIONS=FORCSESS,
         COMPT1=(SINGLE1 [,DPM-Bn],IGNORE),
         SEGSIZE=size,
         OUTBUF=buffer size,
         SESSION=n,
         ...........,
         andere Parameter
```

Die Operanden haben folgende Bedeutungen:

NAME=nodename
: 1-8 Zeichen langer Name, mit dem der SNA-Netzname (SNA netname) für die UTM-Anwendung festgelegt wird. Dieser Name muss mit dem LNETNAME-Operanden in der korrespondierenden LPAP-Anweisung der UTM-Generierung übereinstimmen.

MSGDEL=SYSINFO
: spezifiziert, welche Message-Typen nicht gesendet werden. Für die Kopplung mit openUTM muss SYSINFO angegeben werden.

OPTIONS=(TRANRESP,FORCSESS)
: FORCSESS spezifiziert den Session-Wiederanlauf. Für die Kopplung mit openUTM muss FORCSESS angegeben werden.

  TRANRESP ist für den Dialog mit IMS-Transaktionen erforderlich (Standard).

COMPT1=(SINGLE1[,DPM-Bn],...)
: SINGLE1 spezifiziert bei Ausgabe das Bracket-Protokoll. Für die Kopplung mit openUTM muss SINGLE1 angegeben werden.

  DPM-Bn gibt an, ob MFS (Distributed Presentation Management) zur Verfügung steht. Wird dieser Parameter nicht angegeben, wird VLVB angenommen. Eine zu sendende Nachricht an ein UTM-Anwendungsprogramm muss dann in den ersten 8 Zeichen den UTM-Transaktionscode enthalten (eventuell mit Blanks am Ende). Dem UTM-Anwendungsprogramm wird die komplette Nachricht im Nachrichtenbereich zur Verfügung gestellt (inklusive Transaktionscode).

SEGSIZE=size
: Size gibt die Größe eines Eingabesegmentes an. Der Wert muss mindestens so groß sein wie die maximale Nachrichtenteillänge in den UTM-Anwendungsprogrammen.

OUTBUF=buffer size
: buffer size beschreibt die Größe des Ausgabe-Puffers. Dieser Wert muss kleiner oder gleich dem Wert TRMSGLTH der MAX-Anweisung der UTM-Generierung sein. Zulässige Werte für buffer size sind dem Handbuch „IMS/ESA Installation Volume 2: System Definition and Tailoring“ zu entnehmen.

SESSION=n
: n bestimmt die Anzahl der parallelen Sessions eines Nodes. Für openUTM unter UNIX-Systemen sind keine parallelen Sessions möglich (n=1).

Makro VTAMPOOL

Dieser Makro muss angegeben werden, wenn mit parallelen Sessions gearbeitet werden soll. Er steht am Beginn der Definition eines oder mehrerer LU6-Subpools. Diese werden über SUBPOOL-Makros definiert, von denen wiederum jeder von einem oder mehreren NAME-Makros gefolgt werden kann.

```
VTAMPOOL
```

Der Makro hat keine Operanden.

Makro SUBPOOL

Mit diesem Makro wird ein LU6-Subpool definiert. Für jede Session, die dynamisch aufgebaut werden soll, wird ein SUBPOOL-Makro benötigt.

```
SUBPOOL NAME=subpool-name,
        MSGDEL=SYSINFO
```

Die Operanden haben folgende Bedeutung:

NAME=subpool-name
: subpool-name korrespondiert mit dem RSES-Operanden der zugehörigen LSES-Anweisung in der UTM-Generierung.

MSGDEL=SYSINFO
: Der Parameter muss identisch dem MSGDEL-Operanden des TERMINAL-Makros gewählt werden.

Makro NAME

Mit diesem Makro wird zu einem physikalischen Terminal ein logischer Terminalname definiert. Mit diesem Namen wird im IMS-Programm die UTM-Session adressiert.

Durch Angabe des NAME-Makros nach dem SUBPOOL-Makro wird bestimmt, dass das logische Terminal dynamisch angelegt werden soll. Dies ist erforderlich, wenn mit parallelen Sessions gearbeitet werden soll (TERMINAL SESSION=n, mit n>1).

```
NAME     lterm,
         COMPT=1,
         .........,
         weitere Parameter
```

Die Operanden haben folgende Bedeutung:

lterm definiert einen 1-8 Zeichen langen logischen Terminalnamen.

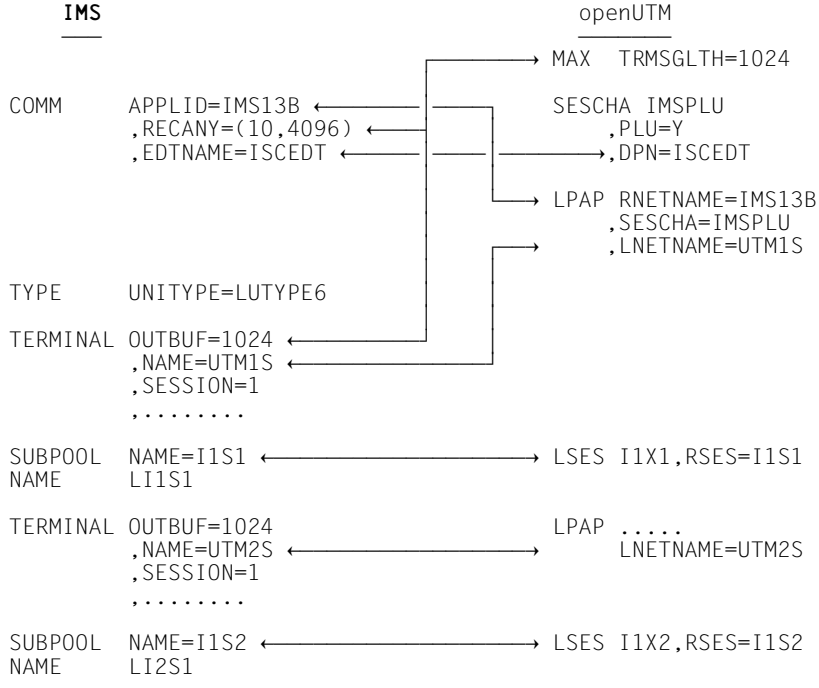
COMPT=1
: spezifiziert die Ausgabekomponenten für dieses Terminal. Der Wert 1 korrespondiert mit dem COMPTn-Operanden des TERMINAL-Makros.

7.1.1 Beispiele zu Abhängigkeiten bei der openUTM-/IMS-Generierung

Für die mit Pfeilen gekennzeichneten Parameter müssen Vereinbarungen zwischen dem IMS- und dem UTM-Anwender getroffen werden.

Beispiel: openUTM unter UNIX-Systemen oder openUTM unter BS2000/OSD bei Verwendung von TRANSIT-CLIENT

IMS ist PLU, openUTM ist SLU, 2 Sessions



Sessionaufbau

Der LSES-Name (z. B. I1X1) wird im IMS nur bei einer Initialisierung der Session durch den Operator mit Hilfe des /OPN-Kommandos benötigt.

Z.B. wird die Session mit dem SUBPOOL I1S1 mit folgendem Aufruf aufgebaut:

```
/OPN NODE UTM1S SUBPOOL I1S1 ID I1X1
```

Die Zuordnung zwischen TERMINAL und SUBPOOL ist bei IMS nicht fest, sondern erfolgt dynamisch anhand der Daten des Verbindungsaufbaus.

7.2 TRANSIT-Generierung für openUTM-IMS Sessions

Die Aussagen zu TRANSIT, die zur openUTM-CICS-Kopplung gemacht wurden, gelten analog für die openUTM-IMS-Kopplung.

Siehe hierzu Seite 214.

7.3 Generieren von IMS-Transaktionen

Makro TRANSACT

Mit diesem Makro werden IMS-Transaktionen definiert. Er entspricht der TAC-Anweisung in der UTM-Generierung.

```
TRANSACT CODE=transaction-code,
         MSGTYPE=(MULTSEG,RESPONSE),
         .........
```

Die Operanden haben folgende Bedeutung.

CODE=transaction-code
: 1-8 Zeichen langer Name für die IMS-Transaktion. Dieser Name muss mit dem RTAC-Operanden in der korrespondierenden LTAC-Anweisung der UTM-Generierung übereinstimmen.

MSGTYPE=(MULTSEG,RESPONSE)
: MULTSEG definiert, dass eine Eingabe-Nachricht aus mehreren Segmenten bestehen kann. RESPONSE wird für Dialog-Programme angegeben, NONRESPONSE für Asynchron-Programme.

Der Parameter SPA darf nicht verwendet werden.

Beispiele

```
LTAC IMSDIA1P                        TRANSACT
     ,RTAC=DIALOG1 ←──────────────→   CODE=DIALOG1
     ,TYPE=D                          ,MSGTYPE=(MULTSEG,RESPONSE)
     ,.........                       ,........

LTAC IMSASY1S                        TRANSACT
     ,RTAC=ASYNC1  ──────────────→    CODE=ASYNC1
     ,TYPE=A                          ,MSGTYPE=(MULTSEG,NONRESPONSE)
     ,.........                        ,........
```

7.4 IMS-Programmierung bei Kopplung mit openUTM

Im Folgenden soll ein grober Überblick einer openUTM-IMS-Kommunikation vermittelt werden.

7.4.1 Kopplungsmöglichkeiten mit IMS

Synchrone Aufträge (Dialog)

IMS erlaubt auf Grund seiner Struktur (message queues) keine synchronen Transaktionen, wenn IMS Auftraggeber (front-end) ist, d.h. wenn eine Anfrage vom Terminal von IMS an openUTM geht und eine synchrone Antwort von openUTM erwartet wird.

Wenn IMS der Auftragnehmer (back-end) ist, kann ein Dialog zwischen openUTM und IMS aufgebaut werden. Ob die IMS Transaktion synchron oder asynchron abläuft, wird im IMS festgelegt durch den Parameter MSGTYPE im TRANSACT-Makro (RESPONSE oder NONRESPONSE). Außer dieser internen Spezifikation kann die Art des Ablaufes auch über die Form der ankommenden Meldung bestimmt werden. Wird von openUTM ein Dialogverkehr gestartet (MPUT) , so antwortet IMS synchron, auch wenn im TRANSACT-Makro NONRESPONSE angegeben wurde.

Asynchron-Aufträge

Bei Kopplungen mit IMS wird auf Grund der Einschränkung des Dialogs oft der asynchrone Datenverkehr verwendet. Die Synchronisierung der Antworten auf entsprechende Anfragen wird dem Benutzer überlassen. Es gibt aber Möglichkeiten, das Routing der Antworten zu automatisieren. Dabei wird das Ziel der Antwort bei der Anfrage mitgeliefert und so eine Möglichkeit für einen Pseudodialog geschaffen. Die Informationen für das Routing werden im function management header 6 (FMH6) hinterlegt und müssen vom Partner ausgewertet werden. Der Zugriff auf die Daten im FMH6 von openUTM aus wird im Kapitel Pseudodialoge zwischen Asynchron-Vorgängen beschrieben.

7.4.2 IMS-Aufrufe

IMS-Programme senden und empfangen Nachrichten durch folgende Aufrufe:

GU <I/O-PCB>,<I/O-area>
: empfängt die erste oder einzige Teilnachricht in dem Bereich <I/O-area>.

GN <I/O-PCB>,<I/O-area>
: empfängt gegebenenfalls weitere Teilnachrichten in dem Bereich <I/O-area>.

CHNG <Alt-PCB>,<destination name>
: ändert die Zieladresse in <destination name>
(lterm aus NAME-Makro der Generierung).

ISRT <I/O-PCB>/<Alt-PCB>,<I/O-area>,<MOD-Name>
: sendet Nachrichten aus dem Bereich <I/O-area>.

RETURN
: beendet die Transaktion und fordert einen Sicherungspunkt an.

Beim Empfangen und Senden einer Nachricht wird ein PCB (program communication block) übergeben, in dem Informationen wie Status, Partnername (lterm) usw. abgespeichert werden. Wird eine Nachricht an den Partner gesendet, der eine Transaktion aufgerufen hat, so wird der I/O-PCB benutzt. Im anderen Fall wird ein alternate-PCB benutzt, bei dem vorher mit dem Aufruf CHNG eine neue Zieladresse gesetzt wird.

Der Transaktionscode wird vor die Nachricht gesetzt oder über einen MOD-Namen (message output description) im MFS (message format service) definiert.

7.4.3 Vergleich mit KDCS-Aufrufen

- Die Aufrufe GU und GN auf den IOPCB entsprechen den KDCS-Aufrufen MGET/FGET. Der Aufruf GN liefert eine Anzeige, die aussagt, ob weitere Nachrichten-Teile vorhanden sind.
- Eine Folge von MPUT/FPUT NT und MPUT/FPUT NE muss von IMS mit GU und GN gelesen werden.
- Der Aufruf ISRT entspricht den KDCS-Aufrufen MPUT (IOPCB) und FPUT (alternate PCB).
- Der Aufruf ROLL entspricht dem KDCS-Aufruf PEND FR.
- Der Aufruf ROLLB entspricht dem KDCS-Aufruf PEND RS.
- Programmende impliziert eine Sicherungspunkt-Anforderung und entspricht PEND FI.

7.4.4 Beispiele für die openUTM-IMS Kommunikation

1. Start eines IMS-Auftrags aus einem UTM-Anwendungsprogramm

```
--->
    INIT
    MGET Client
    APRO DM KCPI=“>A“
    MPUT NE KCRN=“>A“
    PEND KP
                              -- FMH5+Daten ---->
                                                  GU       (IO/PCB)
                                                  ISRT
                                                  RETURN
                              <- Daten+SRQ ------
    INIT
    MGET NT KCRN=“>A“
    MPUT NE Client
    PEND FI
<---                          -- SRSP ---------->
```

2. Dasselbe, jedoch mit PEND RE im UTM-Anwendungsprogramm

```
--->
    INIT
    MGET Client
    APRO DM KCPI=“>A“
    MPUT NE KCRN=“>A“
    PEND RE
                              -- FMH5+Daten+SRQ ->
                                                  GU       (IO/PCB)
                                                  ISRT
                                                  RETURN
                              <- Daten+SRSP ------
                              <- SRQ -------------
    INIT
    MGET NT KCRN=“>A“
    MPUT NE Client
    PEND FI
<---                          -- SRSP ---------->
```

SRQ: Syncpoint Request
SRSP: Syncpoint Response

Die zweite Variante unterscheidet sich von der ersten in der Logik der Transaktionssicherung. In der ersten Variante besteht der gesamte Ablauf aus einer einzigen Transaktion. In der zweiten Variante gehören eventuelle Datenbankänderungen nach dem MGET KCRN=>A nicht zur selben Transaktion wie die auf IMS-Seite durchgeführten Datenbankänderungen.

Sollen auf beiden Seiten Datenbankänderungen durchgeführt werden, so muss die zweite Variante gewählt werden. Kommt es bei der ersten Variante nach dem MGET KCRN=>A zu einem Rücksetzen der Transaktion auf UTM-Seite (z. B. durch PEND ER), so ist IMS nicht mehr in der Lage, die Datenbankänderungen zurückzusetzen.

7.4.5 Beispiele für den Austausch von Asynchron-Aufträgen

1. Start eines UTM-Asynchron-Auftrags aus einem IMS-Anwendungsprogramm

```
ISRT   (Alt.-PCB)
RETURN
                          -- FMH6+Daten+SRQ ->
                          <- SRSP ------------
                                              INIT
                                              FGET
                                              PEND FI
```

2. Start eines IMS-Asynchron-Auftrags aus einem UTM-Anwendungsprogramm

```
INIT
APRO AM KCPI=“>A“
FPUT NE KCRN=“>A“
PEND FI
                          -- FMH6+Daten+SRQ ->
                          <- Daten+SRQ -------
                                              GU    (IO/PCB)
                                              RETURN
```

SRQ: Syncpoint Request
SRSP: Syncpoint Response

7.4.6 Hinweise zur openUTM-IMS-Programmierung

Wenn IMS Auftraggeber ist, wird der Name der aufzurufenden UTM-Transaktion normalerweise mit den Daten gesendet. Der UTM-Anwender muss das im Eingabebereich berücksichtigen.

Ähnlich wie bei einer Kopplung mit CICS muss auch bei einer Kopplung mit IMS das Problem der Datenkonvertierung beachtet werden. Siehe hierzu Seite 234.

Weitere Hinweise zum Austausch von Asynchron-Aufträgen sind ab Seite 249 beschrieben.

Vereinbarungen bei der openUTM-IMS-Programmierung

Zwischen den Partnern openUTM und IMS müssen für die Programmierung folgende Punkte vereinbart werden.

- Namen der Transaktionen
- Art der Übergabe des Transaktionscodes
- Datenformat (VLVB oder Chain of RUs)
- synchron/asynchron
- Werte im FMH6 (asynchron)

8 LU6.1-Pseudo-Dialoge zwischen Asynchron-Vorgängen

Die Programmierung einer LU6.1-Kopplung zwischen openUTM und IMS ist oft schwierig, da von IMS aus keine synchronen Aufträge in einer UTM-Anwendung gestartet werden können. Das LU6.1-Protokoll bietet aus diesem Grund die Möglichkeit, Pseudo-Dialoge zwischen Asynchron-Vorgängen zu bilden. Mittels spezieller Erweiterungen der KDCS, die in den normalen UTM-Handbüchern nicht beschrieben sind, können diese Pseudo-Dialoge auch in UTM-Anwendungen verwendet werden, und zwar bei Kopplungen openUTM-IMS und openUTM-CICS.

Bei LU 6.1 erfolgt die Adressierung von Asynchron-Vorgängen im Prinzip mit dem SCHEDULER Function Management Header FMH6. Dabei sind folgende Felder von Bedeutung:

DPN Destination Process Name
- Transaktionscode in der Auftragnehmer-Anwendung (openUTM, CICS).
- Name eines Editors, der die Nachricht auswertet oder ein Formatname in der Auftragnehmer-Anwendung (IMS).

PRN Primary Resource Name
- Transaktionscode bzw. LTERM-Name in der Auftragnehmer-Anwendung (IMS).
- LTERM-Name in der Auftragnehmer-Anwendung (CICS).

RDPN Return Destination Process Name
- DPN in der lokalen Anwendung

RPRN Return Primary Resource Name
- PRN in der lokalen Anwendung

Die Felder RDPN/RPRN im FMH6 können benutzt werden, um einen Pseudodialog zwischen Asynchron-Vorgängen zu führen. Ein Auftraggeber sendet eine Nachricht mit RDPN/RPRN an einen asynchronen Auftragnehmer-Vorgang, der das Ergebnis an den durch RDPN/RPRN spezifizierten Asynchron-Vorgang in der Auftraggeber-Anwendung sendet.

Bei CICS können die Felder RDPN/RPRN mit den Parametern RTRANSID/RTERMID beim START-Kommando belegt werden. Mit dem RETRIEVE-Kommando erhält ein CICS-Anwendungsprogramm in RTRANSID/RTERMID die Werte von RDPN/RPRN aus dem Eingabe-FMH.

Die Belegung und Auswertung der Daten im FMH6 wird von IMS teilweise automatisch oder mit der Komponente Message Format Service (MFS) erfüllt.

Bei openUTM gibt es spezielle Funktionsaufrufe für UTM-Anwendungsprogramme, mit denen das Senden einer Nachricht an das in RDPN/RPRN angegebene Ziel in der Auftraggeber-Anwendung sowie die Belegung von RDPN und RPRN bei Ausgabe-Nachrichten möglich ist.

8.1 Programmschnittstelle für openUTM

Mit dem Funktionsaufruf

```
INFO GN (get names)
```

erhält ein von einer fernen Anwendung gestarteter Asynchron-Vorgang die Werte von RDPN/RPRN aus dem Eingabe-FMH.

Mit dem Funktionsaufruf

```
APRO IN (insert names)
```

wird ein Asynchron-Vorgang in einer fernen Anwendung adressiert, wobei die Felder DPN, PRN, RDPN und RPRN für den Ausgabe-FMH explizit belegt werden können. Der APRO IN ersetzt in diesem Fall den APRO AM.

Für die Aufrufe muss neben dem KDCS-Parameterbereich ein Datenbereich angegeben werden.

Der Datenbereich hat folgende Struktur:

Feldname	**Bedeutung**	
KCDPN	Destination Process Name	Ziel in der fernen Anwendung
KCPRN	Primary Resource Name	
KCRDPN	Return Destination Process Name	Ziel in der lokalen Anwendung
KCRPRN	Return Primary Resource Name	

Die Felder sind je 8 Bytes lang.

Beim Aufruf APRO IN muss der Datenbereich vom UTM-Anwendungsprogramm mit Werten versorgt werden. Nach einem erfolgreichen INFO GN sind die Felder durch openUTM bereits richtig für eine Antwort in einem Pseudodialog belegt.

INFO GN

Die folgende Tabelle zeigt die notwendigen Angaben im KDCS-Parameterbereich bei INFO GN:

KCOP	KCOM	KCLA
"INFO"	"GN"	32

Alle anderen Parameterfelder müssen auf binär null gesetzt werden.

COBOL-Format des Aufrufs:

```
CALL "KDCS" USING <parm1>, <parm2>
```

<parm1>: KDCS-Parameterbereich
<parm2>: Datenbereich

openUTM gibt folgendes zurück:

- Bei KCRCCC = 000 sind die Felder KCRDPN und KCRPRN im Datenbereich <parm2> mit Leerzeichen belegt, KCDPN und KCPRN enthalten das Ziel in der anderen Anwendung aus der Eingabe-Nachricht.
- KCRLM enthält die Länge des Datenbereichs (32), bei KCRCCC>=40Z ist die Länge 0.
- Das Rückgabefeld KCRCCC kann einen der folgende Werte enthalten:

  000 Die Funktion wurde ausgeführt.
  40Z Die Funktion kann nicht durchgeführt werden.
  – die Namen sind nicht vorhanden (KCRCDC enthält KD20).
  41Z Der Aufruf ist an dieser Stelle nicht erlaubt.
  – INFO GN wurde in einem Dialogvorgang gegeben.
  – INFO GN wurde in einem Vorgang gegeben, der nicht von einer fernen Anwendung gestartet wurde.
  42Z KCOM ist ungültig.
  43Z KCLA ist ungültig
  47Z Die Adresse des Datenbereichs fehlt bzw. ist ungültig.
  49Z Parameterfelder, die von diesem Aufruf nicht verwendet werden, sind nicht auf binär null gelöscht.
  71Z In dem UTM-Anwendungsprogrammlauf wurde noch kein INIT gegeben.

Nach einem erfolgreichen INFO GN können die im Datenbereich <parm2> enthaltenen Werte benutzt werden, um dem Partnerprogramm in CICS oder IMS eine Antwort zu schikken. Das Partnerprogramm muss dann mit APRO IN und Leerzeichen in KCRN adressiert werden, siehe unten.

APRO IN

Die folgende Tabelle zeigt die notwendigen Angaben im KDCS-Parameterbereich bei APRO IN:

KCOP	KCOM	KCLM	KCRN	KCPA	KCPI
"APRO"	"IN"	32	LTAC-name	LPAP-name oder SPACES	Vorgangs-Identifikation
			Leerzeichen	LPAP-Name	

Alle anderen Parameterfelder müssen auf binär null gesetzt werden.

Im Datenbereich müssen alle Felder belegt werden, Leerzeichen bedeutet, dass der Name nicht angegeben wurde. Enthält ein Feld im Datenbereich nicht Leerzeichen, so wird es von openUTM als gültiger Name akzeptiert. Es wird keine Prüfung durchgeführt, ob ein Feld nur alphanumerische Zeichen enthält oder ob z.B. KCRDPN/KCRPRN ein Ziel in der lokalen Anwendung ist.

Für die Adressierung des Auftragnehmers mit APRO IN gibt es folgende Möglichkeiten:

1. Adressierung durch KCRN (LTAC) wie beim APRO AM
   In diesem Fall müssen KCDPN und KCPRN Leerzeichen enthalten.
2. Adressierung durch KCDPN und KCPRN
   KCRN muss Leerzeichen enthalten, KCPA den LPAP-Namen. Diese Art der Adressierung ist nach einem erfolgreichen INFO GN sinnvoll.
3. Adressierung durch die Nachricht
   KCRN, KCDPN und KCPRN enthalten Leerzeichen, KCPA enthält den LPAP-Namen. Die ersten 8 Zeichen beim FPUT/DPUT enthalten den Transaktionscode.

COBOL-Format des Aufrufs:

```
CALL "KDCS" USING <parm1>, <parm2>
```

<parm1>: KDCS-Parameterbereich

<parm2>: Datenbereich

Rückgaben von openUTM:

Das Rückgabefeld KCRCCC kann neben den im Handbuch „openUTM Anwendungen programmieren mit KDCS für COBOL, C und C++“ beschriebenen Returncodes einen der folgenden Werte enthalten:

44Z Bei der Adressierung über KCDPN und/oder KCPRN enthält KCRN nicht Leerzeichen (KCRCDC enthält KD21)

47Z Die Adresse des Datenbereichs fehlt bzw. ist ungültig.

49Z Parameterfelder, die von diesem Aufruf nicht verwendet werden, sind nicht gelöscht.

Erfolgt die Adressierung nicht über KCRN, so werden DPN und PRN im Ausgabe-FMH von openUTM mit den Werten aus dem Datenbereich versorgt. Die Generierung von DPN bei der SESCHA-Anweisung wird nicht berücksichtigt.

Mit dem Aufruf APRO IN gehen Datenschutzfunktionen verloren, falls die Adressierung nicht mit KCRN erfolgt.

Für den Datenbereich gibt es keine von openUTM erstellten Datenstrukturen.

8.2 Übergabe der FMH6-Parameter in openUTM, IMS und CICS

Die Belegung- und Auswertungsmöglichkeiten der Parameter im FMH6 in den verschiedenen Systemen zur Adressierung des Partners wird hier noch einmal genauer beschrieben.

8.2.1 Behandlung des Transaktionscodes in openUTM

Der Transaktionscode kann auf unterschiedliche Weise an openUTM übermittelt werden.

1. im Feld DPN des FMH6
2. im Feld PRN des FMH6
3. in den ersten 8 Bytes der Daten

openUTM ermittelt den Transaktionscode bei Empfang vom fernen System auf folgende Art:

1. Wert in DPN, falls belegt und keine Angabe in SESCHA DPN=
2. Wert in PRN, falls belegt und Angabe SESCHA DPN=name
3. aus den ersten 8 Bytes der Daten

openUTM schickt den Transaktionscode an das ferne System auf folgende Art:

1. wenn mit APRO IN adressiert wird, Belegung entsprechend dem Parameterbereich des Aufrufs (siehe APRO IN und INFO GN)
2. Im Feld DPN, falls keine Angabe in SESCHA DPN=
3. Im Feld PRN, falls Angabe SESCHA DPN=name.
   Dann steht in DPN des FMH6 der Name aus SESCHA DPN

8.2.2 Behandlung des Transaktionscodes in IMS

IMS schickt den Transaktionscode normalerweise in den Daten. DPN ist mit dem Namen des ISC-Edit (iscedt-name) belegt. Für den UTM-Anwender heißt das:

Angabe von SESCHA DPN=iscedt-name und Berücksichtigung des Transaktionscodes bei der Definition des Eingabebereiches. Der Transaktionscode wird von openUTM nicht aus den Daten entfernt. Wird die Angabe DPN=iscedt-name nicht gemacht, so kann als TAC auch der iscedt-name verwendet werden. Dieser kann dazu benutzt werden, um z.B. zunächst gemeinsame Funktionen für mehrere Transaktionen durchzuführen, ehe die eigentliche Transaktion (Name aus den Daten) zum Ablauf kommt.

Als Antwort auf eine asynchrone Anfrage verhält sich IMS wie openUTM nach einem INFO GN. Die Parameter RDPN/RPRN der Eingabe werden bei der Ausgabe durch IMS auf die Felder DPN/PRN gesetzt.

Soll von IMS ein Pseudodialog gestartet werden, so muss das Feld RPRN mit dem Transaktionscode des Empfängers in IMS belegt werden. Standardmäßig enthält dieses Feld den Namen des aufrufenden Terminals oder Programmes. Andere Werte müssen über MFS belegt werden.

8.2.3 Behandlung des Transaktionscodes in CICS

CICS kann die einzelnen Felder bei Asynchronaufträgen individuell durch Parameter im START-Aufruf belegen.

```
START TRANSID(T1)      ⟶ DPN
      TERMID (T2)      ⟶ PRN
      RTRANSID(T3)     ⟶ RDPN
      RTERMID(T4)      ⟶ RPRN
```

CICS schickt den Transaktionscode normalerweise im Feld DPN.

CICS empfängt Daten von openUTM mit dem RETRIEVE Kommando. Der TAC wird aus dem Feld DPN entnommen.

```
RETRIEVE ...
      RTRANSID(T1)     ⟵ RDPN
      RTERMID(T2)      ⟵ RPRN
```

9 Fehlerdiagnose

In diesem Kapitel finden Sie die Hinweise zur Fehlerdiagnose.

9.1 Diagnosehilfsmittel

Bei Kommunikationsproblemen zwischen der UTM-Anwendung und der Partner-Anwendung im IBM-System sollten Sie folgende Diagnosehilfsmittel nutzen:

1. Das Kommando `u62_sta` zum Überprüfen des Status von openUTM-LU62. Dieses Kommando ist auf Seite 58 beschrieben.
2. Bei Verwendung von TRANSIT das Kommando `SNA_status` zum Überprüfen des Status von TRANSIT. Dieses Kommando ist im Handbuch TRANSIT-SERVER beschrieben.
3. Bei Verwendung von SNAP-IX oder des IBM Communications Server die Status-Anzeige-Funktionen dieses Produkts.
4. Die SYSLOG-Datei der UTM-Anwendung. Siehe hierzu die entsprechenden UTM-Handbücher.
5. Bei Verwendung von openUTM-LU62 für UNIX-Systeme die Diagnosedateien
   `/opt/lib/utmlu62/PROT/prot.`*luname*
   Bei Verwendung von openUTM-LU62 für Windows-Systeme die Diagnosedateien
   `Programme\utmlu62\PROT\prot.`*luname*`.txt`
   Die in diesen Dateien vorkommenden Meldungen sind ab Seite 264 beschrieben.
6. Bei Verwendung von openUTM-LU62 mit TRANSIT die Diagnosedateien
   `/opt/lib/transit/PROT/prot.nukleus`
   `/opt/lib/transit/PROT/prot.lu62`
   `/opt/lib/transit/PROT/cnos.mlog`
   Siehe hierzu die TRANSIT-Handbücher.
7. Bei Verwendung von SNAP-IX oder des IBM Communications Server die Protokollanzeigefunktion dieses Produkts.

8. Bei Verwendung der LU6.1-Kopplung mit TRANSIT die Diagnosedateien
   `/opt/lib/transit/PROT/prot.nukleus`
   `/opt/lib/transit/PROT/prot.utmgw1`
   Die Meldungen sind in den TRANSIT-Handbüchern beschrieben.
9. Das JES-Job-Log mit den Konsolmeldungen von CICS oder IMS. Siehe hierzu die entsprechenden CICS- bzw. IMS-Handbücher.
10. Der BCAM-Trace von openUTM, der SNA-Trace von TRANSIT-SERVER oder der entsprechende Trace von SNAP-IX bzw. des IBM Communications Server. Das Vorgehen zum Erstellen eines SNA-Trace ist im Handbuch TRANSIT-SERVER sowie in der Dokumentation zu SNAP-IX und IBM Communications Server beschrieben. Um diese Traces lesen zu können, sind Grundkenntnisse über SNA-Protokolle und bei LU6.2 Grundkenntnisse über OSI-TP-Protokolle erforderlich. Bei einer LU6.1-Kopplung kann es vorkommen, dass openUTM im Fehlerfall eine negative Response oder einen Function Management Header 7 (FMH7) sendet. Beide enthalten ein 4 Byte langes Feld mit einer verschlüsselten Fehlerinformation, den sogenannten Sense-Data. Die von openUTM gesendeten Sense-Data sind ab Seite 259 beschrieben.
11. Bei Verwendung von openUTM-LU62 der auf Seite 66 beschriebene Protokolltrace.

Die von openUTM-LU62 angelegten Diagnosedateien im Dateiverzeichnis `/opt/lib/utmlu62/PROT` (UNIX-Systeme) bzw. `Programme\utmlu62\PROT` (Windows-Systeme) werden beim Neustart einer Instanz von openUTM-LU62 teilweise gelöscht. Sichern Sie also die Diagnosedateien rechtzeitig.

Beim Starten einer Instanz von openUTM-LU62 werden im Dateiverzeichnis `utmlu62/PROT` folgende Diagnosedateien gelöscht:

`in.dump.`*luname*
`xaplog.`*luname*`.*`
`xap.dump.`*luname*`.*`
`prot.`*luname*`.old`
`inlog.`*luname*`.*.old`
`core.luname`

Die Dateien `prot.`*luname* und `inlog.`*luname*`.*` werden mit dem Suffix `.old` versehen. Auf Windows-Systemen hat die Datei `prot.`*luname* zusätzlich das Suffix `.txt`.

9.2 LU6.1-Sense-Data

Im folgenden Abschnitt sind die Sense-Data beschrieben, die von openUTM bei einer LU6.1-Kopplung gesendet werden:

0006 0000	LUSTAT NOOP
0007 0000	LUSTAT QUEUE EMPTY
0809 xxxx	MODE INCONSISTENCY
080B 0000	BRACKET RACE ERROR
0812 0000	INSUFFICIENT RESOURCE. Betriebsmittelengpass, z.B. Pagepool voll.
0813 0000	BRACKET BID REJECT. Eine Sessionbelegungsanforderung von der Contention-Loser-Anwendung wurde abgelehnt, da die Contention-Winner-Anwendung die Session für einen Auftrag belegt hat.
0814 0000	BRACKET BID REJECT, RTR FOLLOWS
0819 0000	RTR NOT REQUIRED
081B 0000	TRANSMISSION RACE. Eine Sessionbelegungsanforderung von der Contention-Loser-Anwendung wurde abgelehnt, da die Contention-Winner-Anwendung die Session für einen Auftrag belegt hat.
082D 0000	LU BUSY
0835 xxxx	INVALID PARAMETER. xxxx ist die Adresse des fehlerhaften Parameters innerhalb des empfangenen Protokollelements.
0846 0000	FMH7 FOLLOWS
084B 0000	REQUESTED RESOURCES NOT AVAILABLE, LPAP NOT ALLOWED
084D xxxx	INVALID BIND
0864 0101	FUNCTION ABORT. LUSTAT CMD erhalten, und CDI=0 und ERI=0.
0864 0104	FUNCTION ABORT. Eine Dialog-Nachricht wurde mit BB empfangen. Die Nachricht kann nicht verarbeitet werden, weil z.B. – der TAC ungültig ist, – der Pagepool voll ist, – die Nachricht mit EB empfangen wurde.
0864 0106	FUNCTION ABORT. Die Länge der empfangenen Nachricht ist ungültig, oder das Längenfeld einer Nachricht im VLVB-Format ist ungültig.
0864 0107	FUNCTION ABORT. Die Gesamtlänge der empfangenen Nachrichtenteile ist zu groß (größer als 32767).

0864 0108	FUNCTION ABORT. End of Chain, aber kein Nachrichten(teil)ende. D.h. das im VLVB-Format erhaltene Längenfeld zeigt eine zu große Länge an.
0864 C5D9	FUNCTION ABORT. PEND ER vom UTM-Anwendungsprogramm
0864 C5E2	FUNCTION ABORT. PEND ER durch openUTM
0864 D9E2	FUNCTION ABORT. PEND RS durch openUTM. Z. B.: Der lokale Vorgang wurde beendet und die letzte Eingabenachricht enthält CD oder DR2.
0864 D9E4	FUNCTION ABORT. PEND RS vom UTM-Anwendungsprogramm
0866 D9E2	FUNCTION ABORT REC RESPONSE. PEND RS durch openUTM
0866 D9E4	FUNCTION ABORT REC RESPONSE. PEND RS vom UTM-Anwendungs-programm
0867 0000	SYNC EVENT RESPONSE
1002 0000	INVALID LENGTH
1003 0000	INVALID TAC. Ungültiger oder gesperrter Transaktionscode.
1008 0106	INVALID FM HEADER. Eine Nachricht mit FMH wurde empfangen, aber der FMH ist nicht vollständig in der Nachricht enthalten.
2001 0000	SEQENCE NUMBER ERROR
2002 0000	CHAINING ERROR
2003 0000	BRACKET ERROR
2004 0000	DIRECTION ERROR
2005 0000	DATA TRAFFIC RESET
2007 0000	DATA TRAFFIC NOT RESET
2008 0000	NO BEGIN BRACKET
2009 0000	SESSION CONTROL PROTOCOL VIOLATION
200A 0000	IMMEDIATE REQUEST MODE ERROR
200B 0000	QUEUED RESPONSE ERROR
200C 0000	ERP SYNC EVENT ERROR
200D 0000	RESPONSE OWED BEFORE SENDING REQUEST
4001 0000	INVALID SC OR NC RH
4003 0000	BB NOT ALLOWED
4004 0000	CEB OR EB NOT ALLOWED
4006 0000	EXCEPTION RESPONSE NOT ALLOWED

4007 0000	DEFINITE RESPONSE NOT ALLOWED
4009 0000	CD NOT ALLOWED
400A 0000	NO-RESPONSE NOT ALLOWED
400B 0000	CHAINING NOT SUPPORTED
400C 0000	BRACKETS NOT SUPPORTED
400D 0000	CD NOT SUPPORTED
400F 0000	INCORRECT USE OF FORMAT INDICATOR
4010 0000	ALTERNATE CODE NOT SUPPORTED
4011 0000	RU CATEGORY NOT CORRECT
4012 0000	REQUEST COED NOT CORRECT
4013 0000	SDI/RTI NOT CORRECT
4014 0000	DR1I/DR2I/ERI NOT CORRECT
4015 0000	QRI NOT CORRECT
4016 0000	EDI NOT CORRECT
4017 0000	PDI NOT CORRECT

10 Meldungen von openUTM-LU62

Dieses Kapitel enthält alle Meldungen des Produktes openUTM-LU62 sowie deren Bedeutung und eventuell durchzuführende Maßnahmen.

Meldungen des Programms *u62_tp* werden auf UNIX-Systemen in die Datei

`/opt/lib/utmlu62/PROT/prot.`*luname*

auf Windows-Systemen in die Datei

`Programme\utmlu62\PROT\prot.`*luname*`.txt`

ausgegeben. Die Dienstprogramme `u62_gen`, `u62_adm` und `u62_sta` geben ihre Meldungen nach `stderr` aus.

Auf Windows-Systemen werden zusätzliche Meldungen in die Dateien

```
Programme\utmlu62\PROT\stdout.txt
Programme\utmlu62\PROT\stderr.txt
```

ausgegeben, sofern openUTM-LU62 als Service bei jedem Systemstart gestartet wird.

Alle Meldungen können in deutscher oder englischer Sprache ausgegeben werden. Die Steuerung erfolgt in UNIX-Systemen über die Umgebungsvariable `$LANG`. Beginnt der Wert von `$LANG` mit den Buchstaben 'De', so werden die Meldungen in deutscher Sprache ausgegeben. In allen anderen Fällen werden englische Meldungen erzeugt. Auf Windows-Systemen werden deutsche Meldungen ausgegeben, wenn die Ländereinstellungen im System auf deutsch gestellt sind, ansonsten englische Meldungen.

In diesem Handbuch werden die Meldungstexte in deutscher Sprache angegeben. Namen in Großbuchstaben, die mit einem '&' beginnen (z. B. &FILENAME), dienen als Platzhalter für variable Elemente in Meldungstexten.

10.1 Meldungen des Programms u62_tp

Im folgenden Abschnitt sind alle Meldungen des Programms `u62_tp` beschrieben.

Die Meldungen des Programms `u62_tp` werden auf UNIX-Systemen in die Datei

`/opt/lib/utmlu62/PROT/prot.`*luname*

auf Windows-Systemen in die Datei

`Programme\utmlu62\PROT\prot.`*luname*`.txt`

ausgegeben, wobei *luname* der jeweilige Name der LU ist. Alle Meldungen beginnen mit der Zeichenfolge

`u62_tp` [*komp*] *nnn*

Hiebei bezeichnet *komp* die Komponente des Programms `u62_tp` und *nnn* die Meldungsnummer.

Es gibt folgende Komponenten:

BD Boundary (Behandlung der Schnittstellen APPC und XAP-TP sowie der Schnittstelle zum System)

DM Dialogue Manager (Behandlung der OSI-TP-Dialoge und LU6.2-Conversations)

RS Resync Service Transaction Program (Wiederaufsetzen in der Startphase und nach Leitungsausfall)

TM Transaction Manager (Behandlung der Transaktionssicherung)

In der Datei `prot.`*luname* stehen außerdem noch Meldungen von `u62_start`. Diese sind mit dem Vorspann `u62_start` versehen und ab Seite 343 beschrieben.

Meldungen der Komponente BD

002 Aufruf:
u62_tp -l <LU-Name> [-c|-k] [-t on[,<Trace-Optionen>]]
-l LU-Name: Name der lokalen LU der openUTM-LU62-Instanz
-c: Kaltstart von openUTM-LU62
-k: „Lauwarmer“ Start von openUTM-LU62
-t on: Explizite Spezifikation des Trace durch Angabe von Optionen moeglich (durch Komma getrennt):
IN=[<Level>]: mit Instanz-Trace
XAP: mit XAP-TP-Provider Trace

Das Programm `u62_tp` wird intern von `u62_start` aufgerufen. Bei fehlerhafter Parameterversorgung durch `u62_start` erscheint diese Meldung.

003 Wechsel ins Basis-Directory &DIRNAME nicht moeglich,
errno &ERRNO (&ERRTEXT)

Das Programm `u62_tp` kann nicht in das Arbeitsverzeichnis *&DIRNAME* wechseln. Die genauere Ursache wird durch den Inhalt der Systemvariablen `errno` = *&ERRNO* angegeben. *&ERRTEXT* enthält einen kurzen Text, der den Fehler genauer beschreibt.

004 Fehler bei Signalbehandlung,
errno &ERRNO (&ERRTEXT)

Es ist ein interner Fehler bei der Initialisierung der Signalbehandlung aufgetreten. Der Inhalt der Systemvariablen `errno` = *&ERRNO* und der kurze Text in *&ERRTEXT* geben genaueren Aufschluss über die Fehlerursache.

006 Die Konfigurationsdatei &FILENAME kann nicht geoeffnet werden,
errno &ERRNO (&ERRTEXT)

007 Fehler beim Einlesen der Konfigurationsdatei &FILENAME,
errno &ERRNO (&ERRTEXT)

008 Die Version von u62_tp stimmt nicht mit der Konfigurationsdatei &FILENAME ueberein (&VERS1 statt &VERS2)

Die vom Generierungsprogramm `u62_gen` erzeugte Konfigurationsdatei *&FILENAME* enthält eine falsche Versionsnummer. Entweder handelt es sich bei *&FILENAME* um gar keine Konfigurationsdatei oder es wurde eine inkompatible Version von `u62_gen` zur Generierung verwendet.

009 Der lokale LU-Name &LUNAME ist nicht konfiguriert.

Die lokale LU *&LUNAME*, für die die Instanz gestartet wurde (Schalter -l), ist nicht in der Konfigurationsdatei enthalten.

010 Die Instanz fuer den lokalen LU-Namen &LUNAME laeuft bereits.

Die Instanz für die lokale LU *&LUNAME* ist bereits gestartet worden. Für eine lokale LU darf nur eine einzige Instanz laufen.

011 Nicht genug Speicherplatz vorhanden

Die Instanz beendet sich, weil der Versuch, temporären Speicher zu besorgen, schiefgegangen ist. Diese Meldung erscheint nur während der Initialisierungsphase.

012 Interner Fehler aufgetreten

Während der Initialisierung ist ein interner Fehler aufgetreten. Die Instanz beendet sich daraufhin. Sie sollten Diagnoseunterlagen sichern.

013 Version: &VERSION &DATE &VERSIONID
System: &SYSTEM (&SYSTEMVERSION)
Rechnername: &SYSTEMNAME

Beim Start gibt die Instanz Informationen zur Programmversion von `u62_tp` und zur Systemumgebung aus. Diese Informationen sind besonders bei der Fehlerdiagnose von Bedeutung. Im Einzelnen enthalten die Variablen folgenden Inhalt:
&VERSION enthält die Versionsnummer des Programms `u62_tp`,
&DATE ist das Erstellungsdatum des Programms `u62_tp`,
&VERSIONID enthält eine interne Kennung des Programms `u62_tp`,
&SYSTEM ist der Name des Betriebssystems (z.B. SunOS),
&SYSTEMVERSION bezeichnet die Betriebssystem-Version,
&SYSTEMNAME ist der Host-Name des lokalen Rechners.

015 Fehler beim Erzeugen der PID-Datei &PIDFILE,
errno &ERRNO (&ERRTEXT)

Die Instanz kann eine ihrer intern verwendeten Systemdateien (*&PIDFILE*) nicht anlegen. Die Systemvariable `errno` = *&ERRNO* gibt Aufschluss über die Ursache.

016 Fehler beim Schreiben in die PID-Datei &PIDFILE,
errno &ERRNO (&ERRTEXT)

Es kann keine Information in der intern verwendeten Datei *&PIDFILE* abgelegt werden. Vermutlich ist das Dateisystem voll.

017 Fehler beim Schreiben in die PID-Datei &PIDFILE
geschrieben: &NUM1 Bytes, erwartet: &NUM2 Bytes

Der Versuch, *&NUM2* Bytes in die Datei *&PIDFILE* zu schreiben, ist fehlgeschlagen. Wahrscheinlich aufgrund eines vollen Dateisystems konnten von den *&NUM2* Bytes nur *&NUM1* Bytes abgelegt werden.

018 Fehler beim Erzeugen der Input Pipe &PIPEFILE,
errno &ERRNO (&ERRTEXT)

Der Versuch, die Named-Pipe *&PIPEFILE* anzulegen, ist fehlgeschlagen. Die Fehlerursache ist in der Variable `errno` enthalten. Da die Input-Pipe nur einmal, und zwar während der Initialisierung angelegt wird, kann diese Meldung nur unmittelbar nach dem Start erscheinen. Da in diesem Fall keine Kommunikation mit den übrigen Komponenten von openUTM-LU62 möglich ist, beendet sich die Instanz.

019 Fehler beim Oeffnen der Input Pipe &PIPEFILE,
errno &ERRNO (&ERRTEXT)

Die soeben erzeugte Named-Pipe *&PIPEFILE* kann nicht zum Lesen geöffnet werden. Ansonsten gilt das Gleiche wie oben in Meldung 018 beschrieben.

020 Es konnten nur &NUM1 von den gewuenschten &NUM2 XAP-TP-Instanzen erzeugt werden.

In der Initialisierungsphase wird versucht, für jede gewünschte parallele Verbindung (Konfigurationsparameter ASSOCIATIONS) und für die sogenannte XAP-TP-Kontrollinstanz (zur Transaktionsüberwachung) eine XAP-TP-Instanz anzulegen. Lässt der XAP-TP-Provider aber statt der gewünschten *&NUM2* XAP-TP-Instanzen nur *&NUM1* zu, so wird diese Meldung ausgegeben.

Es liegt vermutlich ein Betriebsmittelengpass vor, den der Systemverwalter beheben sollte.

021 Neuer Log-Name von openUTM-LU62 erzeugt:
&LOGNAME

Wurde als Application Context der Wert UDTCCR oder UDTSEC generiert, so müssen auf LU6.2-Seite Conversations mit Sync-Level 2 unterstützt werden. Voraussetzung dafür ist der Austausch der Log-Namen mit dem LU6.2-Partner. Zur Information des Systemverwalters wird bei einem Kaltstart der Instanz mit dieser Meldung der neu erzeugte Log-Name *&LOGNAME* von openUTM-LU62 ausgegeben.

022 Log-Name von openUTM-LU62 aus Synclog-Datei restauriert:
&LOGNAME

Bei einem Warmstart von openUTM-LU62 wird kein neuer Log-Name generiert, sondern derjenige aus einem früheren Kaltstart übernommen. Diese Meldung erscheint nur während der Initialisierung und dient lediglich der Information des Systemverwalters.

023 Log-Name des Partners aus Synclog-Datei restauriert:
&LOGNAME

Bei einem Warmstart von openUTM-LU62 wird neben dem eigenen Log-Namen auch der Log-Name *&LOGNAME* des LU6.2-Partners der Synclog-Datei entnommen, sofern er bei einem vorangegangenen Lauf der Instanz durch einen erfolgreichen Austausch der Log-Namen bekannt geworden ist. Diese Meldung erscheint nur während der Initialisierung und dient lediglich der Information des Systemverwalters.

025 openUTM-LU62 beendet.

Diese Meldung wird als letzte in die Protokolldatei geschrieben, wenn eine Instanz von openUTM-LU62 normal durch Administration (`u62_adm -e`) beendet wurde.

026 Abbruch von openUTM-LU62 im Modul &MODULNAME,
Error Nummer = ERRNUM, Error Code = ERRCODE

Die openUTM-LU62-Instanz hat sich aufgrund eines schwerwiegenden internen Fehlers beendet. Die Angaben *&MODULNAME*, *&ERRNUM* und *&ERRCODE* bezeichnen interne Fehlerinformationen und sind für die Fehleranalyse von großer Bedeutung. Sichern Sie den Inhalt des Dateiverzeichnisses `/opt/lib/utmlu62/PROT` (UNIX-Systeme) bzw. `Programme\utmlu62\PROT` (Windows-Systeme). Wenn möglich, sollten Sie den Fehlerfall mit eingeschaltetem Instanz-Trace nachstellen.

027 Abbruch durch den XAP-TP-Provider:
&REASON

Der XAP-TP-Provider hat einen schwerwiegenden Fehler festgestellt und eine Beendigung der Instanz von openUTM-LU62 veranlasst. *&REASON* gibt genaueren Aufschluss über die Fehlerursache.

Die Spalte **Grp**. (Gruppe) in der folgenden Tabelle beschreibt, welcher Ursachengruppe der Abbruch angehört. Es gibt folgende Gruppen:

A Ursache ist ein Anwenderfehler, z.B. ein Fehler beim
- Generieren und administrieren von openUTM-LU62
- Generieren des System (z.B. Aufteilung des Adressraums)

S Ursache ist ein Fehler in einer anderen Systemkomponente (Software oder Hardware).

M Ursache ist ein Speicherengpass.

X Ursache ist ein interner Fehler von openUTM-LU62.

Mehrfachnennungen sind möglich, z.B. XAS.

Bei allen Fehlern der Gruppen S und X und bei allen **nicht** in der folgenden Tabelle aufgelisteten Fehlercodes sollten Sie zur Klärung den Siemens-Systemdienst heranziehen. Sichern Sie den Inhalt des Dateiverzeichnisses `/opt/lib/utmlu62/PROT` (UNIX-Systeme) bzw. `Programme\utmlu62\PROT` (Windows-Systeme). Wenn möglich, sollten Sie den Fehlerfall mit eingeschaltetem XAP-TP-Trace nachstellen.

REASON	Grp.	Ursache
AHQA00	XM	Modul KCOCOTA, Funktion QueueAnno(). Der Makro mGetBufferSpace() lieferte den Returncode LB_NOREM.
ASA001	XA	Der Modul KCOASAM wurde mit einem ungültigen Operationscode aufgerufen
ASA002	XA	Der Modul KCOASAM wurde mit dem Operationscode ASAM_SET_ACCPTS aufgerufen. Die mitgegebene Anzahl an access points ist größer als die Zahl der generierten.

REASON	Grp.	Ursache
ASA003	XA	Der Modul KCOASAM wurde mit dem Operationscode ASAM_SET_ACCPTS aufgerufen. Schlechter Returncode von BerSetAet.
ASA004	XA	Der Modul KCOASAM wurde mit dem Operationscode ASAM_SET_ PARTNER aufgerufen. Schlechter Returncode von BerSetAet.
ASA006	ASX	Der Modul KCOASAM wurde mit dem Operationscode ASAM_ATTACH aufgerufen. Schlechter Returncode von bBuildPAddr.
ASA007	ASX	Der Modul KCOASAM wurde mit dem Operationscode ASAM_ATTACH aufgerufen. OSS gibt Returncode NOTFIRST. Vermutlich ist die Instanz bereits gestartet.
ASA104	AX	Die Funktion „bBuildPAddr“ des Moduls KCOASAM wurde aufgerufen. Der Presentation Selector eines lokalen Accesspoints oder eines fernen Partners ist zu lang.
ASA105	AX	Die Funktion „bBuildPAddr“ des Moduls KCOASAM wurde aufgerufen. Der Session Selector eines lokalen Accesspoints oder eines fernen Partners ist zu lang.
ASA106	AX	Die Funktion „AssociationRequest“ des Moduls KCOASAM wurde aufgerufen. OSS liefert beim „assrq“ Aufruf den Returncode ERROR.
ASA107	AX	Die Funktion „AssociationRequest“ des Moduls KCOASAM wurde aufgerufen. OSS liefert beim „assrq“ Aufruf den Returncode INVREF.
ASA108	AX	Die Funktion „AssociationRequest“ des Moduls KCOASAM wurde aufgerufen. OSS liefert beim „assrq“ Aufruf einen schlechten Returncode.
ASA123	A	Der Modul KCOASAM wurde mit dem Operationscode ASAM_SET_TRANSSYNS aufgerufen. Die zu initialisierende Anzahl an Transfer Syntaxen ist größer als die Zahl der generierten.
ASA124	A	Der Modul KCOASAM wurde mit dem Operationscode ASAM_SET_TRANSSYNS aufgerufen. Die Anzahl der Elemente in einem Object Identifier, der eine Transfer Syntax bezeichnet, ist größer als erlaubt.
ASA125	A	Der Modul KCOASAM wurde mit dem Operationscode ASAM_ATTACH aufgerufen. Die Referenz für die Transfer Syntax enthält einen ungültigen Wert.
CDTN02	M	Modul KCOCOHF, Funktion CheckDtnidTtnid(). Der Makro mGetBufferSpace() lieferte den Returncode LB_NOREM.
CSND04	XM	Ungültiger Returncode nach Aufruf von PutElement() zum Anfordern eines dynamischen Puffers für Concatenator Sendedaten.
DMCA00	M	Modul KCOCODM, Funktion ConnectDynMemArea(). Die Funktion ConnectSharedMem() lieferte den Returncode MEM_NOMEM.

REASON	Grp.	Ursache
DMDI00	XA	Modul KCOCODM, Funktion DynDynMemDmpInfo(). Der Parameter <pnAreaEntries> zeigt auf einen Wert kleiner oder gleich Null.
EHHP00	M	Modul KCOXFEH, Funktion HandlePresEvent(). Der Returncode von mGetBufferSpace() war ungleich LB_OK.
EHRP01	MX	Modul KCOXFEH, Funktion ReloadPresEvent(). Der Makro mGetBufferSpace() lieferte einen Returncode ungleich LB_OK.
EHSP01	M	Modul KCOXFEH, Funktion StorePresEvent(). Die Funktion PutElement() lieferte den Returncode DM_NOMEM. Maßnahme: In der KDCDEF-Generierung den Wert für die Größe der OSI-Scratch-Area erhöhen (Parameter MAX OSI-SCRATCH-AREA).
EVGE00	M	Modul KCOXFEV, Funktion GetOssEvent(). Der Returncode des Makros mGetBufferSpace() war ungleich LB_OK.
FREE01 bis FREE03	XA	Modul KCOXFFO, Funktion ap_free(). Es sind mehr als APFREE_MAX_TO_REL Speicherbereiche freizugeben.
GSYS00	S	Modul KCOCOHF, Funktion GetSystemInfo(). Die UNIX C-Funktion uname() lieferte einen negativen Returncode.
LBBD00	AX	Modul KCOCOLB, Funktion BufferDumpInfo(). Der Parameter <nPartEntries> enthielt einen Wert kleiner oder gleich Null.
MACF02	M	Der Returncode von mGetBufferSpace() war ungleich LB_OK.
MACF03	M	Der Returncode von SetTimer() war ungleich TI_OK.
MACF04	M	Der Returncode von GetLogRecord war ungleich MACF_OK.
MFCR07 bis MFCR12 MFCR16 bis MFCR21 MFCR24	M	Der Returncode vom Makro mGetBufferSpace() war ungleich LB_OK.
MFDM03 bis MFDM06	M	Der Returncode von PutElement war ungleich DM_OK.
MFRM05 bis MFRM07	A	Beim TP_RECOVER_REQ ist kein freier Dialogtabelleneintrag für einen Transaktionszweig vorhanden. Mögliche Ursache: die Anzahl der Associations im vorherigen Anwendungslauf war größer als die Anzahl Associations im aktuellen Anwendungslauf.
MFRM08	M	Der Returncode vom Makro mGetBufferSpace() war ungleich LB_OK.

REASON	Grp.	Ursache
MFRM09 bis MFRM19 MFRM21	M	Der Returncode von PutElement war ungleich DM_OK.
MFRM25	A	Kein freier Tabelleneintrag für einen Log-Damage-Record vorhanden. Maßnahme: Log-Damage-Records mit TP_UPDATE_LOG_DAMAGE_REQ löschen oder nMaxLogDamRec auf einen größeren Wert setzen.
MFT102	M	Der Returncode vom Makro mGetBufferSpace() war ungleich LB_OK.
MFT104	M	Der Returncode von ChangeDescriptor war ungleich DM_OK.
MFT105	M	Der Returncode von PutElement war ungleich DM_OK.
MFT106	M	Der Returncode von CopyElement war ungleich DM_OK.
MFT107	M	Der Returncode von CopyElement war ungleich DM_OK.
MFT108	M	Der Returncode vom Makro mGetBufferSpace() war ungleich LB_OK.
MFT109	M	Der Returncode von CopyElement war ungleich DM_OK.
MFT110 MFT111	M	Der Returncode von PutElement war ungleich DM_OK.
MFT112	M	Ungültige Aktionsnummer (LOGREMOVE).
MFT113	M	Der Returncode von ChangeDescriptor war ungleich DM_OK.
MFT114	M	Der Returncode von CopyElement war ungleich DM_OK.
MFT115	M	Der Returncode von ChangeDescriptor war ungleich DM_OK.
MFT119	M	Der Returncode von GetLogRecord() war ungleich MACF_OK.
MFT120 bis MFT127	M	Der Returncode von PutElement war ungleich DM_OK.
MFT128 MFT129	M	Der Returncode von CopyElement war ungleich DM_OK.
MFT130 MFT131 MFT132	M	Der Returncode von PutElement war ungleich DM_OK.
MFT133	M	Der Returncode von CopyElement war ungleich DM_OK.
MFT134 MFT135	M	Der Returncode von PutElement war ungleich DM_OK.
MFT138 MFT139	M	Die Funktion PutElement hat einen Returncode ungleich DM_OK geliefert.
MFT141	M	Die Funktion PutElement hat einen Returncode ungleich DM_OK geliefert.

REASON	Grp.	Ursache
MFT142	M	Die Funktion CopyElement hat einen Returncode ungleich DM_OK geliefert.
MFT147 MFT151	M	Die Funktion PutElement hat einen Returncode ungleich DM_OK geliefert.
MFTP03	M	Der Returncode von PutElement war ungleich DM_OK.
MFTP04	M	Der Returncode vom Makro mGetBufferSpace() war ungleich LB_OK.
MFTP05 MFTP06	M	Der Returncode von PutElement war ungleich DM_OK.
MFTP07	M	Der Returncode von SetTimer war ungleich TI_OK.
MFTP10	M	Der Returncode vom Makro mGetBufferSpace() war ungleich LB_OK.
MFTP11	M	Der Returncode von RequestBuffer() war ungleich LB_OK.
MFTP12 bis MFTP19 MFTP24	M	Der Returncode vom Makro mGetBufferSpace() war ungleich LB_OK.
NMTE00	M	Modul KCOCOHF, Funktion NewMemTabEntry(). Die Funktion RequestBuffer() lieferte den Returncode LB_NOMEM.
NMTE02	M	Modul KCOCOHF, Funktion NewMemTabEntry(). Der Makro mGetBufferSpace() lieferte den Returncode LB_NOMEM.
PCTR00	M	Modul KCOCOHF, Funktion PrepareCtrlReq(). Der Makro mGetBufferSpace() lieferte den Returncode LB_NOMEM.
POLL03	XM	Modul KCOXFPL, Funktion ap_poll(). Der Returncode des Makros mGetBufferSpace() war ungleich LB_OK.
RCV009	XM	Modul KCOXFRV, Funktion ap_rcv(). Der Returncode der Funktion CopyElement() war ungleich DM_OK.
RCV012	XM	Modul KCOXFRV, Funktion ap_rcv(). Inkonsistenz in den booleschen Variablen <bSwitchToNextTtnid> und <bClearTtnid>.
RQOB00	M	Modul KCOCOHF, Funktion ReqOssInBuff(). Die Funktion RequestBuffer() lieferte den Returncode LB_NOMEM.
RVCA01	XA	Modul KCOXFRV, Funktion CopyVarLthAttr(). Die Funktion AllocUserMem() lieferte einen unerwarteten Returncode.
RVCS03	M	Modul KCOXFRV, Funktion CheckSaRetc(). Der von der Funktion SetAttribute() gelieferte Returncode war SA_NOMEM.
RVUO01	XS	Modul KCOXFRV, Funktion UserDataOut(). In der multitasking Variante reichen die von UTM zur Verfügung gestellten Buffer zur Aufnahme der User Daten nicht aus, um alle User Daten zu übergeben.

REASON	Grp.	Ursache
SACT14	XM	Ungültiger Returncode nach Aufruf von PutElement() zum Anfordern eines dynamischen Speicherbereichs für SACF Aktion COPY.
SDCS02	M	Modul KCOXFSD, Funktion CheckSaRetc(). Der SetAttribute() Returncode war SA_NOMEM.
SDUI01	M	Modul KCOXFSD, Funktion UserDataIn(). Der Makro mGetBufferSpace() lieferte den Returncode LB_NOMEM.
SND007	XM	Modul KCOXFSD, Funktion ap_snd(). Der Returncode der Funktion GetVarLthAttr() war ungleich GA_OK.
XADM12	AX	Modul KCOXFEX, Funktion apext_adm(). Die Funktion apext_adm() wurde in der multitasking Variante mit einem unbekannten Opcode aufgerufen.
XATT04	XM	Modul KCOXFEX, Funktion apext_att(). Der Returncode der Funktion EstablishBuffer() war ungleich LB_OK.
XATT12	XM	Modul KCOXFEX, Funktion apext_att(). Der Returncode der Funktion EstablishBuffer() war ungleich LB_OK.
XATT13	XM	Modul KCOXFEX, Funktion apext_att(). Der Returncode der Funktion RequestBuffer() war ungleich LB_OK.
XFAU03	XA	Modul KCOXFHF, Funktion AllocUserMem(). Der Parameter <Type> enthielt einen ungültigen Wert.
XFDU02	XA	Modul KCOXFHF, Funktion DeallocUserMem(). Der Parameter <Type> enthielt einen ungültigen Wert.
XFGA07	XSA	Modul KCOXFHF, Funktion GetAttribute(). Beim Lesen des Attributs AP_DTNID in der singletasking Variante lieferte die Funktion AllocUserMem() einen unerwarteten Returncode.
XFGA11	M	Modul KCOXFHF, Funktion GetAttribute(). Der Makro mGetBufferSpace() lieferte den Returncode LB_NOMEM.
XFGE02	AX	Modul KCOXFHF, Funktion bCheckAndGetCallEnv(). Die Funktion bCheckAndSetState() lieferte einen schlechten Returncode und der Anwendungsstatus war ungleich WAITING_DUMP_APPL.
XFSA07	MX	Modul KCOXFHF, Funktion SetAttribute(). Beim Setzen des Attributs AP_DTNID lieferte die Funktion PutElement() einen Returncode ungleich DM_OK.

028 Abbruch durch internen Watch Dog.

Das Programm `u62_tp` überwacht ständig, dass jeder Funktionsaufruf nicht länger als eine fest vorgegebene Zeitspanne dauert. Auf diese Weise werden Endlosschleifen und blockierende Systemfunktionen automatisch erkannt. Dieser Fehler kann gelegentlich beim Beenden von TRANSIT auftreten. Tritt er sonst auf, handelt es sich um einen Systemfehler.

030 Kaltstart von openUTM-LU62, da die Synclog-Datei &SYNCLOG fehlt.

openUTM-LU62 wurde ohne den Schalter -c aufgerufen, soll also einen Warmstart durchführen. Wenn jedoch openUTM-LU62 noch nie erfolgreich für die angegebene lokale LU gestartet wurde, fehlt die Datei *&SYNCLOG* mit den Informationen für einen Warmstart. In diesem Fall zeigt die Meldung 030 an, dass stattdessen ein Kaltstart durchgeführt wird. Diese Meldung dient lediglich der Information des Systemverwalters.

031 Fehler beim Oeffnen der Synclog-Datei &SYNCLOG,
errno &ERRNO (&ERRTEXT).
Da die CCR-Syntax fehlt, wird die Initialisierung jedoch fortgesetzt.

Diese Meldung wird während der Initialisierung ausgegeben, wenn openUTM-LU62 ohne den Schalter -c gestartet wurde, der Application Context UDTAC oder UDTDISAC generiert wurde, aber von einem vorangegangenen Start die Synclog-Datei *&SYNCLOG* noch vorhanden ist. Dies kann nur geschehen, wenn zwischen zwei Starts von openUTM-LU62 die Konfiguration geändert wurde, so dass der Application Context von UDTCCR oder UDTSEC auf UDTAC oder UDTDISAC geändert wurde. Bei Verwendung von UDTAC oder UDTDISAC gibt es keine Transaktionssicherung, deshalb wird von openUTM-LU62 auch keine Synclog-Datei angelegt. Die noch bestehende Datei *&SYNCLOG* wird jedoch vorsichtshalber nicht gelöscht. Wenn diese Datei tatsächlich nicht mehr benötigt wird, muss sie vom Systemverwalter manuell gelöscht werden.

032 Fehler beim Oeffnen der Synclog-Datei &SYNCLOG,
errno &ERRNO (&ERRTEXT).
Abbruch von openUTM-LU62, da CCR-Syntax konfiguriert ist.

Während des Warmstarts ist beim Öffnen der Synclog-Datei *&SYNCLOG* mit allen Informationen zum erfolgreichen Wiederanlauf ein Fehler aufgetreten, der durch den Inhalt der Variablen `errno` = *&ERRNO* genauer spezifiziert wird. Da in dieser Datei möglicherweise wichtige Informationen über abgebrochene Transaktionen enthalten sein können, bricht openUTM-LU62 in dieser Situation die Initialisierung ab. Ist sichergestellt, dass kein Transaktionswiederanlauf notwendig ist, so kann mit dem Schalter -c ein Kaltstart veranlasst werden.

033 Die Synclog-Datei &SYNCLOG bleibt unveraendert, da die CCR-Syntax nicht (mehr) konfiguriert ist.

Während des Warmstarts wird festgestellt, dass von einem vorangegangenen Start von openUTM-LU62 noch eine Synclog-Datei *&SYNCLOG* vorhanden ist. Da in der Zwischenzeit der Application Context von UDTCCR oder UDTSEC nach UDTAC oder UDTDISAC geändert wurde (Parameter APPL-CONTEXT), so dass keine Transaktionssicherung mehr notwendig ist, lässt openUTM-LU62 die Synclog-Datei vorsichtshalber unverändert liegen und setzt die Initialisierung fort.

034 Falsches Format der Synclog-Datei &SYNCLOG, Kaltstart empfohlen!

openUTM-LU62 stellt während des Warmstarts fest, dass die Version der Synclog-Datei nicht mit der Version des Programms `u62_tp` übereinstimmt (nur nach Installation einer neuen Version von openUTM-LU62 möglich) oder dass es sich bei *&SYNCLOG* um keine oder um eine beschädigte Synclog-Datei handelt. Dafür gibt es nur zwei plausible Erklärungen:

Entweder ist die Datei gar nicht von openUTM-LU62, sondern von einem System-User erzeugt worden oder die Datei konnte wegen Platzmangels im Dateisystem nicht vollständig auf Platte geschrieben werden und ist daher abgeschnitten. openUTM-LU62 bricht daraufhin die Initialisierung ab. Da mit der vorliegenden Synclog-Datei kein Warmstart möglich ist, sollte openUTM-LU62 anschließend kalt gestartet werden.

035 Die Eintraege in der Synclog-Datei &SYNCLOG sind nicht mit der aktuellen Konfiguration konsistent!
Kaltstart empfohlen!

Während des Warmstarts wird erkannt, dass bestimmte Einträge der Synclog-Datei nicht mit der aktuellen Konfiguration übereinstimmen, genauer handelt es sich dabei um APT und AEQ der lokalen und der fernen OSI-TP-Anwendung. Ein erfolgreicher Start von openUTM-LU62 ist dann nur mehr mit dem Schalter -c möglich.

036 Fehler beim Anlegen eines Shared Memory der Laenge &LEN,
errno &ERRNO (&ERRTEXT)

Lässt der Application Context Transaktionssicherung zu, so besorgt sich das Programm `u62_tp` während der Initialisierungsphase mit der Systemfunktion shmget() einen Shared Memory-Bereich, der ein Abbild der Synclog-Datei darstellt und auf den der Dämon-Prozess `u62_wlog` lesend zugreift, bevor er die Information auf Platte schreibt. Ein Fehler beim Anlegen dieses Shared Memory führt zum Abbruch von openUTM-LU62. Die Systemvariable `errno` gibt Auskunft über die genaue Ursache.

037 Fehler beim Attach an das Shared Memory mit der Id &ID,
errno &ERRNO (&ERRTEXT)

Nach dem Anlegen eines Shared Memory wird diesem mit der Systemfunktion shmat() eine Adresse zugewiesen, um Schreib- und Lesezugriffe zu ermöglichen. Geht dies schief, bricht openUTM-LU62 die Initialisierung ab.

040 Fehler in der Konfiguration der lokalen LU &LLUNAME
oder der Partner-LU &RLUNAME.
Die Kommunikation zum LU6.2- und zum OSI-TP-Partner wird erst nach Richtigstellung der Konfigurierung der LU6.2-Basis-Software gestartet.

Ist die in der Generierung von openUTM-LU62 enthaltene lokale LU *&LLUNAME* (Konfigurationsparameter LOC-LU-ALIAS) oder die ferne LU *&RLUNAME* (Konfigurationsparameter REM-LU-ALIAS) nicht in der LU6.2-Basis-Software konfiguriert, so wartet openUTM-LU62 mit dem Start der Kommunikation mit dem OSI-TP- und dem LU6.2-Partner. Im Falle von TRANSIT als LU6.2-Basis-Software kann die Ursache auch das Fehlen eines geeigneten PAIR-Parameters sein, das den Zusammenhang zwischen der lokalen und der fernen LU herstellt. Wird TRANSIT beendet, umkonfiguriert und wieder gestartet, so sollte alles wieder laufen.

041 Fehler in der Konfiguration der lokalen LU &LLUNAME und der
Partner-LU &RLUNAME ist nun korrigiert.

Die Kommunikation zum LU6.2- und zum OSI-TP-Partner wird jetzt gestartet.

openUTM-LU62 hat beim Start eine Inkonsistenz zur Generierung der LU6.2-Basis-Software (z. B. von TRANSIT) festgestellt und die Meldung 040 ausgegeben. Nachdem nun die Konfiguration der LU6.2-Basis-Software korrigiert worden ist, gibt openUTM-LU62 die Meldung 041 aus und startet die Kommunikation mit seinen beiden Partnern durch Absetzen eines RECEIVE_ALLOCATE-Aufrufs auf LU6.2-Seite und durch Öffnen der XAP-TP-Dialog-Instanzen auf OSI-TP-Seite.

042 Der Netzname der Partner-LU &ALIASNAME ist umkonfiguriert worden von
&OLDNAME nach &NEWNAME.
Kein Aufbau von Conversations moeglich!

openUTM-LU62 ermittelt bei einem Kaltstart den in der LU6.2-Basis-Software (z. B. in TRANSIT) konfigurierten Netzwerknamen der Partner-LU *&ALIASNAME* (in TRANSIT Konfigurationsparameter REM-LU-ALIAS) mit Hilfe eines sogenannten Control Operator Aufrufs. Bei jeder neu aufgebauten Conversation zum LU6.2-Partner wird dann überprüft, ob dieser Name noch immer korrekt ist oder in der Zwischenzeit durch Umkonfigurieren verändert wurde (*&NEWNAME*). Im Fehlerfall wird die Conversation sofort wieder abgebaut.

043 Der Netzname der Partner-LU &ALIASNAME ist wieder richtiggestellt.

openUTM-LU62 hatte festgestellt, dass der Netzwerkname der Partner-LU in der LU6.2-Basis-Software (z. B. in TRANSIT) umkonfiguriert worden war und deshalb die Meldung 042 ausgegeben. Wenn bei dem Versuch eines Conversation-Aufbaus erkannt wird, dass der Netzwerkname in der LU6.2-Basis-Software wieder richtiggestellt wurde, wird zur Information diese Meldung ausgegeben.

044 Der Netzname der lokalen LU &ALIASNAME ist umkonfiguriert worden von &OLDNAME nach &NEWNAME.
Kein Aufbau von Conversations moeglich!

Diese Meldung hat die gleiche Bedeutung wie Nummer 042 mit dem Unterschied, dass nicht die Partner-LU, sondern die lokale LU von der Änderung des Netzwerknamens betroffen ist.

045 Der Netzname der lokalen LU &ALIASNAME ist wieder richtiggestellt.

Diese Meldung hat die gleiche Bedeutung wie Nummer 044 mit der lokalen statt der fernen LU.

046 Aufbau neuer Conversations wieder moeglich.

Diese Meldung erscheint gemeinsam mit Meldung 043 und/oder 045 und zeigt an, dass nach einer Richtigstellung geänderter Netzwerknamen wieder Conversations zum LU6.2-Partner aufgebaut werden können.

047 Der Knoten ist aktiviert: Conversationaufbau moeglich.

Der Knoten des IBM Communications Server ist erfolgreich gestartet worden. Conversations zum LU6.2-Partner können jetzt (wieder) aufgebaut werden.

048 Der Knoten ist deaktiviert: Kein Conversationaufbau moeglich.

Der Knoten des IBM Communications Server ist vermutlich durch Eingriff des Administrators gestoppt worden. Conversations zum LU6.2-Partner können jetzt nicht mehr aufgebaut werden.

049 Der Attach Manager laeuft bereits:
Keine Zustellung ankommender Conversations.

Für die lokale LU der Instanz von openUTM-LU62 ist bereits ein Attach Manager gestartet - eventuell eine openUTM-LU62-Instanz in einer weiteren Umgebung (Variable U62_DIR!). In diesem Fall werden alle vom LU6.2-Partner beim IBM Communications Server ankommenden Anforderungen zum Conversationaufbau nicht an die openUTM-LU62-Instanz weitergegeben, sondern an eine andere Anwendung. D.h. es ist nur noch ein aktiver Verbindungsaufbau zum LU6.2-Partner möglich.

050 Read an Input-Pipe liefert Fehler ENOBUFS.

Beim Lesen mit read() aus einer Named-Pipe ist der Fehler `errno` = ENOBUFS aufgetreten, d.h. im Moment besteht ein Engpass an Systemspeicher. `u62_tp` wiederholt daraufhin nach einer Wartezeit von 1/4 Sekunde den Leseversuch; bei erneutem Auftreten des Fehlers ENOBUFS wird dann allerdings keine Fehlermeldung mehr ausgegeben. Tritt dieser Fehler mehr als 20 mal unmittelbar hintereinander oder mehr als 60 mal insgesamt während der Gesamtlaufzeit von `u62_tp` auf, so beendet sich das Programm abnormal.

051 Fehler beim Oeffnen der Pipe zum Administrationsprogramm mit PID = &ADMPID, errno &ERRNO (&ERRTEXT)

Alle Aufträge von den Administrationsprogrammen `u62_adm` und `u62_sta` an `u62_tp` werden von `u62_tp` durch eine Rückmeldung quittiert (auch `u62_adm -e`). Schlägt das Öffnen der Named-Pipe mit der Systemfunktion open() zum Senden der Rückmeldung fehl, so gibt `u62_tp` diese Meldung aus. Der Inhalt der Variablen `errno` = *&ERRNO* gibt die genauere Ursache dafür an.

Da die Administrationsprogramme nur maximal 5 Sekunden auf die Antwort von `u62_tp` warten und sich nach Ablauf dieser Wartezeit beenden, kann es bei zeitaufwendigen Aufträgen wie der Statusanzeige unter höherer Last vorkommen, dass das Administrationsprogramm seine Eingabe-Pipe wieder gelöscht hat, wenn `u62_tp` seine Quittung senden will.

052 Write in Output-Pipe zu Prozess mit PID = &PID liefert Fehler ENOBUFS.

Beim Schreiben einer Quittung an ein Administrationsprogramm oder eines Auftrags an den Write-Log-Dämon `u62_wlog` mit der Systemfunktion write() ist der Fehler `errno` = ENOBUFS aufgetreten. `u62_tp` wiederholt nach einer Wartezeit von einer 1/4 Sekunde den Schreibversuch, gibt aber bei einem erneuten Auftreten dieses Fehlers keine Meldung mehr aus.

Tritt dieser Fehler 10 mal unmittelbar hintereinander auf, so verwirft `u62_tp` einfach die interne Nachricht. Das Administrationsprogramm würde dann eine Zeitüberschreitung beim Warten auf die Quittung melden. Häuft sich der Fehler, so dass er inzwischen mehr als 60 mal insgesamt aufgetreten ist, so beendet sich `u62_tp` abnormal.

053 Die Output-Pipe zum Prozess mit PID = &PID ist voll.

Das System kann im Moment überhaupt keine Daten in die Named-Pipe zum Administrationsprogramm oder zum Write-Log-Dämon mit der PID = *&PID* schreiben. Da die internen Nachrichten von openUTM-LU62 relativ kurz sind und von den einzelnen Programmen auch sofort aus ihren Eingabe-Pipes ausgelesen werden, sollte diese Meldung nicht erscheinen. `u62_tp` wiederholt ähnlich wie beim Fehler ENOBUFS nach einer 1/4 Sekunde die Schreiboperation und verwirft nach 20 erfolglosen Versuchen die Nachricht.

054 Fehler beim Schreiben in die Pipe zum Prozess mit PID = &PID, errno &ERRNO (&ERRTEXT)

Beim Schreiben in die Output-Pipe zum Administrationsprogramm oder zum Write-Log-Dämon mit der PID = *&PID* ist der Fehler `errno` = *&ERRNO* aufgetreten. Da es sich hierbei um einen schwerwiegenden Fehler handeln muss, wiederholt `u62_tp` im Gegensatz zu den beiden obigen Fehlern (ENOBUFS, EAGAIN) den Schreibversuch nicht.

055 Quittung zum Administrationsprogramm verworfen!

Beim Öffnen einer Named-Pipe zu einem Administrationsprogramm oder beim Schreiben in diese Pipe ist ein Fehler aufgetreten, der noch nicht zum Abbruch von `u62_tp` geführt hat. D.h. dieser Meldung geht immer eine der Meldungen 051 - 054 voraus.

056 Unbekannte Meldung 0x&HEX ueber Pipe empfangen!

`u62_tp` hat eine interne Nachricht mit dem unbekannten Nachrichtentyp *&HEX* (4-stellig, hexadezimal) empfangen. Diese Meldungen deuten entweder auf einen Kommunikationsfehler zwischen `u62_tp` und den Administrationsprogrammen oder auf eine fehlerhafte Installation von openUTM-LU62 hin (inkompatible Versionen von `u62_tp` und `u62_adm` bzw. `u62_sta`).

057 Unbekannte Meldung 0x&HEX vom Administrationsprogramm empfangen!

Siehe Kommentar zur Meldung 056.

058 SNMP wird in dieser Version nicht unterstuetzt; Meldung 0x&HEX negativ quittiert!

Siehe Kommentar zur Meldung 056.

060 Fehler beim Oeffnen der Pipe zum Write-Log-Daemon mit PID = &PID, errno &ERRNO (&ERRTEXT)

Der Write-Log-Dämon `u62_wlog` wird von `u62_tp` während der Initialisierung gestartet, wenn UDTCCR oder UDTSEC als Application-Context konfiguriert ist, also Wiederanlauf-Information auf die Platte geschrieben werden muss. Sobald `u62_wlog` seine „Ready“-Nachricht an `u62_tp` gesendet hat, öffnet `u62_tp` seine Output-Pipe zu diesem Dämon für spätere Aufträge. Geht der Systemaufruf open() schief, so wird diese Meldung ausgegeben mit dem Grund des Fehlers in der Variablen `errno`.

061 Abbruch des Write-Log-Daemon mit PID &PID in der Startphase.

Der von `u62_tp` gestartete Write-Log-Dämon `u62_wlog` konnte seine Initialisierung nicht erfolgreich durchführen. In diesem Fall sendet er noch eine entsprechende interne Nachricht an `u62_tp`, die zur Ausgabe der Meldung 061 führt, und beendet sich daraufhin. `u62_tp` versucht dann nach einer gewissen Wartezeit den Start von `u62_wlog` erneut.

062 Write-Log-Daemon mit PID &PID meldet Fehler beim Schreiben in die Log-Datei => Abbruch!

Das Öffnen oder das Beschreiben der Synclog-Datei durch den Write-Log-Dämon ist misslungen. Dieser Fehler führt sofort zum Abbruch von `u62_tp`. Er sollte nur dann auftreten, wenn aufgrund eines vollen Dateisystems nicht mehr genügend Platz für die Synclog-Datei auf der Platte vorhanden ist. Da sich die Größe der Synclog-Datei während der Laufzeit von openUTM-LU62 jedoch nicht ändert, darf dies nur in der Startphase vorkommen.

063 Unbekannte Meldung 0x&HEX vom Write-Log-Daemon empfangen!

`u62_tp` hat eine interne Nachricht mit dem unbekannten Nachrichtentyp *&HEX* (4-stellig, hexadezimal) vom Write-Log-Dämon empfangen. Diese Meldung deutet entweder auf einen Kommunikationsfehler zwischen `u62_tp` und dem Write-Log-Daemon oder auf eine fehlerhafte Installation von openUTM-LU62 hin (inkompatible Versionen von `u62_tp` und `u62_wlog`).

064 Der Write-Log-Daemon mit PID &PID ist abgestuerzt!

Das Programm `u62_tp` überwacht in regelmäßigen Abständen, ob der von ihm gestartete Write-Log-Dämon mit der PID = *&PID* noch am Leben ist. Wird der Tod des Dämons festgestellt, so wird die Meldung 064 ausgegeben und `u62_wlog` erneut gestartet.

065 Write-Log-Daemon mit PID &PID gestartet.

Diese Meldung bedeutet keinen Fehler, sondern protokolliert nur die PID des soeben mittels fork() erzeugten Kind-Prozesses, der allerdings noch mittels exec() mit dem Programm `u62_wlog` überlagert werden muss.

066 Die maximale Anzahl &ANZ erfolgloser Versuche, den Write-Log-Daemon zu starten, ist ueberschritten.

`u62_tp` hat inzwischen mehrmals hintereinander - in immer größeren Zeitintervallen - erfolglos versucht, den Write-Log-Dämon `u62_wlog` zu starten. Ist die maximale Anzahl *&ANZ* solch erfolgloser Versuche erreicht, beendet sich `u62_tp`.

067 Fehler beim Starten des Write-Log-Daemons mittels fork(),
errno &ERRNO (&ERRTEXT)

Der Systemaufruf fork() zum Starten des Write-Log-Dämon in der Startphase oder nach dem von `u62_tp` festgestellten Verschwinden von `u62_wlog` ist schiefgegangen. Dies ist grundsätzlich auf einen Betriebsmittelmangel des Systems zurückzuführen. Der Inhalt der Variablen `errno` = *&ERRNO* gibt genaueren Aufschluss über die Ursache. `u62_tp` unternimmt nach einer gewissen Wartezeit einen erneuten Startversuch.

068 Fehler beim Starten des Write-Log-Daemons mittels execv(),
errno &ERRNO (&ERRTEXT)

Diese Meldung hat die gleiche Bedeutung wie die Meldung 067 mit dem Unterschied, dass nicht die Funktion fork(), sondern execv() zum Überlagern des Prozesses mit dem Programm `u62_wlog` schiefgegangen ist. Diese Fehlermeldung deutet entweder auf einen Betriebsmittelengpass auf dem System oder auf eine fehlerhafte Umgebung hin (`u62_wlog` nicht vorhanden oder nicht ablauffähig).

069 Meldung an den Write-Log-Daemon mit PID &PID konnte nicht abgesetzt werden! Beende und restarte den Daemon!

Beim Schreiben einer internen Nachricht in die Named-Pipe zum Write-Log-Dämon mit write() ist ein Fehler aufgetreten, der nicht zum Abbruch von `u62_tp` geführt hat. Dieser Meldung geht immer eine der Meldungen 052 - 054 voraus. `u62_tp` beendet daraufhin den Write-Log-Dämon, gibt alle zugehörigen Ressourcen frei und startet den Write-Log-Dämon erneut.

070 ap_set_env fuer Instanz &INSTNO fehlgeschlagen,
ap_errno = &APERRNO: &ERRTXT

Diese Meldung bedeutet einen internen Fehler bei der Verwendung der XAP-TP-Schnittstelle für die Kommunikation über OSI-TP. Dabei bedeuten

&INSTNO: Nummer der XAP-TP-Instanz
&APERRNO: XAP-TP-Fehlernummer
&ERRTXT: XAP-TP-Fehlertext

071 ap_get_env fuer Instanz &INSTNO fehlgeschlagen,
ap_errno = &APERRNO: &ERRTXT

Siehe Kommentar zur Meldung 070.

072 ap_free fuer Instanz &INSTNO fehlgeschlagen,
ap_errno = &APERRNO: &ERRTXT

Siehe Kommentar zur Meldung 070.

073 ap_bind fuer Instanz &INSTNO fehlgeschlagen,
ap_errno = &APERRNO: &ERRTXT

Siehe Kommentar zur Meldung 070.

074 ap_close fuer Instanz &INSTNO fehlgeschlagen,
ap_errno = &APERRNO: &ERRTXT

Siehe Kommentar zur Meldung 070.

075 ap_init_env fuer Instanz &INSTNO fehlgeschlagen,
ap_errno = &APERRNO: &ERRTXT

Siehe Kommentar zur Meldung 070.

076 ap_open fehlgeschlagen,
ap_errno = &APERRNO: &ERRTXT

Siehe Kommentar zur Meldung 070.

077 ap_rcv fuer Instanz &INSTNO fehlgeschlagen,
ap_errno = &APERRNO: &ERRTXT

Siehe Kommentar zur Meldung 070.

078 ap_snd fuer Instanz &INSTNO fehlgeschlagen,
ap_errno = &APERRNO: &ERRTXT

Siehe Kommentar zur Meldung 070.

079 ap_poll fehlgeschlagen,
ap_errno = &APERRNO: &ERRTXT

Siehe Kommentar zur Meldung 070.

080 apext_adm (operation code &OPCODE) fehlgeschlagen,
ap_errno = &APERRNO: &ERRTXT

Siehe Kommentar zur Meldung 070. &OPCODE bezeichnet die Kennung des Administrationsaufrufs

081 apext_att fehlgeschlagen,
ap_errno = &APERRNO: &ERRTXT

Siehe Kommentar zur Meldung 070.

082 apext_det fehlgeschlagen,
ap_errno = &APERRNO: &ERRTXT

Siehe Kommentar zur Meldung 070.

083 apext_init fuer Typ &INITTYPE fehlgeschlagen,
ap_errno = &APERRNO: &ERRTXT

Siehe Kommentar zur Meldung 070. &INITTYPE bezeichnet den Typ der zu initialisierenden Daten

090 Die Dump-Datei &FILENAME kann nicht geoeffnet werden.

`u62_tp` hat erfolglos versucht, aufgrund eines schwerwiegenden internen Fehlers, oder angestoßen durch ein Administrationskommando, einen Dump seiner Kontrollblöcke in die Datei *&FILENAME* auszugeben. Diese Datei konnte jedoch nicht geöffnet werden.

100 Meldung &MSGNUM des XAP-TP Providers:

Diese Meldung leitet eine Systemmeldung des XAP-TP-Providers ein. Der Folgetext ist abhängig von der Meldungsnummer *&MSGNUM*. Die einzelnen Meldungen des XAP-TP-Providers sind ab Seite 316 in einem eigenen Abschnitt aufgelistet.

Meldungen der Komponente DM

301 Conversation wird abgelehnt,
TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber:
CCR-Syntax fuer die OSI-TP-Partner-Anwendung nicht vereinbart

Der LU6.2-Partner hat eine Conversation mit Sync-Level 2 zum TP *&TPNAME* aufgebaut. (*&TPNAME* bezeichnet also den UTM-Transaktionscode.) Da der Application Context UDTAC oder UDTDISAC konfiguriert wurde, kann openUTM-LU62 nicht wie angefordert einen OSI-TP-Dialog mit Functional Unit Commit zum OSI-TP-Partner aufbauen und lehnt deshalb die Conversation sofort wieder ab.

302 Conversation wird abgelehnt,
TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber:
Restart der Kontrollinstanz ist noch nicht abgeschlossen.

Der LU6.2-Partner hat eine Conversation mit Sync-Level 2 zum TP *&TPNAME* aufgebaut. (*&TPNAME* bezeichnet also den UTM-Transaktionscode.) Da jedoch der Wiederanlauf von Transaktionen, die von einem vorangegangenen Start von openUTM-LU62 noch bestehen, noch nicht abgeschlossen ist, ist die XAP-TP-Kontrollinstanz noch nicht im geeigneten Zustand, um neue OSI-TP-Dialoge mit Functional Unit Commit zu unterstützen. Daher weist `u62_tp` die Conversation sofort ab. Dieser Fall kann auftreten, wenn nach einem Absturz die Verbindung zum LU6.2-Partner wieder aufgebaut wurde, die Verbindung zwischen openUTM-LU62 und der UTM-Anwendung aber noch nicht wieder hergestellt ist.

303 Ankommende Conversation
(TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber)
liefert Initialisierungsdaten, die jedoch von openUTM-LU62 verworfen werden.

Diese Meldung gibt den Hinweis, dass eine ankommende Conversation vom LU6.2-Partner Initialisierungsdaten mitgeliefert hat, die jedoch nicht zum OSI-TP-Partner (d.h. zur UTM-Anwendung) weitergereicht werden können. Überprüfen Sie das Anwendungsprogramm des LU6.2-Partners. Wenn die Initialisierungsdaten für die Anwendungslogik benötigt werden, ist keine sinnvolle Kommunikation möglich. *&TPNAME* bezeichnet den UTM-Transaktionscode.

304 Conversation wird abgelehnt,
TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber:
Security-Daten werden vom OSI-TP-Partner nicht unterstuetzt oder enthalten ungueltige Zeichen.

Eine vom LU6.2-Partner ankommende Conversation zum TP *&TPNAME* enthält Security-Daten (Benutzerkennung und eventuell Passwort) und kann nur dann zum OSI-TP-Partner (d.h. zur UTM-Anwendung) weitergereicht werden, wenn auch dieser eine Sicherheitsprüfung unterstützt. Dies ist jedoch nur dann der Fall, wenn APPL-CONTEXT = UDTSEC konfiguriert ist. Bei allen anderen Application Contexts oder bei einem Fehler bei der ASN1-Codierung der Security-Daten weist openUTM-LU62 die Conversation sofort mit dem Fehlercode AP_SECURITY_PARAMS_INVALID zurück. *&TPNAME* bezeichnet den UTM-Transaktionscode.

305 Conversation wird abgelehnt,
TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber:
Der Alias-Name &ALIAS der Partner-LU stimmt nicht mit dem konfigurierten Namen &CONFALIAS ueberein.

Es liegt ein Namensproblem in der Konfigurierung vor.

306 Conversation wird abgelehnt,
TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber:
Keine Unterstuetzung von Service-TPs auf mapped (!) Conversations!

Der LU6.2-Partner hat versucht, eine Conversation mit einem nicht abdruckbaren TP-Namen (d.h. Transaktionscode zwischen X'00' und X'3F') zu starten. openUTM-LU62 unterstützt außer dem Resync-TP X'06F2' keine derartigen sogenannte Service-TPs. Diese Meldung erscheint beispielsweise bei der Verwendung der Funktion EXEC CICS START, die in openUTM-LU62 das Service-TP X'02' anspricht.

307 Conversation wird abgelehnt,
TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber:
Konfigurationsfehler: Der Netzname der lokalen und/oder der fernen LU ist umkonfiguriert worden!

Eine vom LU6.2-Partner ankommende Conversation zum TP *&TPNAME* wird abgelehnt, weil bei der Überprüfung der Netzwerknamen der lokalen und der fernen LU eine Inkonsistenz festgestellt wurde. Dies bedeutet, dass sich hinter den Alias-Namen der lokalen und/oder der fernen LU (z. B. in der TRANSIT-Konfigurierung) nicht mehr dieselben LUs (im SNA-Netz) verbergen wie zu einem früheren Zeitpunkt. Dieser Meldung geht stets mindestens eine der Meldungen 042 und 044 voraus.

308 Conversation wird abgelehnt,
TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber:
Fehler beim Austausch der Log-Namen zwischen der lokalen und der fernen LU!

Der LU6.2-Partner hat zu openUTM-LU62 eine Conversation mit Sync-Level 2 aufgebaut, obwohl zu diesem Zeitpunkt die Log-Namen zwischen den beiden LUs noch nicht ausgetauscht waren oder es beim letzten Austausch der Log-Namen zu einem schwerwiegenden Fehler gekommen ist. Dieser Protokoll-Verstoß wird von openUTM-LU62 mit dem sofortigen Abbau der Conversation geahndet.

309 RECEIVE_ALLOCATE wegen Konfigurationsfehler zurueckgewiesen:
Der lokale LU-Alias-Name &LUNAME ist nicht konfiguriert!

Der RECEIVE_ALLOCATE-Aufruf zum asynchronen Annehmen von ankommenden Conversations ist von der LU6.2-Basis-Software (z. B. von TRANSIT) zurückgewiesen worden, weil der in diesem Aufruf enthaltene Alias-Name der lokalen LU nicht in der LU6.2-Basis-Software konfiguriert ist.

310 RECEIVE_ALLOCATE fehlgeschlagen:
LU6.2-Basis-Software laeuft nicht!

Der RECEIVE_ALLOCATE-Aufruf ist negativ beantwortet worden, weil die LU6.2-Basis-Software nicht läuft. Im Falle von TRANSIT als LU6.2-Basis-Software ist entweder das TRANSIT-Grundsystem oder der LU62Mgr nicht gestartet. openUTM-LU62 setzt nach einer gewissen Wartezeit erneut einen RECEIVE_ALLOCATE-Aufruf ab, gibt aber im Fehlerfall diese Meldung nicht wieder aus.

311 RECEIVE_ALLOCATE fehlgeschlagen:
Systemfehler (errno = &ERRNO) aufgetreten:
&ERRTXT

Bei der Verarbeitung des RECEIVE_ALLOCATE-Aufrufs durch die LU6.2-Basis-Software (z. B. durch TRANSIT) ist irgendein Systemaufruf fehlgeschlagen. Die eigentliche Fehlerursache in der Variablen `errno` sollte von der LU6.2-Basis-Software beim RECEIVE_ALLOCATE-Aufruf geliefert werden. Der Wert von `errno` *&ERRNO* samt einer kurzen Interpretation in *&ERRTXT* wird von openUTM-LU62 daher mit ausgegeben. Da jedoch TRANSIT als LU6.2-Basis-Software keinen Text liefert, erscheint als erklärender Fehlertext stets:

no interpretation of errno with TRANSIT as LU6.2 software

312 Aufbau einer Conversation zum LU6.2-Partner zurueckgewiesen,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Konfigurationsfehler:
Der lokale LU-Alias-Name &ALIASNAME ist nicht konfiguriert!

Die LU6.2-Basis-Software (z.B. TRANSIT) hat den Versuch, eine Conversation zum LU6.2-Partner aufzubauen, abgelehnt, weil der in openUTM-LU62 generierte Alias-Name *&ALIASNAME* der lokalen LU nicht auch in der LU6.2-Basis-Software konfiguriert ist.

313 Aufbau einer Conversation zum LU6.2-Partner fehlgeschlagen,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
LU6.2-Basis-Software laeuft nicht!

Die LU6.2-Basis-Software läuft nicht. Im Falle von TRANSIT ist also das TRANSIT-Grundsystem oder der LU62Mgr nicht gestartet. Damit ist keine Kommunikation über LU6.2 möglich. *&TPNAME* bezeichnet z. B. den CICS-Transaktionscode.

314 Aufbau einer Conversation zum LU6.2-Partner fehlgeschlagen,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Systemfehler (errno = &ERRNO) aufgetreten:
&ERRTXT

Bei dem Versuch, eine Conversation zum LU6.2-Partner aufzubauen, ist in der LU6.2-Basis-Software (z. B. in TRANSIT) irgendein Systemaufruf schiefgegangen. Ähnlich wie beim RECEIVE_ALLOCATE-Aufruf (siehe Meldung 311!) stellt TRANSIT keinen Fehlertext zur Verfügung, so dass als *&ERRTXT* immer folgender Text ausgegeben wird:

no interpretation of errno with TRANSIT as LU6.2 software

315 Bereits aufgebaute Conversation zum LU6.2-Partner
(TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber) wegen Konfigurationsfehler wieder abgebaut:
Der Netzname der lokalen und/oder der fernen LU ist umkonfiguriert worden!

Eine bereits aktiv aufgebaute Conversation zum TP *&TPNAME* des LU6.2-Partners wird unmittelbar danach wieder abgebaut, weil bei der Überprüfung der Netzwerknamen der lokalen und der fernen LU eine Inkonsistenz festgestellt wurde. Dies bedeutet, dass sich hinter den Alias-Namen der lokalen und/oder der fernen LU (z. B. in der TRANSIT-Konfigurierung) nicht mehr dieselben LUs (im SNA-Netz) verbergen wie zu einem früheren Zeitpunkt. Dieser Meldung geht stets mindestens eine der Meldungen 042 und 044 voraus.

316 Ankommende Conversation fuer TP &TPNAME nicht mehr vorhanden:
Vermutlich bei einer Kollision von RECEIVE_ALLOCATE mit TP_ENDED versehentlich abgebaut!

Ein temporärer Fehler in der LU6.2-Basis-Software führte zum Abbruch einer vom LU6.2-Partner ankommenden Conversation. *&TPNAME* bezeichnet den UTM-Transaktionscode.

320 Aufbau einer Conversation zum LU6.2-Partner fehlgeschlagen,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Allokationsfehler (Code = &ERRTXT)

Im Moment lässt sich keine Conversation zum TP *&TPNAME* des LU6.2-Partners aktiv aufbauen. (*&TPNAME* ist in diesem Fall z. B. der CICS-Transaktionscode.) Der Fehlercode *&ERRTXT* enthält dann im Klartext den Returncode des entsprechenden LU6.2-Aufrufs.

321 Aufbau einer Conversation zum LU6.2-Partner fehlgeschlagen,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Inkonsistenz in den Konfigurationen von openUTM-LU62 und
der LU6.2-Basis-Software
(MODENAME, LOC_LU_ALIAS, REM_LU_ALIAS)

Mit den aktuellen Generierungen der LU6.2-Basis-Software (z. B. TRANSIT) und openUTM-LU62 lassen sich grundsätzlich keine Conversations zum LU6.2-Partner aufbauen, weil der Alias-Name der lokalen oder der fernen LU oder der Name des Modes nicht in der Konfiguration der LU6.2-Basis-Software enthalten ist. Diese Situation bleibt solange bestehen, bis die LU6.2-Basis-Software oder die openUTM-LU62-Instanz gestoppt, umgeneriert und wieder gestartet ist.

322 Aufbau einer Conversation zum LU6.2-Partner fehlgeschlagen,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Security-Daten ungueltig

Die LU6.2-Basis-Software (z. B. TRANSIT) hat die Anforderung zum Aufbau einer Conversation zum LU6.2-Partner abgelehnt, weil die Security-Daten Benutzerkennung und/oder Passwort irgendwelche Sonderzeichen enthalten oder weil der Partner beim Aufbau der Session angezeigt hat, dass er keine Security-Daten unterstützt. *&TPNAME* ist in diesem Fall z. B. der CICS-Transaktionscode.

323 Aufbau einer Conversation zum LU6.2-Partner fehlgeschlagen,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Der Sync-Level &SYNCLEVEL wird von der
LU6.2-Basis-Software nicht unterstuetzt.

Die LU6.2-Basis-Software (z. B. TRANSIT) hat die Anforderung zum Aufbau einer Conversation zum LU6.2-Partner abgelehnt, weil der angeforderte Sync-Level *&SYNCLEVEL* SYNCPT (2) oder CONFIRM (1) nicht möglich ist. Wenn der LU6.2-Partner den angegebenen Sync-Level unterstützt, genügt es bei TRANSIT, für die lokale LU den Generierungsparameter PAIR_EXT in der TRANSIT-Konfigurierung entsprechend anzupassen. *&TPNAME* ist in diesem Fall z. B. der CICS-Transaktionscode.

330 Conversation zum LU6.2-Partner
(TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber)
vom Partner mit ABEND abgebrochen!

Der LU6.2-Partner hat die Conversation abnormal mit Deallocate(Abend) abgebrochen. openUTM-LU62 beendet daraufhin den OSI-TP-Dialog zum OSI-TP-Partner (d.h. zur UTM-Anwendung) ebenfalls abnormal mit U_ABORT_REQ. Der eigentliche zugrundeliegende Fehler ist in der Regel im Anwendungsprogramm des LU6.2-Partners zu suchen.

331 Conversation zum LU6.2-Partner
(TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber)
vom Partner mit ABEND abgebrochen!

Siehe Kommentar zur Meldung 330.

332 Conversation zum LU6.2-Partner
(TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber)
durch Ausfall der LU6.2-Basis-Software abgebrochen!

Die LU6.2-Basis-Software hat sich beendet. Bei Verwendung von TRANSIT heißt dies, dass das TRANSIT-Grundsystem oder der LU62Mgr beendet worden sind. Dies wird von openUTM-LU62 beim nächsten Aufruf der LU6.2-Basis-Software bemerkt und führt zum abnormalen Abbruch des OSI-TP-Dialogs mit dem OSI-TP-Partner (d.h. mit der UTM-Anwendung) mittels U_ABORT_REQ.

333 Conversation zum LU6.2-Partner
(TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber)
durch Ausfall der LU6.2-Basis-Software abgebrochen!

Siehe Kommentar zur Meldung 332.

334 Conversation zum LU6.2-Partner
(TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber)
wegen Conversation-Fehler abgebrochen!

Auf der Session, die der Conversation zum LU6.2-Partner zugrundeliegt, ist ein schwerwiegender Fehler aufgetreten, z.B. durch Ausfall der Link-Verbindung zur Partner-PU oder durch administrativen Abbau aller Sessions mit dem Kommando `u62_adm -as`. openUTM-LU62 bricht daraufhin den OSI-TP-Dialog zum OSI-TP-Partner (d.h. zur UTM-Anwendung) abnormal ab.

335 Conversation zum LU6.2-Partner
(TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber)
wegen Conversation-Fehler abgebrochen!

Siehe Kommentar zur Meldung 334.

336 Auf Conversation zum LU6.2-Partner
(TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber)
User Control Daten empfangen =>
Abbruch der Conversation durch openUTM-LU62!

Der LU6.2-Partner hat an openUTM-LU62 sogenannte User Control Daten gesendet. Da diese nicht ins OSI-TP-Protokoll umgesetzt werden können, bricht openUTM-LU62 die Conversation zum LU6.2-Partner und den OSI-TP-Dialog mit dem OSI-TP-Partner (d.h. mit der UTM-Anwendung) ab.

337 Auf Conversation zum LU6.2-Partner
(TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber)
User Control Daten empfangen =>
Abbruch der Conversation durch openUTM-LU62!

Siehe Kommentar zur Meldung 336.

338 Auf Conversation zum LU6.2-Partner
(TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber)
wurde AP_STATE_CHECK fuer ein RECEIVE-Verb gemeldet =>
Abbruch der Conversation durch openUTM-LU62!

Die LU6.2-Basis-Software hat beim Empfang eines Datenpakets vom LU6.2-Partner einen formalen Codierungsfehler der Daten festgestellt. Daraufhin hat sie die Daten verworfen, selbtständig die Senderichtung gewechselt und eine negative Quittung an den LU6.2-Partner gesendet. openUTM-LU62 erhält in dieser Situation den Fehlercode AP_STATE_CHECK und baut daraufhin die Conversation und den zugehörigen OSI-TP-Dialog ab.

339 Auf Conversation zum LU6.2-Partner
(TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber)
wurde AP_STATE_CHECK fuer ein RECEIVE-Verb gemeldet =>
Abbruch der Conversation durch openUTM-LU62!

Siehe Kommentar zur Meldung 338.

340 Ankommende Conversation zum LU6.2-Partner abgelehnt,
TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber:
Ablauf des Timers zur Ueberwachung des Associationsaufbaus

Der LU6.2-Partner hat eine Conversation zum TP *&TPNAME* aufgebaut. (*&TPNAME* bezeichnet also den UTM-Transaktionscode.) openUTM-LU62 kann jedoch im Moment aufgrund eines Mangels an freien parallelen Verbindungen keinen OSI-TP-Dialog zum OSI-TP-Partner (d.h. zur UTM-Anwendung) aufbauen. Wird innerhalb einer bestimmten Wartezeit keine parallele Verbindung frei, so weist openUTM-LU62 die Conversation zurück. Diese maximale Wartezeit ist in openUTM-LU62 mit dem Parameter ALLOC-TIME konfigurierbar, der Standardwert beträgt 30 Sekunden. Erscheint diese Meldung häufiger, so sollte entweder das Sessionlimit (in TRANSIT der Generierungsparameter SESS-MAX bei XMODE) erniedrigt oder die Anzahl der parallelen Verbindungen von openUTM-LU62 (Generierungsparameter ASSOCIATIONS) erhöht werden.

341 Ankommender Dialog zum OSI-TP-Partner abgelehnt,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Ablauf des Timers zur Ueberwachung des Conversationaufbaus

Der OSI-TP-Partner (d.h. die UTM-Anwendung) hat einen OSI-TP-Dialog zum TP *&TPNAME* aufgebaut. (*&TPNAME* bezeichnet also z. B. den CICS-Transaktionscode.) openUTM-LU62 kann jedoch im Moment aufgrund eines Mangels an freien Sessions keine Conversation zum LU6.2-Partner aufbauen. Wird innerhalb einer bestimmten Wartezeit keine Session frei, so weist openUTM-LU62 den OSI-TP-Dialog zurück. Diese maximale Wartezeit ist in openUTM-LU62 mit dem Parameter ALLOC-TIME konfigurierbar, der Standardwert beträgt 30 Sekunden. Erscheint diese Meldung häufiger, so sollte entweder das Sessionlimit (in TRANSIT der Generierungsparameter SESS-MAX bei XMODE) erhöht oder die Anzahl der parallelen Verbindungen von openUTM-LU62 (Generierungsparameter ASSOCIATIONS) erniedrigt werden.

350 XAP-TP-Provider lehnt Assoziationswunsch ab
(TP &TPNAME): result = &RES, source = &SRC, reason = &RSN

openUTM-LU62 hat eine ankommende Conversation vom LU6.2-Partner angenommen und versucht, eine parallele Verbindung für einen OSI-TP-Dialog zum TP *&TPNAME* des OSI-TP-Partners (d.h. zum Transaktionscode *&TPNAME* der UTM-Anwendung) zu belegen. Der lokale XAP-TP-Provider hat jedoch den Request abgelehnt. openUTM-LU62 bricht daraufhin die Conversation zum LU6.2-Partner ab.

Bedeutung von *&RES*:

1	Association-Allocation-Request dauerhaft abgelehnt
2	Association-Allocation-Request vorübergehend abgelehnt

Bedeutung von *&SRC*:

0	ACSE Service User
1	ACSE Service Provider
2	Presentation Service Provider
7	APM Service Provider (Association Pool Manager)

Bedeutung von *&RSN*:

-1	kein Grund angegeben
0	CCR-Version 2 nicht verfügbar
1	Versions-Inkompatibilität
2	Contention Winner Eigenschaft abgelehnt
3	Bid mandatory Wert abgelehnt
6	lokaler oder ferner Application Entity Title (AET) ungültig
7	kein Association-Pool gefunden für den angegebenen AET
58	Alle Associations aus dem Association-Pool sind belegt, jedoch gestatten die Pool-Grenzen den Aufbau weiterer Associations. Der XAP-TP-Provider hat mit dem Aufbau weiterer Associations begonnen.
59	Alle Associations aus dem Association-Pool sind belegt und die Pool-Grenzen gestatten keinen Aufbau weiterer Associations.
60	Der Timer des Association-Pool-Managers ist abgelaufen. Dieser Timer bestimmt, wie lange maximal auf die Freigabe einer Association gewartet wird, wenn alle Associations aus dem Pool belegt sind.
61	Die Association soll für eine protected Conversation belegt werden und die aktuelle Transaktion befindet sich in der Endphase. Zu diesem Zeitpunkt können keine weiteren OSI-TP-Dialoge mehr an den Transaktionsbaum angegliedert werden.

351 XAP-TP-Provider meldet Verlust der Assoziation (TP &TPNAME): source = &SRC, reason = &RSN, event = &EVT

openUTM-LU62 hat eine ankommende Conversation vom LU6.2-Partner angenommen und bereits eine parallele Verbindung für einen OSI-TP-Dialog zum TP *&TPNAME* des OSI-TP-Partners (d.h. zum Transaktionscode *&TPNAME* der UTM-Anwendung) erfolgreich belegt. Bevor es jedoch zum OSI-TP-Dialogaufbau kommt, geht die parallele Verbindung wieder verloren. openUTM-LU62 bricht daraufhin die Conversation zum LU6.2-Partner ab.

Bedeutung von *&SRC*:

0	ACSE Service User
1	ACSE Service Provider
2	Presentation Service Provider
7	APM Service Provider (Association Pool Manager)

Bedeutung von *&RSN* bei *&SRC* gleich 0 oder 1:

-1	A-RELEASE-IND: kein Grund angegeben
0	A-RELEASE-IND: normaler Release-Request
1	A-RELEASE-IND: dringender Release-Request
12	A-ABORT-IND
30	A-RELEASE-IND: benutzerdefinierter Release-Request

Bedeutung von *&RSN* bei *&SRC* gleich 2:

-1	kein Abbruchgrund angegeben
0	Abbruchgrund nicht bekannt
1	unbekannte Presentation Protocol Data Unit empfangen
2	unerwartete Presentation Protocol Data Unit empfangen
3	unerwartete Session Service Primitive empfangen
4	Presentation Protocol Data Unit mit unbekanntem Parameter empfangen
5	Presentation Protocol Data Unit mit unerwartetem Parameter empfangen
6	Presentation Protocol Data Unit mit ungültigem Parameter empfangen

Bedeutung von *&RSN* bei *&SRC* gleich 7:

5	Ein OSI-TP-Dialog vom Partner hat die Association belegt

352 OSI-TP-Partner (XAP-TP-User) hat Dialogwunsch abgelehnt:
TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber

openUTM-LU62 hat aufgrund einer ankommenden Conversation vom LU6.2-Partner versucht, einen OSI-TP-Dialog zum TP *&TPNAME* des OSI-TP-Partners (d.h. zum Transaktionscode *&TPNAME* der UTM-Anwendung) aufzubauen. Der ferne XAP-TP-User (d.h. das UTM-Anwendungsprogramm) hat jedoch den OSI-TP-Dialog abgelehnt, so dass bei openUTM-LU62 eine TP_BEGIN_DIALOGUE-confirmation mit result = rejected eingetroffen ist. openUTM-LU62 beendet daraufhin die Conversation zum LU6.2-Partner.

353 OSI-TP-Partner (XAP-TP-Provider) hat Dialogwunsch abgelehnt,
TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber:
Reason = &RSN

openUTM-LU62 hat aufgrund einer ankommenden Conversation vom LU6.2-Partner versucht, einen OSI-TP-Dialog zum TP *&TPNAME* des OSI-TP-Partners (d.h. zum Transaktionscode *&TPNAME* der UTM-Anwendung) aufzubauen. Der ferne XAP-TP-Provider (d.h. von openUTM selbst) hat jedoch den OSI-TP-Dialog abgelehnt, so dass bei openUTM-LU62 eine TP_BEGIN_DIALOGUE-confirmation mit result = rejected eingetroffen ist. openUTM-LU62 beendet daraufhin die Conversation zum LU6.2-Partner. Der Ablehnungsgrund ist in der SYSLOG-Datei der UTM-Anwendung zu finden.

Bedeutung von *&RSN*:

-1	kein Grund angegeben
1	Der angegebene TPSU Title (Transaktionscode) ist beim Partner nicht bekannt
2	Der angegebene TPSU Title (Transaktionscode) ist beim Partner dauerhaft nicht verfügbar
3	Der angegebene TPSU Title (Transaktionscode) ist beim Partner vorübergehend nicht verfügbar
4	Die Partner-Anwendung hat ein TP-BEGIN-DIALOGUE-indication ohne Angabe eines TPSU Title empfangen
5	Eine oder mehrere der gewünschten Funktionseinheiten (Functional Units) werden vom Partner nicht unterstützt
6	Die gewünschte Kombination von Funktionseinheiten (Functional Units) wird vom Partner nicht unterstützt
7	Die Association, die für den OSI-TP-Dialog belegt wurde, wird bereits von der Partner-Anwendung benutzt
8	Die Application Entity Parameter, die im TP-BEGIN-DIALOGUE-request angegeben wurde, identifizieren keine bekannte Application Entity Invocation

354 Diagnose-Information in den Initialisierungsdaten von AP_TP_BEGIN_DIALOGUE_CNF: 0xhhhh (d,d)

wobei hhhh Hexadezimal-Werte und d,d die entsprechenden Dezimalwerte darstellen.

Bei UTM als OSI-TP-Partner enthalten die Initialisierungsdaten genauere Informationen über den Grund der Ablehnung des Dialogs. Der erste Wert zeigt an, ob es sich um einen transienten (1) oder permanenten (2) Fehler handelt.

Der zweite Wert gibt den genauen Ablehnungsgrund an.

Die Bedeutung dieser Werte ist dem Handbuch openUTM V5.3 Meldungen, Test und Diagnose in BS2000/OSD zu entnehmen (Beschreibung DIA3 in Meldung K119 bei DIA1=2)

355 Protokollverstoss: AP_TP_ACCEPT in AP_TP_BEGIN_DIALOGUE_CNF empfangen, TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber

Der OSI-TP-Partner hat den ankommenden OSI-TP-Dialog positiv quittiert, obwohl openUTM-LU62 keine explizite Quittung beim OSI-TP-Dialogaufbau angefordert hat. Dieser Protokollverstoß führt dann zum Abbruch des OSI-TP-Dialogs und der Conversation zum LU6.2-Partner. Diese Meldung kann nur auftreten bei einem OSI-TP-Partner, der keine UTM-Anwendung ist.

356 Protokollverstoss: AP_TP_BEGIN_TRANSACTION_IND erhalten, TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber

openUTM-LU62 hat vom OSI-TP-Partner ein BEGIN_TRANSACTION-Primitiv erhalten. Dies ist ein schwerer Verstoß gegen das OSI-TP-Protokoll. Folglich bricht openUTM-LU62 den OSI-TP-Dialog und die Conversation zum LU6.2-Partner ab. Diese Meldung kann nur auftreten bei einem OSI-TP-Partner, der keine UTM-Anwendung ist.

357 Protokollverstoss:
AP_TP_BEGIN_TRANSACTION_IND erhalten,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber

Siehe Kommentar zur Meldung 356.

358 Die Laenge (&LEN) der vom OSI-TP-Partner empfangenen User-Daten ueberschreitet den Maximalwert von &MAXLEN:
TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber

Der OSI-TP-Partner (d.h. die UTM-Anwendung) hat Netto-Daten der Länge *&LEN* zu openUTM-LU62 gesendet. Zum LU6.2-Partner können allerdings nur *&MAXLEN* Bytes - bei TRANSIT als LU6.2-Basis-Software sind das 32763 Bytes - in einem Datenpaket weitergereicht werden. openUTM-LU62 beendet OSI-TP-Dialog wie Conversation daraufhin abnormal. Das UTM-Anwendungsprogramm muss also geändert werden.

359 Die Laenge (&LEN) der vom OSI-TP-Partner empfangenen User-Daten ueberschreitet den Maximalwert von &MAXLEN:
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber

Siehe Kommentar zur Meldung 358.

360 Ankommender Dialogwunsch abgelehnt, OSI-TP-Seite ist Auftraggeber:
Kein lokaler TPSU Title angegeben

openUTM-LU62 hat vom OSI-TP-Partner eine TP_BEGIN_DIALOGUE-indication ohne Angabe eines TPSU Title, d.h. ohne Angabe des Transaktionscodes empfangen. Der OSI-TP-Dialogwunsch wird von openUTM-LU62 abgelehnt. Diese Meldung kann nur auftreten bei einem OSI-TP-Partner, der keine UTM-Anwendung ist.

361 Ankommender Dialogwunsch abgelehnt, OSI-TP-Seite ist Auftraggeber:
Lokaler TPSU Title kann nicht dekodiert werden

Beim Dekodieren des in der TP_BEGIN_DIALOGUE-indication enthaltenen TPSU Title (d.h. des Transaktionscodes) ist ein Fehler aufgetreten; vermutlich hat der OSI-TP-Partner (d.h. die UTM-Anwendung) den TPSU Title fehlerhaft codiert. openUTM-LU62 lehnt daher den OSI-TP-Dialogwunsch ab. Diese Meldung kann nur auftreten bei einem OSI-TP-Partner, der keine UTM-Anwendung ist.

362 Ankommender Dialogwunsch abgelehnt, OSI-TP-Seite ist Auftraggeber:
Typ des lokalen TPSU Title fehlerhaft

Der Typ des in der TP_BEGIN_DIALOGUE-indication enthaltenen TPSU Title (d.h. des Transaktionscodes) ist weder PRINTABLE_STRING noch T61_STRING. Da openUTM-LU62 nur diese beiden Typen unterstützt, wird der OSI-TP-Dialogwunsch abgelehnt.

363 Ankommender Dialogwunsch abgelehnt, OSI-TP-Seite ist Auftraggeber:
Die Laenge des lokalen TPSU Title (&LEN) ueberschreitet den Maximalwert von &MAXLEN

Der in der TP_BEGIN_DIALOGUE-indication enthaltene dekodierte TPSU Title (d.h. z. B. der CICS-Transaktionscode) ist *&LEN* Zeichen lang. Beim Conversationaufbau zum LU6.2-Partner hin stehen jedoch nur *&MAXLEN* Zeichen (bei TRANSIT als LU6.2-Basis-Software 64 Zeichen) zur Verfügung. openUTM-LU62 lehnt daher den OSI-TP-Dialogwunsch ab.

364 Ankommender Dialogwunsch abgelehnt,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Shared Control gewuenscht

openUTM-LU62 hat vom OSI-TP-Partner eine TP_BEGIN_DIALOGUE-indication mit der Funktionseinheit (Functional Unit) Shared Control empfangen. Da diese Funktionalität von openUTM-LU62 nicht unterstützt wird, wird der OSI-TP-Dialog abgebrochen. Diese Meldung kann nur auftreten bei einem OSI-TP-Partner, der keine UTM-Anwendung ist.

365 Ankommender Dialogwunsch abgelehnt,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Codierung des fernen Application Process Title fehlerhaft

openUTM-LU62 hat vom OSI-TP-Partner eine TP_BEGIN_DIALOGUE-indication mit einem fehlerhaft codierten APT (application Process Title) bzw AEQ (Application Entity Qualifier) erhalten. Diese Meldung kann nur auftreten bei einem OSI-TP-Partner, der keine UTM-Anwendung ist.

366 Ankommender Dialogwunsch abgelehnt,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Typ des fernen Application Entity Qualifier fehlerhaft

Siehe Kommentar zur Meldung 365.

369 Ankommender Dialogwunsch abgelehnt,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
CCR-Syntax fuer die Partner-Anwendung nicht vereinbart

openUTM-LU62 hat vom OSI-TP-Partner (d.h. von der UTM-Anwendung) eine TP_BEGIN_DIALOGUE-indication mit der Functional Unit Commit empfangen, obwohl ein Application Context UDTAC oder UDTDISAC konfiguriert wurde und damit keine Transaktionssicherung möglich ist. openUTM-LU62 lehnt den OSI-TP-Dialogwunsch daher ab.

Wenn Die Funktionseinheit Commit von openUTM-LU62 unterstützt werden soll, muss ein geeigneter mit dem OSI-TP-Partner abgestimmter Application Context generiert werden:

APPL-CONTEXT = UDTSEC (Standard) oder
APPL-CONTEXT = UDTCCR

370 Ankommender Dialogwunsch abgelehnt,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Explizite Bestaetigung mittels DIALOGUE_RSP angefordert

Der OSI-TP-Partner hat beim OSI-TP-Dialogaufbau eine explizite Bestätigung mit einer TP_BEGIN_DIALOGUE_RSP-Primitiven angefordert. Da das LU6.2-Protokoll keine direkte Entsprechung dafür hat, wird der OSI-TP-Dialogwunsch von openUTM-LU62 abgewiesen. Diese Meldung kann nur auftreten bei einem OSI-TP-Partner, der keine UTM-Anwendung ist.

371 Ankommender Dialogwunsch abgelehnt,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Keine Unterstuetzung von unchained transactions

openUTM-LU62 hat vom OSI-TP-Partner eine TP_BEGIN_DIALOGUE-indication mit der Functional Unit Commit_and_unchained empfangen. Das LU6.2-Protokoll bietet jedoch keine entsprechende Funktionalität, so dass openUTM-LU62 den OSI-TP-Dialogwunsch ablehnt. Diese Meldung kann nur auftreten bei einem OSI-TP-Partner, der keine UTM-Anwendung ist.

373 Ankommender Dialogwunsch abgelehnt,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Restart der Kontrollinstanz ist noch nicht abgeschlossen

Der OSI-TP-Partner (d.h. die UTM-Anwendung) hat zu openUTM-LU62 einen OSI-TP-Dialog mit Functional Unit Commit aufgebaut zu einem Zeitpunkt, als nach dem Start von openUTM-LU62 der Wiederanlauf der noch bestehenden Transaktionen noch nicht abgeschlossen war. In dieser Situation ist die XAP-TP-Kontrollinstanz noch nicht in der Lage, die Transaktionssicherung auf neuen OSI-TP-Dialogen mit Functional Unit Commit zu unterstützen. openUTM-LU62 weist deshalb den OSI-TP-Dialogwunsch ab. *&TPNAME* bezeichnet den TP-Namen auf LU6.2-Seite, also z. B. den CICS-Transaktionscode.

374 Ankommender Dialogwunsch abgelehnt,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Der ferne Application Process Title stimmt nicht mit der Konfigurierung ueberein

375 Ankommender Dialogwunsch abgelehnt,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Der ferne Application Entity Qualifier stimmt nicht mit der Konfigurierung ueberein

openUTM-LU62 hat vom OSI-TP-Partner (d.h. von der UTM-Anwendung) eine TP_BEGIN_DIALOGUE-indication mit einem APT (application Process Title) bzw. AEQ (Application Entity Qualifier) empfangen, der nicht mit der Generierung von openUTM-LU62 übereinstimmt. Die entsprechenden Parameter in der openUTM-LU62-Generierung, die gegebenenfalls zu modifizieren sind, sind REM-APT und REM-AEQ.

376 Ankommender Dialogwunsch abgelehnt,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Codierung des Application Context Name fehlerhaft

openUTM-LU62 hat vom OSI-TP-Partner eine TP_BEGIN_DIALOGUE-indication mit einem fehlerhaft codierten Application Context Name, was zur Ablehnung des OSI-TP-Dialogwunsches führt. Diese Meldung kann nur auftreten bei einem OSI-TP-Partner, der keine UTM-Anwendung ist.

377 Warnung: Application Context Name nicht konfiguriert.
Dialogwunsch fuer TP &TPNAME wird trotzdem akzeptiert!

openUTM-LU62 hat vom OSI-TP-Partner (d.h. von der UTM-Anwendung) eine TP_BEGIN_DIALOGUE-indication mit einem Application Context Name empfangen, der nicht mit der Generierung von openUTM-LU62 übereinstimmt. Dieser Fehler wird als nicht besonders schwerwiegend angesehen, so dass es trotzdem zum Aufbau einer Conversation zum LU6.2-Partner kommt. Allerdings sollte der openUTM-LU62-Generierungsparameter APPL-CONTEXT entsprechend angepasst werden!

378 Ankommender Dialogwunsch abgelehnt,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Presentation Context Identifier nicht ermittelbar

Der Presentation Context Identifier ist eine Größe, die das Paar (abstrakte Syntax, Transfersyntax) innerhalb des Application Contexts eindeutig beschreibt und für die Codierung und Decodierung von Nutz- und Security-Daten benötigt wird. Diese Größe wird nach dem Empfang einer TP_BEGIN_DIALOGUE-indication ermittelt. Ist dies nicht möglich, so liegt ein schwerwiegender Fehler vor, der zum Abbruch des OSI-TP-Dialogs führt.

379 Ankommender Dialogwunsch abgelehnt,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Keine Unterstuetzung von Initialisierungsdaten

openUTM-LU62 hat vom OSI-TP-Partner eine TP_BEGIN_DIALOGUE-indication mit angehängten Initialisierungsdaten empfangen. Solche Daten sind nur dann erlaubt, wenn der Application Context UDTSEC ist. In diesem Fall sind dies gerade die Security-Daten (Benutzerkennung und Passwort bzw. der „Already verified“-Indikator). Für alle anderen Application Contexts ist nicht klar, wie die Initialisierungsdaten umzukodieren sind, um sie als sogenannte PIP-Daten im ALLOCATE dem LU6.2-Partner zu übermitteln. Daher lehnt openUTM-LU62 hier den OSI-TP-Dialogwunsch ab. Diese Meldung kann nur auftreten bei einem OSI-TP-Partner, der keine UTM-Anwendung ist.

380 Ankommender Dialogwunsch abgelehnt,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Fehlerhafte Codierung der Security-Daten

openUTM-LU62 hat vom OSI-TP-Partner eine TP_BEGIN_DIALOGUE-indication mit Initialisierungsdaten empfangen. Da der Application Context UDTSEC ist, werden diese Daten als Security-Daten interpretiert und gemäß ASN.1 dekodiert. Geht diese Dekodierung schief, so liegt ein schwerwiegender Fehler vor, der zur Ablehnung des OSI-TP-Dialogwunsches führt. Diese Meldung kann nur auftreten bei einem OSI-TP-Partner, der keine UTM-Anwendung ist.

381 Ankommender Dialogwunsch abgelehnt,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
User-Id oder Passwort zu lang:
Laenge der User-Id = &ULEN, Laenge des Passwortes = &PLEN

openUTM-LU62 hat vom OSI-TP-Partner (d.h. von der UTM-Anwendung) eine TP_BEGIN_DIALOGUE-indication mit Initialisierungsdaten empfangen und diese Daten als Security-Daten interpretiert (Application Context = UDTSEC). Bei der Dekodierung dieser Daten hat sich herausgestellt, dass die Länge der angegebenen Benutzerkennung *&ULEN* oder des angegebenen Passworts *&PLEN* den Maximalwert von 10 überschreitet, der an der LU6.2-Schnittstelle vorgegeben ist. Da also die Security-Daten dem LU6.2-Partner nicht in voller Länge übermittelt werden können, lehnt openUTM-LU62 den OSI-TP-Dialogwunsch ab.

382 Ankommender Dialogwunsch abgelehnt,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Die leere User-Id ist unzulaessig

383 Ankommender Dialogwunsch abgelehnt,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
User-Id als Octet String ist unzulaessig

384 Ankommender Dialogwunsch abgelehnt,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Ein leeres Passwort ist unzulaessig

385 Ankommender Dialogwunsch abgelehnt,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Passwort als Octet String ist unzulaessig

Die voranstehenden vier Meldungen zeigen an, dass die vom OSI-TP-Partner (d.h. von der UTM-Anwendung) empfangene TP_BEGIN_DIALOGUE-indication eine unzulässige Benutzerkennung oder ein unzulässiges Passwort enthält. Leere Benutzerkennungen und Passworte werden jedoch von openUTM-LU62 ebenso wenig unterstützt wie Benutzerkennungen bzw. Passworte, die als Octet String codiert sind, da dies nicht ins LU6.2-Protokoll umgesetzt werden kann. Zulässig sind nur Codierungen als PRINTABLE_STRING oder als T61_STRING sowie die Angabe von „Already verified“ statt eines Passwortes.

386 Ankommender Dialogwunsch abgelehnt,
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber:
Fehler beim Austausch der Log-Namen zwischen der lokalen
und der fernen LU!

Der OSI-TP-Partner (d.h. die UTM-Anwendung) hat einen OSI-TP-Dialog mit Functional Unit Commit zu openUTM-LU62 aufgebaut. Bevor jedoch openUTM-LU62 eine entsprechende Conversation mit Sync-Level 2 zum LU6.2-Partner aufbauen darf, müssen im Fall, dass die Anzahl der Sessions inzwischen auf 0 gesunken war, die Log-Namen zwischen den beiden LUs erneut ausgetauscht werden - weil ja dann der LU6.2-Partner in der Zwischenzeit einen Kaltstart durchgeführt haben kann. Kommt es dann bei diesem Austausch der Log-Namen zu einem Fehler, so wird der OSI-TP-Dialogwunsch des OSI-TP-Partners (d.h. der UTM-Anwendung) abgewiesen. Dabei ist es unerheblich, ob der aufgetretene Fehler nur temporärer Natur war (z.B. Sessionverlust des Resync-TP) oder ob die Log-Namen tatsächlich inkonsistent sind. Genaueren Aufschluss über die Ursache gibt jedoch eine Meldung der Komponente RS (Resync-TP), die dieser Meldung vorausgeht.

390 OSI-TP-Dialog durch Partner (XAP-TP-Provider) abgebrochen:
TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber

391 OSI-TP-Dialog durch Partner (XAP-TP-Provider) abgebrochen:
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber

392 OSI-TP-Dialog durch Partner (XAP-TP-User) abgebrochen:
TP-Name &TPNAME, LU6.2-Seite ist Auftraggeber

393 OSI-TP-Dialog durch Partner (XAP-TP-User) abgebrochen:
TP-Name &TPNAME, OSI-TP-Seite ist Auftraggeber

Der OSI-TP-Dialog zwischen openUTM-LU62 und dem OSI-TP-Partner (d.h. der UTM-Anwendung) ist vom Partner abnormal abgebrochen worden. Ist der Verursacher der ferne XAP-TP-Provider (d.h. openUTM selbst), so ist eine TP_P_ABORT-indication empfangen worden (Meldungen 390 und 391). Ist der Verursacher der ferne XAP-TP-User (also das Anwendungsprogramm), so hat openUTM-LU62 eine TP_U_ABORT-indication erhalten (Meldungen 392 und 393). In allen Fällen bricht openUTM-LU62 die zugehörige Conversation zum LU6.2-Partner daraufhin mit Deallocate(Abend) oder Deallocate(Resource-Failure) ab.

394 Empfangene Log-Daten:

Diese Meldung wird zusätzlich zu Meldung 392 bzw. 393 ausgegeben, wenn die TP_U_ABORT-indication Log-Daten enthält. Da diese Daten jedoch nicht an den LU6.2-Partner weitergereicht werden können, werden sie in hexadezimaler Schreibweise in der Datei `prot.`*luname* protokolliert und danach verworfen.

399 Statistik:
Aktiv aufgebaute ungesicherte Conversations = &NA1
Aktiv aufgebaute gesicherte Conversations = &NA2
Passiv aufgebaute ungesicherte Conversations = &NP1
Passiv aufgebaute gesicherte Conversations = &NP2

Gesamtzahl aller bisherigen Conversations = &N3

Diese Meldung wird jede Stunde und bei der Beendigung von `u62_tp` ausgegeben. Sie gibt Auskunft darüber, wieviele Transaktionen gelaufen sind.

Meldungen der Komponente TM

400 Startphase erfolgreich abgeschlossen.

Das Programm `u62_tp` hat die Startphase erfolgreich abgeschlossen. Erst nachdem diese Meldung ausgegeben wurde, können Transaktionen gestartet werden.

401 Warmstart.
&ANZAHL Transaktionen müssen resynchronisiert werden.

Beim vorhergehenden Beenden von openUTM-LU62 waren noch *&ANZAHL* Transaktionen in undefiniertem Zustand. Bevor neue Transaktionen gestartet werden können, müssen zunächst die offenen Transaktionen resynchronisiert werden. Die offenen Transaktionen werden genauer in der nachfolgenden Meldung 402 beschrieben.

402 Information über offene Transaktion:
TP-Name &TPNAME, &PARTNER-Seite ist Auftraggeber.
LUWID (LU6.2):
LU name = &LUNAME
instance = &INSTANCE
sequence = &SEQUENCE
AAID (OSI-TP):
size = &SIZE
aaid = &AAID
Log-Records: &RECORDS
LU6.2-Resync-Rolle: &ROLE

Diese Meldung beschreibt die beim Start der openUTM-LU62-Instanz in der Synclog-Datei gefundenen Log-Records.
&TPNAME bezeichnet den Transaktionscode des Auftragnehmers, der vom Auftraggeber aufgerufen wurde.
&PARTNER kann den Wert LU6.2 oder OSI-TP haben. Bei LU6.2 wurde vom LU6.2-Partner (z. B. von einem CICS-Programm) der UTM-Transaktionscode *&TPNAME* aufgerufen. Bei OSI-TP wurde von einem UTM-Anwendungsprogramm der Transaktionscode *&TPNAME* beim LU6.2-Partner (also z. B. der CICS-Transaktionscode *&TPNAME*) aufgerufen.
Die Transaktion kann eindeutig durch ihre LUWID und ihre AAID identifziert werden. Der LU6.2-Partner kennt nur die LUWID der Transaktion, sie wird im Fehlerfall z.B. im CICS-Logfile ausgegeben. Die UTM-Anwendung kennt nur die AAID der Transaktion, sie wird im Fehlerfall im SYSLOG von UTM ausgegeben.
&RECORDS beschreibt die Zustandsinformation der Transaktion. Die einzelnen Werte sind unten beschrieben.
&ROLE beschreibt die Rolle der openUTM-LU62-Instanz beim Transaktionswiederanlauf zum LU6.2-Partner. Die einzelnen Werte sind unten beschrieben.

RECORDS	**Beschreibung**
agent,committed	Der LU6.2-Partner hat die Transaktion gestartet, openUTM-LU62 hat die Transaktion bereits festgeschrieben, der LU6.2-Partner ist hierüber möglicherweise noch nicht informiert.
agent,in-doubt	Der LU6.2-Partner hat die Transaktion gestartet, openUTM-LU62 kann die Transaktion im Moment noch vor- oder zurücksetzen.
agent,HM	Der LU6.2-Partner hat die Transaktion gestartet, der OSI-TP-Partner (d.h. die UTM-Anwendung) hat einen heuristischen Mix gemeldet, d.h. die beteiligten Datenbanken sind nicht konsistent. Dies muss dem LU6.2-Partner noch mitgeteilt werden.
agent,RIP	Der LU6.2-Partner hat die Transaktion gestartet, der OSI-TP-Partner (d.h. die UTM-Anwendung) hat einen heuristischen Hazard gemeldet, d.h. es ist nicht sicher, ob die beteiligten Datenbanken konsistent sind. Dies muss dem LU6.2-Partner noch mitgeteilt werden.

RECORDS	Beschreibung
init,spm-pend	Der OSI-TP-Partner (d.h. die UTM-Anwendung) hat die Transaktion gestartet. Es ist noch keine Information vom LU6.2-Partner eingetroffen, ob er die Transaktion vor- oder zurücksetzen will.
log-commit	openUTM-LU62 hat die Transaktion bereits festgeschrieben, der OSI-TP-Partner (d.h. die UTM-Anwendung) ist hierüber möglicherweise noch nicht informiert.
log-ready	openUTM-LU62 kann die Transaktion im Moment noch vor- oder zurücksetzen.
log-damage-mix	Der LU6.2-Partner hat einen heuristischen Mix gemeldet, d.h. die beteiligten Datenbanken sind nicht konsistent. Dies muss dem OSI-TP-Partner (d.h. der UTM-Anwendung) noch mitgeteilt werden.
log-damage-haz	Der LU6.2-Partner hat „Resync in progress“ gemeldet, d.h. es ist nicht sicher, ob die beteiligten Datenbanken konsistent sind. Dies muss dem OSI-TP-Partner (d.h. der UTM-Anwendung) noch mitgeteilt werden.

ROLE	Beschreibung
ACTIVE	openUTM-LU62 ist verantwortlich für den Transaktionswiederanlauf zum LU6.2-Partner.
PASSIVE	Der LU6.2-Partner ist verantwortlich für den Transaktionswiederanlauf, d.h. er muss ein „Compare States“ senden.
NONE	Aus Sicht von openUTM-LU62 ist kein Transaktionswiederanlauf zum LU6.2-Partner erforderlich.

403 Verlust der Verbindung zum LU6.2-Partner.
Transaktion muss resynchronisiert werden.
TP-Name &TPNAME, &PARTNER-Seite ist Auftraggeber.
LUWID (LU6.2):
LU name = &LUNAME
instance = &INSTANCE
sequence = &SEQUENCE
AAID (OSI-TP):
size = &SIZE
aaid = &AAID

Während der Commit-Phase einer Transaktion ging die Verbindung zum LU6.2-Partner verloren. Nach Wiederherstellung der Verbindung muss die Transaktion resynchronisiert werden.
Zur Bedeutung von *&TPNAME*, *&PARTNER*, LUWID und AAID siehe Meldung 402.

404 Naechster Wiederanlaufversuch in &ZEIT Sekunden.

Diese Meldung wird ausgegeben, wenn während der Resynchronisierung einer Transaktion ein Fehler, z.B. ein Verbindungsabbruch auftritt und die Resynchronisierung später nochmals wiederholt werden muss.

405 Transaktion wurde erfolgreich resynchronisiert.
TP-Name &TPNAME, &PARTNER-Seite ist Auftraggeber.
LUWID (LU6.2):
LU name = &LUNAME
instance = &INSTANCE
sequence = &SEQUENCE
AAID (OSI-TP):
size = &SIZE
aaid = &AAID

Eine durch Verbindungsverlust oder sonstige Fehler abgebrochene Transaktion wurde erfolgreich resynchronisiert.
Zur Bedeutung von *&TPNAME*, *&PARTNER*, LUWID und AAID siehe Meldung 402.

406 LU6.2-Partner meldet heuristischen Mix.
TP-Name &TPNAME, &PARTNER-Seite ist Auftraggeber.
LUWID (LU6.2):
LU name = &LUNAME
instance = &INSTANCE
sequence = &SEQUENCE
AAID (OSI-TP):
size = &SIZE
aaid = &AAID

Der LU6.2-Partner hat die Transaktion z.B. nach einem Verbindungsverlust eigenständig beendet ohne auf die UTM-Anwendung zu warten. Die Transaktion ist damit inkonsistent. Datenbankänderungen auf UTM-Seite und auf LU6.2-Seite, die durch die Transaktion koordiniert werden sollten, wurden fehlerhaft ausgeführt. Beispielsweise wurde die geforderte Datenbankänderung auf LU6.2-Seite ausgeführt, auf UTM-Seite aber nicht.
Zur Bedeutung von *&TPNAME*, *&PARTNER*, LUWID und AAID siehe Meldung 402.

Üblicherweise sind in diesem Fall manuelle Eingriffe in eine der beteiligten Datenbanken notwendig, um die inkonsistenten Datenbankstände wieder zu koordinieren.

407 LU6.2-Partner meldet resync in progress.
TP-Name &TPNAME, &PARTNER-Seite ist Auftraggeber.
LUWID (LU6.2):
LU name = &LUNAME
instance = &INSTANCE
sequence = &SEQUENCE
AAID (OSI-TP):
size = &SIZE
aaid = &AAID

Der LU6.2-Partner hat die Transaktion z.B. nach einem Verbindungsverlust eigenständig beendet ohne auf die UTM-Anwendung zu warten. Es ist nicht bekannt, ob die Transaktion inkonsistent ist oder nicht.
Zur Bedeutung von *&TPNAME*, *&PARTNER*, LUWID und AAID siehe Meldung 402.

Die Verwalter der an der Transaktion beteiligten Datenbanken sollten prüfen, ob eine Datenbank-Inkonsistenz entstanden ist.

408 OSI-TP-Partner meldet heuristischen Mix.
TP-Name &TPNAME, &PARTNER-Seite ist Auftraggeber.
LUWID (LU6.2):
LU name = &LUNAME
instance = &INSTANCE
sequence = &SEQUENCE
AAID (OSI-TP):
size = &SIZE
aaid = &AAID

Der OSI-TP-Partner (d.h. UTM-Anwendung oder die Datenbank auf UTM-Seite) hat die Transaktion z.B. nach einem Verbindungsverlust eigenständig beendet ohne auf den LU6.2-Partner zu warten. Die Transaktion ist damit inkonsistent. Datenbankänderungen auf UTM-Seite und auf LU6.2-Seite, die durch die Transaktion koordiniert werden sollten, wurden fehlerhaft ausgeführt. Beispielsweise wurde die geforderte Datenbankänderung auf LU6.2-Seite ausgeführt, auf UTM-Seite aber nicht.
Zur Bedeutung von *&TPNAME*, *&PARTNER*, LUWID und AAID siehe Meldung 402.

Üblicherweise sind in diesem Fall manuelle Eingriffe in eine der beteiligten Datenbanken notwendig, um die inkonsistenten Datenbankstände wieder zu koordinieren.

409 OSI-TP-Partner meldet heuristischen Hazard.
TP-Name &TPNAME, &PARTNER-Seite ist Auftraggeber.
LUWID (LU6.2):
LU name = &LUNAME
instance = &INSTANCE
sequence = &SEQUENCE
AAID (OSI-TP):
size = &SIZE
aaid = &AAID

Nach einem Verbindungsverlust zum OSI-TP-Partner (d.h. zur UTM-Anwendung) kann nicht mehr mit Sicherheit festgestellt werden, ob die offene Transaktion durchgeführt oder zurückgesetzt wurde.
Zur Bedeutung von *&TPNAME*, *&PARTNER*, LUWID und AAID siehe Meldung 402.

Die Verwalter der an der Transaktion beteiligten Datenbanken sollten prüfen, ob eine Datenbank-Inkonsistenz entstanden ist.

410 Syncpoint-Request vom LU6.2-Auftragnehmer mit TP-Name &TPNAME erhalten. Transaktion wird abgebrochen.

Ein Auftragnehmerprogramm auf LU6.2-Seite hat einen Syncpoint angefordert (z.B. EXEC CICS SYNCPOINT) ohne vom UTM-Anwendungsprogramm dazu aufgefordert worden zu sein. Dies ist bei openUTM verboten. Die Transaktion wird somit abgebrochen. *&TPNAME* gibt den Transaktionscode auf LU6.2-Seite an.

Das Anwendungsprogramm auf LU6.2-Seite, z.B. das CICS-Anwendungsprogramm muss geändert werden.

420 Fehler beim Resynchronisieren einer Transaktion.
LU6.2-Partner akzeptiert den Zustand &STATE nicht.
TP-Name &TPNAME, &PARTNER-Seite ist Auftraggeber.
LUWID (LU6.2):
LU name = &LUNAME
instance = &INSTANCE
sequence = &SEQUENCE
AAID (OSI-TP):
size = &SIZE
aaid = &AAID
Evtl. muss openUTM-LU62 kalt gestartet werden.

Eine Transaktion wurde wegen eines Verbindungsverlustes auf der LU6.2-Seite in der Commit-Phase unterbrochen. Beim Resynchronisieren mit dem LU6.2-Partner trat ein Fehler auf. Der LU6.2-Partner akzeptiert den von openUTM-LU62 vorgeschlagenen Zustand der Transaktion nicht.
Zur Bedeutung von *&TPNAME*, *&PARTNER*, LUWID und AAID siehe Meldung 402.

Die Transaktion muss in diesem Fall durch manuelle Eingriffe beendet werden. Auf Seite von openUTM-LU62 besteht der einzig mögliche manuelle Eingriff in einer Beendigung der betroffenen openUTM-LU62-Instanz und anschließendem Kaltstart. Wenn es sich beim LU6.2-Partner um CICS handelt, muss dort das Kommando „CEMT SET CONNECTION NOTPENDING“ abgesetzt werden. Es ist jedoch zu beachten, dass durch diese manuellen Eingriffe eventuell Datenbank-Inkonsistenzen entstehen können. Dies sollte von den Verwaltern der beteiligten Datenbanken geprüft werden.

421 Log-Name Fehler beim Wiederaufsetzen einer Transaktion.
TP-Name &TPNAME, &PARTNER-Seite ist Auftraggeber.
LUWID (LU6.2):
LU name = &LUNAME
instance = &INSTANCE
sequence = &SEQUENCE
AAID (OSI-TP):
size = &SIZE
aaid = &AAID
Evtl. muss openUTM-LU62 kalt gestartet werden.

Eine Transaktion wurde wegen eines Verbindungsverlustes auf der LU6.2-Seite in der Commit-Phase unterbrochen. Beim Resynchronisieren mit dem LU6.2-Partner trat ein Fehler auf. Der LU6.2-Partner akzeptiert den von openUTM-LU62 vorgeschlagenen Log-Name nicht. Ursache ist vermutlich ein Kaltstart von openUTM-LU62 oder des LU6.2-Partners oder ein „CEMT SET CONNECTION NOTPENDIG“ bei CICS.
Zur Bedeutung von *&TPNAME*, *&PARTNER*, LUWID und AAID siehe Meldung 402.

Wenn die Ursache in einem Kaltstart des LU6.2-Partners liegt, sollte geprüft werden, ob ein Zurückgehen auf den vorherigen Anwendungslauf möglich ist, d.h. ob sich der Kaltstart rücksetzen lässt. Wenn dies nicht möglich ist, muss die openUTM-LU62-Instanz beendet und kalt gestartet werden.
Wenn die Ursache in einem Kaltstart von openUTM-LU62 liegt, muss auch der LU6.2-Partner einen Kaltstart durchführen. Bei CICS erreicht man dies z.B. durch das Kommando „CEMT SET CONNECTION NOTPENDING“.
Es ist jedoch zu beachten, dass durch diese manuellen Eingriffe eventuell Datenbank-Inkonsistenzen entstehen können. Dies sollte von den Verwaltern der beteiligten Datenbanken geprüft werden.

422 Protokollfehler des LU6.2-Partners:
&EVENT im Zustand &STATE1/&STATE2 erhalten.
TP-Name &TPNAME, &PARTNER-Seite ist Auftraggeber.
LUWID (LU6.2):
LU name = &LUNAME
instance = &INSTANCE
sequence = &SEQUENCE
AAID (OSI-TP):
size = &SIZE
aaid = &AAID
Transaktion wird abgebrochen.

Der LU6.2-Partner hat gegen das LU6.2-Protokoll verstoßen.
Zur Bedeutung von *&TPNAME*, *&PARTNER*, LUWID und AAID siehe Meldung 402.

Wenn der Protokollfehler reproduzierbar ist, das Ganze mit Trace wiederholen und den Systemdienst verständigen.

423 Protokollfehler des LU6.2-Partners:
Compare States &LUWSTATE &RRI im Zustand &STATE1/&STATE2 erhalten.
TP-Name &TPNAME, &PARTNER-Seite ist Auftraggeber.
LUWID (LU6.2):
LU name = &LUNAME
instance = &INSTANCE
sequence = &SEQUENCE
AAID (OSI-TP):
size = &SIZE
aaid = &AAID

Der LU6.2-Partner hat während dem Resynchronisieren einer Transaktion gegen das LU6.2-Protokoll verstoßen.
Zur Bedeutung von *&TPNAME*, *&PARTNER*, LUWID und AAID siehe Meldung 402.

Wenn der Protokollfehler reproduzierbar ist, das Ganze mit Trace wiederholen und den Systemdienst verständigen. Möglicherweise müssen die openUTM-LU62-Instanz und der LU6.2-Partner kalt gestartet werden um die Situation zu bereinigen. Genaueres hierzu siehe Meldung 421.

424 LU6.2-Partner meldet Protokollfehler.
Zustand: &STAT1/&STATE2,
TP-Name &TPNAME, &PARTNER-Seite ist Auftraggeber.
LUWID (LU6.2):
LU name = &LUNAME
instance = &INSTANCE
sequence = &SEQUENCE
AAID (OSI-TP):
size = &SIZE
aaid = &AAID

Der LU6.2-Partner ist der Meinung, dass openUTM-LU62 gegen das LU6.2-Protokoll verstoßen hat. Wenn der Protokollverstoß im normalen Betrieb gemeldet wird, führt dies zu einem Rücksetzen der Transaktion.
Zur Bedeutung von *&TPNAME*, *&PARTNER*, LUWID und AAID siehe Meldung 402.

Wenn der Protokollfehler reproduzierbar ist, das Ganze mit Trace wiederholen und den Systemdienst verständigen. Es ist möglich, dass Datenbank-Inkonsistenzen entstanden sind. Möglicherweise müssen auch die openUTM-LU62-Instanz und der LU6.2-Partner kalt gestartet werden um die Situation zu bereinigen. Genaueres hierzu siehe Meldung 421.

425 Fehlerhafter Log-Record (Fehlertyp &ERROR).
Transaktion wird geloescht.
LUWID (LU6.2):
LU name = &LUNAME
instance = &INSTANCE
sequence = &SEQUENCE
AAID (OSI-TP):
size = &SIZE
aaid = &AAID

Beim Warmstart von openUTM-LU62 wurde ein fehlerhafter Log-Record eingelesen. Die in diesem Log-Record beschriebene Transaktion kann deshalb nicht resynchronisiert werden.
Zur Bedeutung von LUWID und AAID siehe Meldung 402.

Wenn vom vorhergehenden Lauf von openUTM-LU62 noch irgendwelche Diagnoseunterlagen vorhanden sind, sichern Sie diese und stellen Sie sie dem Systemdienst zur Verfügung. Es ist möglich, dass Datenbank-Inkonsistenzen entstanden sind. Möglicherweise müssen auch die openUTM-LU62-Instanz und der LU6.2-Partner kalt gestartet werden um die Situation zu bereinigen. Genaueres hierzu siehe Meldung 421.

426 Problem an der XAP-TP-Schnittstelle.
&EVENT erhalten.

An der XAP-TP-Schnittstelle, die zur Kommunikation zwischen openUTM-LU62 und UTM-Anwendung benutzt wird, ist ein Fehler aufgetreten.

Verständigen Sie den Systemdienst.

427 Fehler &ERRNR beim Aufruf von &CALL.

Bei einem UNIX-Systemaufruf ist ein Fehler aufgetreten. Dies kann zu einer abnormalen Beendigung von openUTM-LU62 führen.

Meldungen der Komponente RS

510 Die LU (Alias-Name = &ALIASNAME, Netzname = &NETNAME)
hat eine negative Antwort auf eine Exchange Logname Nachricht gesendet. Diese LU hat einen warm/kalt- oder einen Log-Namen-Fehler festgestellt.

Die ferne LU, identifiziert durch den Alias-Namen *&ALIASNAME* in der LU6.2-Basis-Software (z. B. TRANSIT) bzw. durch den Netzwerknamen *&NETNAME* im SNA-Netz, hat einen schwerwiegenden Fehler beim Austausch der Log-Namen festgestellt. Eine mögliche Ursache ist ein Kaltstart von openUTM-LU62 obwohl der LU6.2-Partner noch Transaktionen resynchronisieren muss.

Falls noch Transaktionen resynchronisiert werden müssen und die Synclog-Datei von openUTM-LU62 noch nicht gelöscht ist, sollten Sie einen Warmstart versuchen, da sonst Gefahr von Datenbank-Inkonsistenzen besteht. Wenn dies nicht möglich ist, müssen Sie entweder die Partner-LU kalt starten oder beim LU6.2-Partner noch bestehende Transaktionen („Logical Units of Work“) abnormal beenden. Solange der Fehler noch nicht behoben ist, werden von openUTM-LU62 keine OSI-TP-Dialoge mit Functional Unit Commit und keine Conversations mit Sync-Level 2 unterstützt.

511 Die LU (Alias-Name = &ALIASNAME, Netzname = &NETNAME)
hat einen Kaltstart versucht, die lokale LU hat jedoch noch LUWs (logical units of work) von einer frueheren Aktivierung zu resynchronisieren.

Der LU6.2-Partner hat einen Kaltstart durchgeführt, obwohl noch Transaktionen (Logical Units of Work) zu openUTM-LU62 bestehen, die noch resynchronisiert werden müssen. Wenn möglich sollte der Kaltstart des LU6.2-Partners rückgängig gemacht werden, da ansonsten die Gefahr von Datenbankinkonsistenzen besteht. Wenn dies nicht möglich ist, kann die Situation nur durch Beendigen der openUTM-LU62-Instanz und nachfolgenden Kaltstart bereinigt werden.

512 Die LU (Alias-Name = &ALIASNAME, Netzname = &NETNAME)
hat andere Daten von einer frueheren Aktivierung gespeichert als die lokale LU.

Der LU6.2-Partner hat einen Warmstart durchgeführt, wobei die bei ihm gespeicherten Daten (Netzwerkname der eigenen LU, eigener Log-Name, Log-Name von openUTM-LU62) nicht mit den Werten übereinstimmen, die openUTM-LU62 jeweils dafür gespeichert hat.

Diese Fehlersituation ist so schwerwiegend, dass openUTM-LU62 nun keine OSI-TP-Dialoge mit Functional Unit Commit und keine Conversations mit Sync-Level 2 mehr unterstützt, bis alle Sessions zum LU6.2-Partner abgebaut und danach die Log-Namen erfolgreich ausgetauscht werden.

513 Formatfehler festgestellt in einer von der LU
(Alias-Name = &ALIASNAME, Netzname = &NETNAME) empfangenen Syncpoint Nachricht.

In einer der Syncpoint-Nachrichten „Exchange Log-Name“ oder „Compare States“, die von der Partner-LU (Alias-Name in der LU6.2-Basis-Software- und UTM-Generierung = *&ALIASNAME*, Netzwerkname im SNA-Netz = *&NETNAME*) empfangen wurde, war ein Formatfehler enthalten. In diesem Fall weist zwar openUTM-LU62 alle angekommenen OSI-TP-Dialoge mit Functional Unit Commit, die auf das Ergebnis des Austausches der Log-Namen warten, zurück, der Fehler wird jedoch nicht als permanenter Fehler angesehen. D.h. vor dem Aufbau der nächsten Conversation mit Sync-Level 2 wird auf jeden Fall ein erneuter Austausch der Log-Namen angestoßen, unabhängig davon, ob die Zahl der Sessions zum LU6.2-Partner inzwischen auf 0 gesunken war oder nicht.

Wenn der Formatfehler immer wieder auftritt, ist keine transaktionsgesicherte Kommunikation mit dem LU6.2-Partner möglich.

515 Die Partner LU (Alias-Name = &ALIASNAME, Netzname = &NETNAME)
hat einen Protokollverstoss in einer von der lokalen LU abgeschickten Syncpoint Nachricht festgestellt.

Die Partner-LU (Alias-Name in der LU6.2-Basis-Software- und openUTM-Genrierung = *&ALIASNAME*, Netzwerkname im SNA-Netz = *&NETNAME*) hat in einer Syncpoint-Nachricht von openUTM-LU62 einen Formatfehler festgestellt. Die Reaktion von openUTM-LU62 ist dieselbe wie in Meldung 513 beschrieben:

Abweisen der wartenden OSI-TP-Dialoge mit Functional Unit Commit, aber Wiederholung des Austauschs der Log-Namen bei der nächsten Gelegenheit.

Wenn der Formatfehler immer wieder auftritt, ist keine transaktionsgesicherte Kommunikation mit dem LU6.2-Partner möglich.

516 Ein Austausch der Log-Namen mit der LU
(Alias-Name = &ALIASNAME, Netzname = &NETNAME)
ist fehlgeschlagen. Diese LU hat einen kalten Exchange Logname gesendet. Die Ursache war vermutlich ein vom Partner festgestellter warm/kalt-Fehler.

Die Partner-LU (Alias-Name in der LU6.2-Basis-Software- und UTM-Generierung = *&ALIASNAME*, Netzwerkname im SNA-Netz = *&NETNAME*) hat beim Austausch der Log-Namen einen schwerwiegenden Fehler, vermutlich einen warm/kalt-Fehler festgestellt. Ab diesem Zeitpunkt sind keine Conversations mit Sync-Level 2 mehr möglich. In der Regel hilft in diesem Fall nur das administrative Beenden aller noch bestehenden Transaktionen (Logical Units of Work) beim LU6.2-Partner oder gleich ein Kaltstart des Partners. Beachten Sie jedoch, dass dabei Datenbank-Inkonsistenzen entstehen können.

520 Aktiver Aufbau einer Conversation zum Resync TP X'06F2' fehlgeschlagen:
Allokationsfehler (Code = &ERRTXT)

Im Moment lässt sich keine Conversation zum Resync-TP des LU6.2-Partners aktiv aufbauen, so dass kein Austausch der Log-Namen und kein Abgleich von Transaktionszuständen möglich ist. Der Fehlercode *&ERRTXT* enthält dann im Klartext den Returncode des entsprechenden LU6.2-Aufrufs. openUTM-LU62 weist auch hier alle auf das Ergebnis des Austausches der Log-Namen wartenden OSI-TP-Dialoge mit Functional Unit Commit zurück, wiederholt den Austausch aber bei der nächsten Gelegenheit.

521 Aktiver Aufbau einer Conversation zum Resync TP X'06F2' fehlgeschlagen:
Inkonsistenz in den Konfigurationen von openUTM-LU62 und
der LU6.2-Basis-Software (LOC_LU_ALIAS, REM_LU_ALIAS)

Aufgrund der Inkonsistenz der Konfigurierungen der LU6.2-Basis-Software (z. B. TRANSIT) und openUTM-LU62 sind derzeit überhaupt keine Conversations zwischen openUTM-LU62 und dem LU6.2-Partner möglich. Ebenso wie bei Meldung 321 beschrieben, kann die Fehlersituation nur durch Stoppen, Umgenerieren und erneutes Starten der LU6.2-Basis-Software und/oder openUTM-LU62 beseitigt werden.

523 Aktiver Aufbau einer Conversation zum Resync TP X'06F2' fehlgeschlagen:
Der Sync-Level AP_CONFIRM_SYNC_LEVEL wird von der
LU6.2-Basis-Software nicht unterstuetzt

openUTM-LU62 hat zum Austausch der Log-Namen bzw. zum Abgleich von Transaktionszuständen versucht, eine Conversation zum Resync-TP des LU6.2-Partners aufzubauen. Diese Conversation benutzt immer den Sync-Level CONFIRM (1). Die LU6.2-Basis-Software lässt jedoch zwischen der lokalen und fernen LU nur Conversations mit dem Sync-Level NONE (0) zu. In TRANSIT muss hierzu der Konfigurationsparameter PAIR_EXT geändert werden.

Da das Resync-TP nur dann angestoßen wird, wenn der OSI-TP-Partner (d.h. die UTM-Anwendung) Transaktionssicherung wünscht, sollte der Sync-Level im Parameter PAIR_EXT gleich auf den Wert SYNCLEVEL gesetzt werden - aber natürlich nur, wenn der LU6.2-Partner dies auch unterstützt.

524 Aktiver Aufbau einer Conversation zum Resync TP X'06F2' fehlgeschlagen: LU6.2-Basis-Software laeuft nicht!

Derzeit ist keine Kommunikation über LU6.2 möglich, weil die LU6.2-Basis-Software (also bei TRANSIT das TRANSIT-Grundsystem oder der LU62Mgr) nicht gestartet ist. Alle OSI-TP-Dialoge zum OSI-TP-Partner (zur UTM-Anwendung) werden daher im Moment zurückgewiesen.

525 Aktiver Aufbau einer Conversation zum Resync TP X'06F2' wegen Konfigurationsfehlers zurueckgewiesen: Der lokale LU-Alias-Name &ALIASNAME ist nicht konfiguriert!

Diese Meldung wird ausgegeben, wenn der erste Aufruf zum Start des lokalen Resync-TP schiefgeht, weil der angegebene lokale LU-Alias-Name *&ALIASNAME* nicht in der LU6.2-Basis-Software konfiguriert ist. Die Maßnahmen in diesem Fall sind bei der Meldung 521 beschrieben.

526 Aktiver Aufbau einer Conversation zum Resync TP X'06F2' fehlgeschlagen: Systemfehler (errno = &ERRNO) aufgetreten: &ERRTXT

Bei dem Versuch, eine Conversation zum LU6.2-Partner aufzubauen, ist in der LU6.2-Basis-Software (z. B. TRANSIT) irgendein Systemaufruf schiefgegangen. Wie bei den Meldungen 311 und 314 beschrieben, stellt TRANSIT keinen Fehlertext zur Verfügung, als *&ERRTXT* erscheint dann also immer:

no interpretation of errno with TRANSIT as LU6.2 software

Dieser temporäre Fehler führt nur zum Abweisen aller OSI-TP-Dialoge mit Functional Unit Commit zum OSI-TP-Partner (d.h. zur UTM-Anwendung), die auf das Ende des Austausches der Log-Namen warten. Ein misslungener „Compare States“ für eine bestimmte LUW wird nach einer gewissen Wartezeit wiederholt.

530 Basic Conversation fuer TP &TPNAME abgelehnt: Das einzig moegliche TP, das basic Conversations unterstuetzt, ist das Resync TP X'06F2'.

Der LU6.2-Partner hat zum TP *&TPNAME* von openUTM-LU62 eine basic Conversation aufgebaut. Da basic Conversations nur ganz speziellen Transaktionsprogrammen vorbehalten sind, von denen openUTM-LU62 ausschließlich das Resync-TP X'06F2' unterstützt, weist openUTM-LU62 die Conversation mit dem Code TP_NOT_AVAIL_NO_RETRY zurück.

531 Ankommende Conversation fuer das Resync TP X'06F2' abgelehnt:
Der angezeigte Sync Level &SYNCLEVEL ist nicht gleich AP_CONFIRM_SYNC_LEVEL!

Der LU6.2-Partner hat eine Protokollverletzung begangen, indem er das Resync-TP mit einem Sync-Level ungleich CONFIRM (1) gestartet hat. Diese Meldung zeigt also ein krasses Fehlverhalten des LU6.2-Partners an.

532 Ankommende Conversation fuer das Resync TP X'06F2' abgelehnt:
Die CCR Syntax ist gar nicht vereinbart!

Da ein Application Context UDTAC oder UDTDISAC konfiguriert wurde und damit auf OSI-TP-Seite keine Transaktionssicherung möglich ist, sind auch Conversations mit Sync-Level 2 zum LU6.2-Partner nicht erlaubt. Ein Austausch der Log-Namen über das Resync-TP ist also nicht notwendig. Wenn Sie Conversations mit Sync-Level 2 verwenden wollen, müssen Sie einen Application Context UDTSEC oder UDTCCR verwenden. Wenn Sie keine derartigen Conversations benutzen wollten, können Sie die Meldung ignorieren.

533 Ankommende Conversation fuer das Resync TP X'06F2' abgelehnt:
Initialisierungsdaten werden nicht unterstuetzt!

Diese Meldung zeigt einen Protokollverstoß des LU6.2-Partners an, der beim Aufbau der Conversation zum Resync-TP sogenannte PIP-Daten mitgesendet hat.

534 Ankommende Conversation fuer das Resync TP X'06F2' abgelehnt:
Der Alias-Name &ALSNAME der Partner-LU stimmt nicht mit dem konfigurierten Namen &CONFNAME ueberein.

Der LU6.2-Partner will eine Conversation zum lokalen Resync-TP aufbauen. Der LU-Name des LU6.2-Partners stimmt aber nicht mit dem bei openUTM-LU62 mittels REM-LU-ALIAS generierten Namen überein. Es liegt folglich ein Fehler in der Gesamtkonfigurierung vor. Eine auf dem lokalen Rechner residierende LU, die von openUTM-LU62 als Stellvertreter-LU für eine UTM-Anwendung benutzt werden soll, kann eine Kommunikation nur zu einer bestimmten Partner-LU unterhalten.

540 Conversation zum Resync TP X'06F2' abgebrochen:
Die LU6.2-Basis-Software ist gestoppt worden!

541 Conversation zum Resync TP X'06F2' abgebrochen:
Conversation Failure!

Diese Fehlermeldung wird ausgegeben, wenn die Conversation zum Resync-TP durch einen schwerwiegenden Fehler auf der darunterliegenden Session abgebrochen wird, etwa durch Ausfall der Link-Verbindung zur Partner-PU oder durch administratives Beenden aller bestehenden Sessions zur Partner-LU (`u62_adm -as`). Da dieser Fehler damit nur temporärer Natur ist, werden zwar alle OSI-TP-Dialoge mit Functional Unit Commit, die auf das Ende des Austausches der Log-Namen warten, abgelehnt. Beim Eintreffen des nächsten OSI-TP-Dialogs mit Functional Unit Commit wird allerdings der Austausch der Log-Namen erneut angestoßen. Sie müssen also nach Behebung des Leitungsproblems openUTM-LU62 nicht wieder neu starten.

Weitere Meldungen der Komponente BD auf Windows-Systemen

900 Fehler beim Oeffnen des Listener-Sockets, errno &ERRNO (&ERRTEXT)

Jede Instanz von openUTM-LU62 richtet beim Start einen Listener-Socket ein, um ankommende Verbindungswünsche von den Administrationsprogrammen entgegenzunehmen. Geht die Systemfunktion socket() schief, wird diese Meldung ausgegeben, und die Instanz beendet sich.

901 Fehler beim Binden des Listener-Sockets an Port, errno &ERRNO (&ERRTEXT)

Für openUTM-LU62 ist kein freier Port >= 5000 verfügbar. Die Instanz beendet sich nach der Ausgabe dieser Fehlermeldung.

902 Fehler beim Erzeugen der Port-Datei &PORTFILE,
errno &ERRNO (&ERRTEXT)

903 Fehler beim Schreiben in die Port-Datei &PORTFILE,
errno &ERRNO (&ERRTEXT)

904 Fehler beim Schreiben in die Port-Datei &PORTFILE,
geschrieben: &ANZ1 Bytes, erwartet: &ANZ2 Bytes

Beim Erzeugen oder beim Beschreiben der Datei &PORTFILE, die in binärer Form den Port des Listener-Sockets der Instanz beinhalten soll, ist ein Fehler aufgetreten in der Systemfunktion open() oder write(). Die Instanz beendet sich daraufhin.

905 Fehler beim Annehmen einer ankommenden TCP/IP-Verbindung,
errno &ERRNO (&ERRTEXT)

Die Systemfunktion accept() zum Annehmen einer ankommenden Verbindungsanforderung von einem Administrationsprogramm ist mit dem Fehlercode ERRNO schiefgegangen. Dies führt jedoch nicht zum Abbruch, sondern ist nur als Hinweis zu verstehen.

10.2 Meldungen des XAP-TP-Providers

Die Meldungen des XAP-TP-Providers werden in die gleiche Datei wie die Meldungen des Programms `u62_tp` ausgegeben. Sie beginnen mit dem Buchstaben 'P'. Die Werte der Inserts sind entweder im Anschluss an die Meldung erläutert oder - für mehrfach vorkommende Inserts - in Abschnitt „Allgemeine Inserts der XAP-TP-Meldungen“ auf Seite 334 aufgeführt.

Ohne Verwendung von TNSX werden in den folgenden Meldungen die lokalen Access Points und Partner-Adressen wie folgt dargestellt:
T'LOCTSEL'(26544)
T'REMTSEL'(host:5632)
Die entsprechenden Inserts sind im Folgenden mit (*) markiert.

P001 Fehler beim OSS-Aufruf (&XPFUNC):
&XPRET, &XPERR, &XP1INFO, &XP2INFO

Diese Meldung wird ausgegeben, wenn der Aufruf einer OSS-Funktion (*&XPFUNC*) einen Fehler liefert. Handelt es sich um einen vom Transportsystem gemeldeten Fehler, wird zusätzlich die Meldung P012 ausgegeben.

Die Inserts haben folgende Bedeutung:

&XPFUNC	Name der OSS-Funktion
&XPRET	siehe Tabelle auf Seite 334
&XPERR	siehe Tabelle auf Seite 334
&XP1INFO	OSS-Zusatzinformation
&XP2INFO	OSS-Zusatzinformation

P002 Fehler beim Assoziationsaufbau (&XPFUNC)
&ACPNT, &OSLPAP, &XPRET, &XPERR, &XP1INFO, &XP2INFO

Diese Meldung wird ausgegeben, wenn der Aufruf einer OSS-Funktion (*&XPFUNC*), die zum Aufbau einer Association benötigt wird, einen Fehler liefert. Handelt es sich um einen vom Transportsystem gemeldeten Fehler, wird zusätzlich die Meldung P012 ausgegeben. Ist es kein vom Transportsystem gemeldeter Fehler, wird die Anwendung mit “Termapplication” beendet.

Die Inserts haben folgende Bedeutung:

&XPFUNC	Name der OSS-Funktion
&ACPNT	Name des lokalen ACCESS-POINT (*)
&OSLPAP	Name des Partners in der lokalen Anwendung (*)
&XPRET	siehe Tabelle auf Seite 334

&XPERR siehe Tabelle auf Seite 334

&XP1INFO OSS-Zustatzinformation

&XP2INFO OSS-Zustatzinformation

P003 Assoziation abgelehnt (a_assin()):
&ACPNT, Grund: &XPRJCT, Laenge: &XPLTH

Diese Meldung wird ausgegeben, wenn der Aufbau einer Association von außen abgelehnt wird.

Die Inserts haben folgende Bedeutung:

&ACPNT Name des lokalen ACCESS-POINT (*)

&XPLTH fehlerhafte Länge

&XPRJCT siehe Tabelle auf Seite 337

P004 Assoziation abgelehnt (a_assin()):
&ACPNT
&OSLPAP
Grund: &XPRJCT

Diese Meldung wird ausgegeben, wenn der Aufbau einer Association von außen abgelehnt wird.

Die Inserts haben folgende Bedeutung:

&ACPNT Name des lokalen ACCESS-POINT (*)

&OSLPAP Name des Partners in der lokalen Anwendung (*)

&XPRJCT siehe Tabelle auf Seite 337

P005 Assoziation abgelehnt (a_assin()):
&ACPNT, Grund: Partner unbekannt,
Partner-Adresse: (*)

Diese Meldung wird ausgegeben, wenn der Aufbau einer Association von außen abgelehnt wird, weil der ferne Partner in der lokalen Anwendung nicht bekannt ist.

Die Inserts haben folgende Bedeutung:

&ACPNT Name des lokalen ACCESS-POINT (*)

P006 Assoziation abgelehnt (a_assin()):
&ACPNT, &OSLPAP, Grund: Falscher Application Context Name
(&XP0OBID, &XP1OBID, &XP2OBID, &XP3OBID, &XP4OBID,
&XP5OBID, &XP6OBID, &XP7OBID, &XP8OBID, XP9OBID)

Diese Meldung wird ausgegeben, wenn der Aufbau einer Association von außen abgelehnt wird. Der Application Context Name für den fernen Partner stimmt nicht mit dem in der lokalen Anwendung für diesen Partner generierten Application Context Namen überein.

Die Inserts haben folgende Bedeutung:

&ACPNT Name des lokalen ACCESS-POINT (*)

&OSLPAP Name des Partners in der lokalen Anwendung (*)

&XP0OBID - &XP9OBID
sind die (maximal) zehn Elemente des Objektbezeichners, die den Application Context Namen des fernen Partners bilden.
Für nicht belegte Elemente wird *-1* ausgegeben.

P007 Fehler beim Assoziationsaufbau(a_assrs()):
&ACPNT, &OSLPAP, &XPRET, &XPERR, &XP1INFO, &XP2INFO

Diese Meldung wird ausgegeben, wenn der Aufruf der OSS-Funktion *a_assrs()*, mit der der Associationsaufbau von außen beantwortet wird, einen Fehler liefert. Handelt es sich um einen vom Transportsystem gemeldeten Fehler, wird zusätzlich die Meldung P012 ausgegeben.

Die Inserts haben folgende Bedeutung:

&ACPNT Name des lokalen ACCESS-POINT (*)

&OSLPAP Name des Partners in der lokalen Anwendung (*)

&XPRET siehe Tabelle auf Seite 334

&XPERR siehe Tabelle auf Seite 334

&XP1INFO OSS-Zusatzinformation

&XP2INFO OSS-Zusatzinformation

P008 Assoziation aufgebaut: &ACPNT, &OSLPAP

Diese Meldung wird ausgegeben, wenn eine Association aufgebaut wurde.

Die Inserts haben folgende Bedeutung:

&ACPNT Name des lokalen ACCESS-POINT (*)

&OSLPAP Name des Partners in der lokalen Anwendung (*)

P009 Assoziation abgelehnt (a_asscf()):
&ACPNT, &OSLPAP, Grund: &XPRJCT, Laenge: &XPLTH

Diese Meldung wird ausgegeben, wenn der aktive Aufbau einer Association abgelehnt wird, weil die Bestätigung des Partners nicht akzeptiert werden kann.

Die Inserts haben folgende Bedeutung:

&ACPNT Name des lokalen ACCESS-POINT (*)

&OSLPAP Name des Partners in der lokalen Anwendung (*)

&XPRJCT siehe Tabellle auf Seite 337

&XPLTH eventuell eine fehlerhafte Länge

P010 Assoziation abgelehnt (a_asscf()):
&ACPNT,
&OSLPAP,
Grund: Partner unbekannt
Partner-Adresse: &PARTADDR (*)
(&XASSREF)

Diese Meldung wird ausgegeben, wenn der aktive Aufbau einer Association abgelehnt wird, weil der ferne Partner bei der Bestätigung des Associationsaufbaus mit einer Adresse (*&XPADDR*) antwortet, die in der lokalen Anwendung nicht bekannt ist.

Die Inserts haben folgende Bedeutung:

&ACPNT Name des lokalen ACCESS-POINT (*)

&OSLPAP Name des Partners in der lokalen Anwendung (*)

&PARTADDR Tatsächliche Adresse des Partners

&XASSREF XAPTP-interne Referenz der Assoziation

P011 Assoziation abgelehnt (a_asscf()):
&ACPNT, &OSLPAP, Grund: Falscher Application Context Name
(&XP0OBID, &XP1OBID, &XP2OBID, &XP3OBID, &XP4OBID,
&XP5OBID, &XP6OBID, &XP7OBID, &XP8OBID, XP9OBID)

Diese Meldung wird ausgegeben, wenn der aktive Aufbau einer Association abgelehnt wird, weil der ferne Partner bei der Bestätigung des Associationsaufbaus mit einem Application Context Namen antwortet, der nicht mit dem in der lokalen Anwendung für diesen Partner konfigurierten Application Context Namen übereinstimmt.

Die Inserts haben folgende Bedeutung:

&ACPNT Name des lokalen ACCESS-POINT (*)

&OSLPAP Name des Partners in der lokalen Anwendung (*)

&XP0OBID - &XP9OBID
sind die (maximal) zehn Elemente des Objektbezeichners, die den Application Context Namen des fernen Partners bilden.
Für nicht belegte Elemente wird *-1* ausgegeben.

P012 CMX Diagnoseinformation:
&XPCTYPE, &XPCCLS, &XPCVAL

Diese Meldung wird ausgegeben, wenn eine vorhergehende Meldung aufgrund eines vom Transportsystem gemeldeten Fehlers ausgegeben wurde. Es wird der Diagnosecode des Transportsystems aufbereitet. In den folgenden Tabellen werden einige Werte für *&XPCTYPE*, *&XPCCLS* und *&XPCVAL* beschrieben. Eine vollständige Auflistung enthält die zu CMX gehörende Include-Datei *cmx.h*.

XPCTYPE	Bedeutung (CMX-Fehlertyp)
0	T_CMXTYPE: CMX-Fehler, der von der CMX-Bibliothek erkannt wurde
2	T_DSTEMPERR: Temporärer TNS-Fehler
3	T_DSCALL_ERR: TNS-Aufruffehler
4	T_DSPERM_ERR: Permanenter TNS-Fehler
5	T_DSWARNING: TNS-Warnung
>15	CMX-Fehler aufgrund von Fehlercodes aus dem Transportsystem

XPCCLS	Bedeutung (CMX-Fehlerklasse, gültig für &XPCTYPE < 15)
0	T_CMXCLASS: CMX-Klasse
2	T_DSNOT_SPEC: TNS-Klasse nicht spezifiziert
3	T_DSPAR_ERR: TNS-Parameterfehler
4	T_DSILL_VERS: Ungültige TNS-Version
5	T_DSSYS_ERR: TNS-Systemfehler
6	T_DSINT_ERR: Interner TNS-Fehler
7	T_DSMESSAGE: TNS-Hinweis

XPCVAL	Bedeutung (CMX-Fehlerwert)
0	T_NOERROR: Kein Fehler
5	T_EIO: Momentaner Engpass bzw. Fehler im Transportsystem
14	T_EFAULT: IO_Area nicht allokiert
100	T_UNSPECIFIED: Nicht näher spezifizierter Fehler, i. a. Fehler bei einem Systemaufruf
101	T_WSEQUENCE: Unzulässige Aufrufreihenfolge
103	T_WPARAMETER: Ungültiger Parameter
104	T_WAPPLICATION: Die Anwendung ist im TNS nicht bekannt oder die Task ist zur Anmeldung der Anwendung nicht berechtigt oder die Anwendung ist von dieser Task bereits eröffnet.
105	T_WAPP_LIMIT: Der Grenzwert für gleichzeitig aktive Anwendungen ist erreicht.
106	T_WCONN_LIMIT: Der Grenzwert für gleichzeitig aktive Verbindungen ist erreicht.
107	T_WTREF: Ungültige Transportreferenz oder Transportverbindung bereits abgebaut.
111	T_NOCCP: Das Transportsystem unterstützt die gewünschte Anwendung oder Verbindung nicht.
114	T_CCP_END: Das Transportsystem wurde beendet oder die Anwendung wurde vom Administrator geschlossen.
255	T_WLIBVERSION: Kein Anschluss an das CMX-Subsystem möglich.
-100	T_INVREF: Die evid ist ungültig. CMX kann den Aufruf keinem Wartepunkt zuordnen.

P013 Assoziation abgelehnt (a_asscf()):
&ACPNT, &OSLPAP, Grund: &XPCRES, &XPSRC, &XPNDIA
CCR V2 = &XP1BOOL, Version Incompatibility = &XP2BOOL,
ContWin Assignment rejected = &XP3BOOL,
Bid mandatory rejected = &XP4BOOL, No reason = &XP5BOOL

Diese Meldung wird ausgegeben, wenn der aktive Aufbau einer Association vom fernen Partner abgelehnt wird.

Die Inserts haben folgende Bedeutung:

&ACPNT Name des lokalen ACCESS-POINT (*)

&OSLPAP Name des Partners in der lokalen Anwendung (*)

&XPCRES gibt an, ob die Ablehnung vorübergehend oder dauerhaft ist:
0 = permanent reject (dauerhafte Ablehnung)
1 = transient reject (vorübergehende Ablehnung)

&XPCSRC gibt an, wer den Aufbau abgelehnt hat:
0 = ACSE Service User
1 = ACSE Service Provider
2 = Presentation Service Provider

&XPNDIA siehe Tabelle auf Seite 341

&XP1BOOL - &XP5BOOL
können die Werte *TRUE* und *FALSE* annehmen. Die mit *TRUE* belegten Werte geben an, welche Gründe der Partner für die Ablehnung des Associationsaufbauwunsches mitgeschickt hat:

&XP1BOOL: Die CCR Version 2 ist nicht verfügbar
&XP2BOOL: Die TP-Protokollversionen sind nicht kompatibel
&XP3BOOL: Die Contention-Winner-Zuordnung wird abgelehnt
&XP4BOOL: Die Festlegung "Bidding ist Pflicht" oder "Bidding ist nicht Pflicht" wird abgelehnt
&XP5BOOL: Es wird kein Grund angegeben

P014 Fehler beim Assoziationsaufbau (&XPFUNC)
&ACPNT, &OSLPAP, &XPRET, &XPERR, &XP1INFO, &XP2INFO

Diese Meldung wird ausgegeben, wenn der Aufruf einer OSS-Funktion, die zum Aufbau einer Association benötigt wird, einen Fehler liefert. Handelt es sich um einen vom Transportsystem gemeldeten Fehler, wird zusätzlich die Meldung P012 ausgegeben. Ist es kein vom Transportsystem gemeldeter Fehler, wird die Anwendung mit “Termapplication” beendet.

Die Inserts haben folgende Bedeutung:

&XPFUNC Name der OSS-Funktion

&ACPNT Name des lokalen ACCESS-POINT (*)

&OSLPAP Name des Partners in der lokalen Anwendung (*)

&XPRET siehe Tabelle auf Seite 334

&XPERR siehe Tabelle auf Seite 334

&XP1INFO OSS-Zusatzinformation

&XP2INFO OSS-Zusatzinformation, zur Zeit immer Null.

P015 Assoziation abgebaut (&XPFUNC)
&ACPNT, &OSLPAP, &XPLNK, &XPSRC, &XPNDIA,
&XPINI, &XP1INFO, &XP2INFO

Diese Meldung wird ausgegeben, wenn eine Association abgebaut wird.

Die Inserts haben folgende Bedeutung:

&XPFUNC Name der OSS-Funktion

&ACPNT Name des lokalen ACCESS-POINT (*)

&OSLPAP Name des Partners in der lokalen Anwendung (*)

&XPLNK repräsentiert den internen Zustand der Association
0 = Association nicht verknüpft
1 =Association mit Channel verknüpft
2 = Association mit Instanz verknüpft

&XPCSRC Verursacher des Abbaus
0 = ACSE Service User
1 = ACSE Service Provider
2 = Presentation Service Provider

&XPNDIA siehe Tabelle auf Seite 341

&XP1INFO OSS-Zusatzinformation

&XP2INFO OSS-Zusatzinformation

&XPINI siehe folgende Tabelle:

XPINI	Bedeutung
0	Die Association wurde von innen abgebaut.
401	O_LOC_TRAN Der Verursacher ist das lokale Transportsystem. &XP1INFO enthält den CMX-Returncode. Dieser wird in der Folgemeldung P012 detailliert ausgegeben.
402	O_REM_TRAN Der Verursacher ist das ferne Transportsystem. &XP1INFO enthält den Grund des CMX-Events t_disin. Die Werte sind in *cmx.h* definiert. Nachfolgend einige ausgewählte Werte von &XP1INFO: 0 (T_USER) Der Abbau erfolgte durch den Kommunikationspartner, u. U. auch durch einen Benutzerfehler auf der Partnerseite. 1 (T_RTIMEOUT) Wegen Inaktivität der Verbindung gemäß Parameter t_timeout wurde die Verbindung lokal durch CMX abgebaut. 2 (T_RADMIN) Wegen Außerbetriebnahme des CCP durch die Administration wurde die Verbindung lokal durch CMX abgebaut. 3 (R_CCPEND) Wegen CCP-Ausfall wurde die Verbindung lokal durch CMX abgebaut. 256 (T_RUNKOWN) Der Partner oder das CCP hat die Verbindung abgebaut. Ein Grund für den Abbau wurde nicht angegeben. 257 (T_RSAP_CONGEST) Wegen eines TSAP-spezifischen Engpasses hat das Partner-CCP die Verbindung abgebaut. 258 (T_RSAP_NOTATT) Das Partner-CCP hat die Verbindung abgebaut, weil der adressierte TSAP dort nicht angemeldet ist. 259 (T_RUNSAP) Das Partner-CCP hat die Verbindung abgebaut, weil der adressierte TSAP dort nicht bekannt ist. 261 (T_RPERMLOST) Abbau durch die Netzadministration oder Administration des Partner-CCP 262 (T_RSYSERR) Fehler im Netz 385 (T_RCONGEST) Das Partner-CCP hat die Verbindung wegen Betriebsmittelengpass abgebaut.

XPINI	Bedeutung
	386 (T_RCONNFAIL) Das Partner-CCP hat die Verbindung wegen Misslingen des Verbindungsaufbaus abgebaut. 387 (T_RDUPREF) Weil für ein NSAP-Paar eine zweite Verbindungsreferenz vergeben wurde (Systemfehler), wurde die Verbindung vom Partner-CCP abgebaut. 388 (T_RMISREF) Das Partner-CCP hat die Verbindung wegen einer nicht zuzuordnenden Verbindungsreferenz (Systemfehler) abgebaut. 389 (T_PROTERR) Das Partner-CCP hat die Verbindung wegen eines Protokollfehlers (Systemfehler) abgebaut. 391 (T_PREFOFLOW) Das Partner-CCP hat die Verbindung wegen Verbindungsreferenz-Überlauf abgebaut. 392 (T_RNOCONN) Das Partner-CCP hat den Aufbau der Netzverbindung abgelehnt. 394 (T_RINLNG) Das Partner-CCP hat die Verbindung wegen falscher Header- oder Parameterlänge (Systemfehler) abgebaut. 448 (T_RLCONGEST) Das lokale CCP hat die Verbindung wegen Betriebsmittelengpass abgebaut. 449 (T_RLNOQOS) Das lokale CCP hat die Verbindung abgebaut, weil “Quality of Service” nicht mehr unterstützt wird. 451 (T_RILLPWD) Ungültiges Verbindungspasswort. 452 (RNETACC) Netzzugang wurde verweigert. 464 (T_RLPROTERR) Das lokale CCP hat die Verbindung wegen eines Transportprotokollfehlers (Systemfehler) abgebaut. 465 (T_RLINTIDU) Das lokale CCP hat die Verbindung abgebaut, weil es eine zu lange Schnittstellen-Dateneinheit erhalten hat (Systemfehler). 466 (T_RLNORMFLOW) Das lokale CCP hat die Verbindung wegen Verletzung der Flusskontrollregeln für Normaldaten (Systemfehler) abgebaut.

XPINI	Bedeutung
	467 (T_RLEXFLOW) Das lokale CCP hat die Verbindung wegen Verletzung der Flusskontrollregeln für Vorrangdaten (Systemfehler) abgebaut. 468 (T_RLINSAPID) Das lokale CCP hat die Verbindung abgebaut, weil es eine ungültige TSAP-Identifikation erhalten hat (Systemfehler). 469 (T_RLINCEPID) Das lokale CCP hat die Verbindung abgebaut, weil es eine ungültige TCEP-Identifikation erhalten hat (Systemfehler). 470 (T_RLINPAR) Das lokale CCP hat die Verbindung wegen eines unzulässigen Parameterwerts (z. B. Benutzerdaten zu lang oder Vorrangdaten nicht zugelassen) abgebaut. 480 (T_RLNOPERM) Die Administration des lokalen CCP hat den Verbindungsaufbau verhindert. 481 (T_RLPERMLOST) Die Administration des lokalen CCP hat die Verbindung abgebaut. 482 (T_RLNOCONN) Weil keine Netzverbindung verfügbar ist, konnte das lokale CCP den Verbindungsaufbau nicht durchführen. 483 (T_RLCONNLOST) Das lokale CCP hat die Verbindung wegen Verlust der Netzverbindung abgebaut. Häufigste Ursache: Generierungsfehler auf CCP- und PDN-Seite, z. B. unstimmige Link-Adressen. Als Problemursache kommen außerdem in Frage: Partner ist nicht vorhanden, Modem ist defekt oder falsch eingestellt, DÜ-Anschluss ist nicht gesteckt, DFÜ-Board defekt. 484 (T_RLNORESP) Weil der Partner nicht auf den CONRQ antwortet, kann das lokale CCP die Verbindung nicht aufbauen. 485 (T_RLIDLETRAF) Das lokale CCP hat die Verbindung wegen Verbindungsverlust (Idle Traffic Timeout) abgebaut. 486 (T_RLRESYNC) Das lokale CCP hat die Verbindung abgebaut, weil die Resynchronisierung (mehr als zehn Wiederholungen) erfolglos war. 487 (T_RLEXLOST) Das lokale CCP hat die Verbindung abgebaut, weil der Vorrangdatenkanal defekt ist (mehr als drei Wiederholungen).

XPINI	Bedeutung
403	O_LOC_SESS Der Versursacher ist der lokale Session Provider. &XP1INFO kann folgende Werte annehmen: 4 (S_PROTERROR) Protokollfehler: fehlerhafter Aufbau der Session PDU oder fehlerhafter SPDU Parameter 16 (S_PICSREST) Verstoß gegen implementierungsbedingte Beschränkungen.
404	O_REM_SESS Der Verursacher ist der entferne Session Provider. &XP1INFO kann folgende Werte annehmen: 1 (S_TCDISCON) transport disconnect 4 (S_PROTERROR) protocol error 8 (S_UNDEFINED) undefined 16 (S_PICSREST) violation against restriction stated in PICS
405	O_LOC_PRES Der Verursacher ist der lokale Presentation Provider. &XP1INFO kann folgende Werte annehmen: 0 (P_ARRNO) reason not specified – Ein intern angeforderter Dekodierpuffer kann wegen Speichermangel nicht bereitgestellt werden. – Überlauf des internen Datenpuffers beim Reassemblieren von fragmentierten Nachrichten. – Ein unbekanntes Session-Ereignis wurde angezeigt. Systemengpass oder Systemfehler. 1 (P_ARNRPDU) unrecognized PPDU – Es sind keine Session-Userdaten vorhanden oder es kann deren Presentation-Teil nicht dekodiert werden (Systemfehler). 4 (P_ARNRPAR) unrecognized PPDU parameter – Fehler beim Dekodieren der ACSE-, Presentation- oder einer Benutzersyntax. 5 (P_ARNEPAR) unexpected PPDU parameter – PPDU-Parameter nicht im normal mode.

XPINI	Bedeutung
	6 (P_ARNIPAR) invalid PPDU parameter – Ungültiger context identifier beim Dekodieren. – Ungültiger PPDU-Parameter, z. B. falsche Länge. Dieser “abort” kann vom UTM-Benutzer durch die Angabe von ungültigen Presentation-/Session-Selektoren ausgelöst werden.
406	O_REM_PRES Der Verursacher ist der ferne Presentation Provider. &XP1INFO kann folgende Werte annehmen: -1 (O_NOVALUE) optional parameter is not present 0 (P_ARNNO) reason not specified 1 (P_ARNRPDU) unrecognized PPDU 2 (P_ARNEPDU) unexpected PPDU 3 (P_ARNESSP) unexpected session service primitive 4 (P_ARNRPAR) unrecognized PPDU parameter 5 (P_ARNEPAR) unexpected PPDU parameter 6 (P_ARNIPAR) invalid PPDU parameter value
407	O_LOC_ACSE Der Verursacher ist der lokale ACSE Provider &XP1INFO hat immer den folgenden Wert: 1 (A_ABSASP) ACSE-service-provider initiated the abort Es wird die Instanz angegeben, die aus ACSE-Sicht den “abort” ausgelöst hat (“abort source”).
408	O_REM_ACSE Der Verursacher ist der entferne ACSE-Service-Provider. &XP1INFO kann folgende Werte annehmen: 0 (A_ABSASU) ACSE-service-user initiated the abort 1 (A_ABSASP) ACSE-service-provider initiated the abort

P016 Assoziation abgebaut (a_relin()):
&ACPNT, &OSLPAP, &XPLNK, &XPNDIA

Diese Meldung wird ausgegeben, wenn eine Association abgebaut wird, weil eine “release indication” empfangen wurde.

Die Inserts haben folgende Bedeutung:

&ACPNT Name des lokalen ACCESS-POINT (*)

&OSLPAP Name des Partners in der lokalen Anwendung (*)

&XPLNK repräsentiert den internen Zustand der Association
0 = Association nicht verknüpft
1 =Association mit Channel verknüpft
2 = Association mit Instanz verknüpft

&XPNDIA siehe Tabelle auf Seite 341

P017 OSS Dekodierfehler: &XPPDU, &XP1DIA, &XP2DIA, &XP3DIA

Diese Meldung wird ausgegeben, wenn OSS beim Dekodieren einer TP-, CCR- oder Benutzerdaten-PDU einen Fehler erkennt.

Die Inserts haben folgende Bedeutung:

XPPDU	Bedeutung
0	PDU_UNKNOWN
1	TP_BEGIN_DIALOGUE_RI
2	TP_BEGIN_DIALOGUE_RC
3	TP_BID_RI
4	TP_BID_RC
5	TP_EBD_DIALOGUE_RI
6	TP_END_DIALOGUE_RC
7	TP_U_ERROR_RI
8	TP_U_ERROR_RC
9	TP_ABORT_RI
10	TP_GRANT_CONTROL_RI
11	TP_REQUEST_CONTROL_RI
12	TP_HANDSHAKE_RI
13	TP_HANDSHAKE_RC
14	TP_HSK_AND_GRT_CTRL_RI
15	TP_HSK_AND_GRT_CTRL_RC
16	TP_DEFER_RI

XPPDU	Bedeutung
17	TP_PREPARE_RI
18	TP_HEURISTIC_REPORT_RI
19	TP_TOKEN_GIVE_RI
20	TP_TOKEN_PLEASE_RI
21	TP_RECOVER_RI
22	TP_INITIALIZE_RI
23	TP_INITIALIZE_RC
24	CCR_INITIALIZE_RI
25	CCR_INITIALIZE_RC
26	CCR_BEGIN_RI
27	CCR_BEGIN_RC
28	CCR_PREPARE_RI
29	CCR_READY_RI
30	CCR_COMMIT_RI
31	CCR_COMMIT_RC
32	CCR_ROLLBACK_RI
33	CCR_ROLLBACK_RC
34	CCR_RECOVER_RI
35	CCR_RECOVER_RC
50	PDU_ANY
51	PDU_UASE_RI

XP1DIA XP2DIA	Bedeutung
1	not supported parameter was received and skipped
2	received data truncated
4	required transfer syntax name missing in user data or not specified in AVX list, error codes in &XP2DIA
6	no transfer syntax name in user data though presentation negotiation was not completed
7	transfer syntax name encoded in user data not found in AVX list
10	invalid value in data structure
11	invalid object identifier in data structure
12	invalid length or count in data structure
13	invalid index in data structure (EXTERNAL, CHOICE)
14	invalid value of ax_typtag in corresponding syntax table

&XP3DIA zugehöriger Index in der Syntaxtabelle

P018 FSM Protokollfehler: &ACPNT, &OSLPAP, &XPPTYP, &XPFSMN

Diese Meldung wird ausgegeben, wenn die Finite-State-Machine einen Fehler meldet.

Die Inserts haben folgende Bedeutung:

&ACPNT Name des lokalen ACCESS-POINT (*)

&OSLPAP Name des Partners in der lokalen Anwendung (*)

&XPPTYP Typ des Service-Protokollelements

&XPFSMN Name der Finite-State-Machine

P019 APDU enthaelt ungueltigen Wert:
&ACPNT, &OSLPAP, &XPAPDU, &XP3INFO

Diese Meldung wird ausgegeben, wenn eine ungültige APDU empfangen wurde.

Die Inserts haben folgende Bedeutung:

&ACPNT Name des lokalen ACCESS-POINT (*)

&OSLPAP Name des Partners in der lokalen Anwendung (*)

&XPAPDU Typ der APDU

&XP3INFO Zusatzinformation zum Fehler

P020 OTRACE implizit ausgeschaltet. Grund: &XPTRFAIL

Diese Meldung wird ausgegeben, wenn der Versuch, einen Trace-Record zu schreiben, fehlgeschlagen ist. Als Folge des Fehlers wurde der OSS-Trace implizit ausgeschaltet. Nachdem der Fehler behoben worden ist, kann über die Administration der OSS-Trace wieder eingeschaltet werden.

Die Inserts haben folgende Bedeutung:

XPTRFAIL	Bedeutung
1	Die OSS-Funktion o_wutr() lieferte den Returncode O_ERROR. In der vorangegangenen P001-Meldung finden Sie nähere Informationen zu dem Fehler.
2	Die OSS-Funktion o_wutr() lieferte den Returncode O_INVEREF.
3	Die OSS-Funktion o_wutr() lieferte einen unbekannten Returncode.

P021 Unerwartetes Event &XPEVT fuer Association aufgetreten, Event wird ignoriert: &ACPNT, &OSLPAP, &XPOSAS, &XPASST

Diese Meldung wird ausgegeben, wenn ein Ereignis auftritt, das nicht zum momentanen Zustand der Association passt. XAP-TP berücksichtigt dieses Ereignis nicht.

Die Inserts der Meldung haben folgende Bedeutung:

&XPEVT Typ des aufgetretenen Ereignisses

&ACPNT Name des lokalen ACCESS-POINT (*)

&OSLPAP Name des Partners in der lokalen Anwendung (*)

&XPOSAS Index der betroffenen Association

*&XPASS*T Zustand der betroffenen Association

P100 Instance Allocation Timeout:
&ACPNT,
&OSLPAP,
&XPOSAS

Diese Meldung wird ausgegeben, wenn der Versuch, eine XAP-TP-Instanz für eine Verbindung zu allokieren, innerhalb einer vorgegebenen Zeitspanne gescheitert ist.

&ACPNT Name des lokalen ACCESS-POINT (*)

&OSLPAP Name des Partners in der lokalen Anwendung (*)

&XPOSAS Index der betroffenen Association

P101 CMX Error:
&ACPNT,
&OSLPAP

Diese Meldung wird ausgegeben, wenn ein CMX-Fehler aufgetreten ist. Zusätzlich wird die Meldung P012 ausgegeben.

&ACPNT Name des lokalen ACCESS-POINT (*)

&OSLPAP Name des Partners in der lokalen Anwendung (*)

Eine mögliche Ursache kann sein, dass in der TNSX-Generierung die Einträge für PSEL und SSEL fehlen.

10.2.1 Allgemeine Inserts der XAP-TP-Meldungen

XPRET	Bedeutung
2	Not first process of application
-1	Function call not successful due to permanent error.
-2	Function call not successful due to transient error. Retry the call later
-3	Function call not successful, data flow stopped Continue after event GO
-4	Session call: Expedited function call stopped due to expedited data flow control shortage Continue after event S_XGO/S_GO Presentation call: Function call not successful, apref invalid Local function call: Invalid connection reference ACSE call: Function call not successful, apref resp. aref invalid
-5	Invalid waiting point reference
-6	Invalid application reference
-7	Waiting period to obtain a lock on a shared association expired

XPERR	Bedeutung
1	No memory available (temporary)
100	Call sequence error
101	Application not attached
102	Sending of data not allowed; wait for GO event
103	Internal error
104	Shared association is not locked
200	Missing ACSE/presentation reference
201	Invalid ACSE/presentation reference
202	Presentation call: missing AVX list (o_attach) ACSE call: missing application reference
203	Presentation call: invalid AVX list ACSE call: invalid application reference

XPERR	Bedeutung
204	Presentation call: invalid abstract syntax name in AVX ACSE call: missing ACSE parameters
205	Presentation call: invalid decoding mode in AVX ACSE call: missing presentation parameters
206	Presentation call: invalid user data length ACSE call: missing session parameters
207	Presentation call: invalid context id in p_udl ACSE call: missing application context name
208	Presentation call: invalid next parameter in p_udl ACSE call: invalid application context name
209	Presentation call: invalid pdv parameter in p_udl ACSE call: invalid calling AP Title
210	Presentation call: invalid chaining parameter ACSE call: invalid calling AE Qualifier
211	Presentation call: missing token parameter ACSE call: invalid called AP Title
212	Presentation call: invalid token parameter ACSE call: invalid called AE Qualifier
213	Presentation call: missing rtype parameter ACSE call: invalid responding AP Title
214	Presentation call: invalid rtype parameter ACSE call: invalid responding AE Qualifier
215	Presentation call: missing type parameter ACSE call: missing called p_address
216	Presentation call: invalid type parameter ACSE call: invalid called p_address
217	Presentation call: invalid syncp parameter ACSE call: missing calling p_address
218	Presentation call: missing syncp parameter ACSE call: missing responding p_address
219	Presentation call: invalid ctxlst parameter ACSE call: no mode parameter
220	Presentation call: invalid number of abstract syntaxes passed to OSS ACSE call: invalid mode parameter
221	Presentation call: invalid transfer syntax name ACSE call: missing result
222	Presentation call: invalid number of transfer syntaxes ACSE call: invalid result

XPERR	Bedeutung
223	Presentation call: invalid number of abstract syntaxes ACSE call: missing result source
224	Presentation call: same abstract syntax occurred already in transparent or non-transparent mode ACSE call: invalid result source
225	Presentation call: invalid data separation parameter ACSE call: invalid diagnostic
226	ACSE call: missing reason
227	ACSE call: invalid reason
228	ACSE call: missing provider reason
229	ACSE call: invalid provider reason
230	ACSE call: missing abort source
231	ACSE call: invalid p-requirements
232	ACSE call: invalid s-requirements
233	ACSE call: invalid syntax identifier
234	ACSE call: invalid p-context identifier
235	ACSE call: invalid p-context definition list
236	ACSE call: invalid p-context definition result list
237	ACSE call: invalid result in p-context definition result list
238	ACSE call: invalid default p-context result
239	ACSE call: invalid default p-context name
240	ACSE call: invalid user data length
241	ACSE call: invalid quality of service
242	ACSE call: invalid sync point serial number
243	ACSE call: invalid tokens
244	ACSE call: invalid SS-user reference
245	ACSE call: invalid SS-common reference
246	ACSE call: invalid SS-additional reference
250	Presentation call: ASN encoding error ACSE call: ASN encoding error
251	Presentation call: ASN decoding error ACSE call: ASN decoding error
252	Presentation call: ASN: invalid value in data struct ACSE call: ASN: invalid value in data struct

XPERR	Bedeutung
253	Presentation call: ASN: invalid object id in data struct ACSE call: ASN: invalid object id in data struct
254	Presentation call: ASN: invalid length in data struct ACSE call: ASN: invalid length in data struct
255	Presentation call: ASN: invalid index in data struct ACSE call: ASN: invalid index in data struct
256	Presentation call: ASN: invalid tag in syntax table ACSE call: ASN: invalid tag in syntax table
300	Presentation call: invalid protocol state ACSE call: invalid protocol state Local function call: error on system call
301	Presentation call: protocol error ACSE call: protocol error Local function call: error on transport system call
302	Local function call: error on local function call
305	Local function call: error on session call
306	Local function call: error on presentation call
307	Local function call: error on ACSE call

XPRJCT	Bedeutung
0	NO_REJECT
1	APPLICATION_CONTEXT_NAME_TOO_LONG Der vom Partner empfangene Objektbezeichner, der den Application Context Name bildet, enthält mehr Elemente, als von openUTM unterstützt.
2	CALLING_APT_TOO_LONG Bei der Association Indication wird für den Application Process Title eine Länge angegeben, die openUTM nicht unterstützt.
3	CALLING_AEQ_TOO_LONG Bei der Association Indication wird für den Application Entity Qualifier eine Länge angegeben, die openUTM nicht unterstützt.
4	CALLED_APT_TOO_LONG Der gerufene Application Process Title ist länger als von openUTM unterstützt.
5	CALLED_AEQ_TOO_LONG Der gerufene Application Entity Qualifier ist länger als von openUTM unterstützt.
6	CONTEXT_DEFINITION_LIST_TOO_LONG Bei der Association Indication werden mehr abstrakte Syntaxen mitgegeben, als openUTM unterstützt.

XPRJCT	Bedeutung
7	CONTEXT_RESULT_LIST_TOO_LONG Die beim Associationsaufbau (Association indication bzw. confirmation) mitgegebene Liste der unterstützten abstrakten Syntaxen enthält mehr Elemente, als von openUTM unterstützt.
9	ADDRESS_NO_PSAP_INFO Die bei Association indication bzw. confirmation mitgegebene Adresse enthält keine Informationen zum PSAP.
10	ADDRESS_NO_INFO_VERS_0_PSAP Die bei Association indication bzw. confirmation mitgegebene Adresse enthält für die PSAP-Information eine falsche Version.
11	ADDRESS_INVALID_P_SEL_LENGTH Die bei Association indication bzw. confirmation mitgegebene Adresse enthält für den Presentation-Selektor eine ungültige Länge.
12	ADDRESS_NO_SSAPINFO Die bei Association indication bzw. confirmation mitgegebene Adresse enthält keine Information zum SSAP.
13	ADDRESS_NO_INFOVERS_0_SSAP Die bei Association indication bzw. confirmation mitgegebene Adresse enthält für die SSAP-Information eine falsche Version.
14	ADDRESS_INVALID_S_SEL_LENGTH Die bei Association indication bzw. confirmation mitgegebene Adresse enthält keinen gültigen Teil für den Session-Selektor.
15	ADDRESS_NO_PARTNER_MODE Die bei Association indication bzw. confirmation mitgegebene Adresse enthält keinen gültigen Teil für den Network- und Transport-Selektor.
16	ADDRESS_TNSX_ERROR Die bei Association indication bzw. confirmation mitgegebene Adresse wird vom TNS abgewiesen.
17	UNKNOWN_PARTNER Die bei Association indication bzw. confirmation mitgegebene Adresse ist in der lokalen Anwendung nicht bekannt.
18	WRONG_APPLICATION_CONTEXT_NAME Der bei Association indication bzw. confirmation mitgegebene Application Context Name entspricht nicht dem in der lokalen Anwendung generierten Application Context Namen.
19	ABSTRACT_SYNTAX_MISSING Bei Association indication bzw. confirmation werden weniger abstrakte Syntaxen unterstützt, als in der lokalen Anwendung generiert wurden.
20	OSITP_SYNTAX_MISSING Bei Association indication bzw. confirmation wird die abstrakte Syntax für OSI TP nicht unterstützt.

XPRJCT	Bedeutung
21	NO_TP_INITIALIZE Bei Association indication bzw. confirmation wird keine TP-INITIALIZE-RI/RC PDU mitgeliefert.
22	OSITP_NO_VERSION_1 Der Partner unterstützt die Version 1 des OSI-TP-Protokolls nicht.
23	OSITP_RCH_WRONG_LENGTH Der bei der TP-INITIALIZE-indication bzw. TP-INITIALIZE-confirmation mitgegebene Recovery Context Handle hat eine Länge, die von openUTM nicht unterstützt wird.
24	NO_CCR_INITIALIZE Die CCR-INITIALIZE-RI PDU fehlt.
25	CCR_NOT_VERSION_2 Der Partner unterstützt die Version 2 des CCR-Protokolls nicht.
26	SESSION_NO_FDX Die Session-Funktionalität „Full Duplex“ ist nicht gesetzt.
27	SESSION_NO_DATA_SEPARATION Die Session-Funktionalität „Data Separation“ ist nicht gesetzt, obwohl CCR im Kontext ist.
28	SESSION_NO_TYPED_DATA Die Session-Funktionalität „Typed Data“ ist nicht gesetzt, obwohl CCR im Kontext ist.
29	SESSION_NO_MINOR_SYNCHRONIZE Die Session-Funktionalität „Minor Synchronize“ ist nicht gesetzt, obwohl CCR im Kontext ist.
30	SESSION_NO_RESYNCHRONIZE Die Session-Funktionalität „Resynchronize“ ist nicht gesetzt, obwohl CCR im Kontext ist.
31	TOKEN_CONTENTION_WINNER_AND_NO_TOKEN Die lokale Anwendung ist Contention Winner, aber nicht im Besitz des „Tokens“ (nur wenn CCR im Kontext ist).
32	TOKEN_CONTENTION_LOSER_AND_TOKEN Die lokale Anwendung ist Contention Loser, aber im Besitz des „Tokens“ (nur wenn CCR im Kontext ist).
33	INITIAL_SYNC_POINT_SERIAL_NUMBER_NOT_SET Die Initial Syncpoint Serial Number ist nicht gesetzt, obwohl CCR im Kontext ist.
34	NO_MORE_CONTENTION_LOSER_ASSOCIATIONS Der Associationsaufbau von außen wird abgelehnt, weil in der lokalen Anwendung alle Contention Loser Associationen bereits aufgebaut sind.
35	NO_MORE_CONTENTION_WINNER_ASSOCIATIONS Der Associationsaufbau von außen wird abgelehnt, weil in der lokalen Anwendung alle Contention Winner Associationen bereits aufgebaut sind.

XPRJCT	Bedeutung
36	CCR_BUT_NO_PARTNER_AET Der Partner hat keinen Application Entity Title angegeben, obwohl CCR im Kontext ist.
37	CCR_BUT_NO_OWN_AET In der lokalen Anwendung ist kein Application Entity Title angegeben, obwohl CCR im Kontext ist.
38	RESPONDING_APT_TOO_LONG Der in der Association confirmation angegebene Application Process Title ist länger, als von openUTM unterstützt.
39	RESPONDING_AEQ_TOO_LONG Der in der Association confirmation angegebene Application Entity Qualifier ist länger, als von openUTM unterstützt.
40	ASS_ESTABLISHMENT_TIMEOUT Der von der lokalen Anwendung begonnene Associationsaufbau kann nicht in der vorgegebenen Zeit vollendet werden.
41	PARTNER_IS_IN_QUIET_STATE Der Associationsaufbau wird abgelehnt, weil der Partner in der lokalen Anwendung auf Quiet gesetzt ist.
42	NO_SPACE_FOR_RCH Der PutElement Aufruf zum Abspeichern des Recovery Context Handle liefert einen schlechten Returnwert.
43	REMOTE_AET_2_BIG Der Application Entity Title des Partners ist länger, als von openUTM unterstützt.
44	REMOTE_AET_CHANGED Beim Aufbau von parallelen Associationen zu einem Partner wird vom Partner nicht derselbe Application Entity Title, wie für die erste aufgebaute Association zu diesem Partner, geliefert.
45	NO_SPACE_FOR_REMOTE_AET Der PutElement Aufruf zum Abspeichern des Application Entity Title des Partners liefert einen schlechten Returnwert.
46	PARTNER_HAS_STATUS_OFF Der Aufbau der Association wird abgelehnt, weil der Partner in der lokalen UTM-Anwendung gesperrt ist (es ist STATUS=OFF gesetzt).
47	ADDRESS_PRES_ERROR Die bei der Association Indication oder Confirmation mitgelieferte Adresse konnte nicht ausgewertet werden.

XPNDIA	Bedeutung
0	NO_REASON_GIVEN
1	NO_COMMON_ACSE_VERSION Der Partner lehnt den Associationsaufbauwunsch ab, weil es keine gemeinsame ACSE Version gibt.
2	APPL_CONTXT_NAM_NOT_SUPPORTD Der Partner lehnt den Associationsaufbauwunsch ab, weil er den Application Context Name nicht unterstützt.
3	CALLING_NSEL_NOT_RECON Der Partner lehnt den Association Aufbauwunsch ab, weil beim Partner der Absender nicht richtig generiert ist (z.B. falscher N-SEL). oder (nur bei heterogener Kopplung): CALLING_AP_TITLE_NOT_RECON Der Partner lehnt den Associationsaufbauwunsch ab, weil er den Calling Application Process Title nicht kennt.
4	CALLING_AE_QUALI_NOT_RECON Der Partner lehnt den Associationsaufbauwunsch ab, weil er den Calling Application Entity Qualifier nicht kennt.
5	CALLING_AP_INVOC_ID_NOT_RECON Der Partner lehnt den Associationsaufbauwunsch ab, weil er den Calling Application Process Invocation Identifier nicht kennt.
6	CALLING_AE_INVOC_ID_NOT_RECON Der Partner lehnt den Associationsaufbauwunsch ab, weil er den Calling Application Entity Invocation Identifier nicht kennt.
7	CALLED_AP_TITLE_NOT_RECON Der Partner lehnt den Associationsaufbauwunsch ab, weil er den Called Application Process Title nicht kennt.
8	CALLED_AE_QUALI_NOT_RECON Der Partner lehnt den Associationsaufbauwunsch ab, weil er den Called Application Entity Qualifier nicht kennt.
9	CALLED_AP_INVOC_ID_NOT_RECON Der Partner lehnt den Associationsaufbauwunsch ab, weil er den Called Application Process Invocation Identifier nicht kennt.
10	CALLED_AE_INVOC_ID_NOT_RECON Der Partner lehnt den Associationsaufbauwunsch ab, weil er den Called Application Entity Invocation Identifier nicht kennt.
11	PERMANENT_FAILURE Der Partner baut die Association ab, weil ein permanenter Fehler aufgetreten ist.
12	BEGIN_TRANSACTION_REJECT Der Partner baut die Association ab, weil er den Beginn einer Transaktion ablehnt.

XPNDIA	Bedeutung
13	TRANSIENT_FAILURE Der Partner baut die Association ab, weil ein vorübergehender Fehler aufgetreten ist.
14	PROTOCOL_ERROR Der Partner baut die Association ab, weil ein Protokollfehler aufgetreten ist.
15	UNRECOGNIZED_PDU Die Association wird von außen mit P-ABORT abgebaut, weil die Presentation Schicht eine unbekannte Presentation PDU erhalten hat.
16	UNEXPECTED_PDU Die Association wird von außen mit P-ABORT abgebaut, weil die Presentation Schicht eine unerwartete Presentation PDU erhalten hat.
17	UNEXPECTED_SESSION_SERVICE_PRIMITIVE Die Association wird von außen mit P-ABORT abgebaut, weil die Session Schicht eine unerwartete Session Service Primitive erhalten hat.
18	UNRECOGNIZED_PDU_PARAMETER Die Association wird von außen mit P-ABORT abgebaut, weil die Presentation Schicht einen unbekannten PPDU Parameter erhalten hat.
19	UNEXPECTED_PPDU_PARAMETER Die Association wird von außen mit P-ABORT abgebaut, weil die Presentation Schicht einen unerwarteten PPDU Parameter erhalten hat.
20	INVALID_PPDU_PARAMETER_VALUE Die Association wird von außen mit P-ABORT abgebaut, weil die Presentation Schicht einen ungültigen PPDU Parameterwert erhalten hat.
21	RELEASE_NORMAL Die Association wird vom Partner mit release abgebaut. Der Partner gibt als Grund release normal an.
22	RELEASE_URGENT Die Association wird vom Partner mit release abgebaut. Der Partner gibt als Grund release urgent an.
23	RELEASE_USER_DEFINED Die Association wird vom Partner mit release abgebaut. Der Partner gibt als Grund user defined an
24	IDLE_TIMEOUT_ABORT Die Association wird von der lokalen Anwendung abgebaut, weil die Association in der mit IDLETIME generierten Zeit nicht benutzt worden ist.

10.3 Meldungen der Dienstprogramme

10.3.1 Meldungen von u62_start

Alle Meldungen von `u62_start` mit Ausnahme der Meldung 18 werden nach `stdout` ausgegeben. Die Meldungen 17 und 18 werden auf UNIX-Systemen in die Datei

`/opt/lib/utmlu62/PROT/prot.`*luname*

auf Windows-Systemen in die Datei

`Programme\utmlu62\PROT\prot.`*luname*`.txt`

geschrieben.

02 Aufruf:
u62_start [-l <LU-Name>] [-c|-k] [-t on[,<Trace-Optionen>]]
-l LU-Name: Name der lokalen LU der openUTM-LU62-Instanz
-c: Kaltstart von openUTM-LU62
-k: „Lauwarmer“ Start von openUTM-LU62
-t on: Explizite Spezifikation des Trace durch Angabe
von Optionen moeglich (durch Komma getrennt):
IN[=<Level>]: mit Instanz-Trace
XAP: mit XAP-TP-Provider Trace

`u62_start` wurde mit falschen Schaltern oder Parametern aufgerufen. openUTM-LU62 wird nicht gestartet.

03 Wechsel ins Basis-Directory &DIRNAME nicht moeglich,
errno &ERRNO (&ERRTEXT)

04 Das Directory &DIRNAME laesst sich nicht anlegen, errno &ERRNO (&ERRTEXT)

Sind bei Aufruf von `u62_start` die Unterverzeichnisse `PROT` und auf UNIX-Systemen `.pipes` von `/opt/lib/utmlu62` (bzw. `Programme\utmlu62`) nicht vorhanden (z.B. beim ersten Start nach der Installation von openUTM-LU62), so legt `u62_start` diese vor dem Aufruf von `u62_tp` an. Geht der entsprechende Systemaufruf (mkdir) schief, so gibt diese Meldung genauere Auskunft über die Fehlerursache:
&ERRNO = Inhalt der Systemvariablen `errno`
&ERRTXT = Kurzerklärung zu *&ERRNO*

05 Das Directory &DIRNAME kann nicht gelesen werden,
errno &ERRNO (&ERRTEXT)

06 Die Konfigurationsdatei &FILENAME kann nicht geoeffnet werden,
errno &ERRNO (&ERRTEXT)

07 Fehler beim Einlesen der Konfigurationsdatei &FILENAME, errno &ERRNO (&ERRTEXT)

08 Die Version von u62_start stimmt nicht mit der Konfigurationsdatei &FILENAME ueberein (&VERS1 statt &VERS2)

Die von `u62_gen` erzeugte Konfigurationsdatei *&FILENAME* enthält eine falsche Versionsnummer. Entweder handelt es sich bei *&FILENAME* um gar keine Konfigurationsdatei oder es wurde eine inkompatible Version von `u62_gen` zur Generierung verwendet.

09 Der lokale LU-Name &LUNAME ist nicht konfiguriert.

10 Die Instanz fuer den lokalen LU-Namen &LUNAME laeuft bereits.

11 Nicht genug Speicherplatz vorhanden

12 Interner Fehler aufgetreten

13 System Call fork() schiefgegangen, errno &ERRNO (&ERRTEXT)

`u62_start` benutzt den Systemaufruf fork(), um im Hintergrund zu verschwinden, so dass der Aufruf nicht blockiert. Geht dieser Aufruf schief, gibt `u62_start` mit dieser Meldung die Fehlerursache aus. In der Regel ist die maximale Anzahl von Prozessen (auf dem System oder für den Benutzer) überschritten, was durch Änderung der Systemeinstellungen zu beheben ist.

Auf Windows-Systemen wird anstelle von fork() der Systemaufruf CreateProcess() benutzt.

14 System Call fork() zum Start der Instanz fuer den lokalen LU-Namen &LUNAME ist schiefgegangen, errno &ERRNO (&ERRTEXT)

Genau wie bei Meldung 13 beschrieben, liegt auch hier ein Mangel an Systemressourcen vor.

15 System Call execv() zum Start der Instanz fuer den lokalen LU-Namen &LUNAME ist schiefgegangen, errno &ERRNO (&ERRTEXT)

Fehler beim Systemaufruf execv(). Auf Windows-Systemen wird anstelle von execv() der Systemaufruf CreateProcess() benutzt.

16 Das Oeffnen der Protokoll-Datei &FILENAME ist schiefgegangen, errno &ERRNO (&ERRTEXT)

Vermutlich sind Systemressourcen aufgebraucht (Dateisystem voll, maximale Anzahl von I-Nodes erreicht, ...).

17 * * * openUTM-LU62 gestartet: * * *
lokaler LU-Alias-Name = &LUNAME,
PID = &INSTPID,
Start-Typ = c|w|k

Diese Meldung wird immer beim Start einer Instanz ausgegeben.

18 * * * Ende von openUTM-LU62 * * *

`u62_start` gibt in die Protokolldatei einer gestarteten Instanz von openUTM-LU62 eine Start- und Endemeldung aus, jeweils versehen mit der genauen Uhrzeit. *&INSTPID* in der Startmeldung 17 gibt die Prozess-Id des Programms `u62_tp` an, die die Protokollumsetzung für die lokale LU *&LUNAME* vornimmt.

19 Der aktuelle User ist nicht zum Start von openUTM-LU62 berechtigt!

Benutzer des UNIX-Systems (ungleich root), die nicht in der Konfigurationsdatei `u62_users` aufgelistet sind, sind nicht zum Start von openUTM-LU62-Instanzen berechtigt.

20 Variable U62_INST_DIR (Installationsverzeichnis) nicht gesetzt!

Die Variable U62_INST_DIR wird während des Installationsvorgangs in die Registry von Windows-Systemen eingetragen und muss demnach immer gesetzt sein. Ist dieser Eintrag aus der Registry gelöscht, bricht `u62_start` sofort ab.

21 UTM-LU62 laeuft als Service.

Dieser Hinweis wird nur in die Datei `stderr.txt` ausgegeben.

22 stdout kann nicht umgeleitet werden in &FILENAME,
errno &ERRNO (&ERRTEXT)

23 stderr kann nicht umgeleitet werden in &FILENAME,
errno &ERRNO (&ERRTEXT)

Wird openUTM-LU62 als Service gestartet, besteht keine Verbindung zu einem Terminal, an das Meldungen ausgegeben werden können. Daher werden stdout und stderr beim Start als Service in eine Datei umgelenkt. Geht dies schief, beendet sich openUTM-LU62 sofort wieder mit einer bei beiden obigen Fehlermeldungen.

24 Id. des Basis-Thread = &ID.

25 Id. des Service-Thread = &ID.

Beim Start als Service werden nur zur Information die Thread-Id's des Basis- und des eigentlichen Service-Threads in die Datei `stdout.txt` ausgegeben.

26 Service Control Dispatcher erfolgreich zurueckgekehrt.

Diese Meldung wird - mit einem Zeitstempel versehen - nach erfolgreicher Rückkehr des Service-Dispatcher, d.h. der Systemfunktion, die den eigentlichen Dienst startet, in die Datei `stdout.txt` ausgegeben.

27 Service Control Dispatcher fehlgeschlagen,
Fehlercode &ERRNO (&ERRTEXT)

Diese Meldung wird - mit einem Zeitstempel versehen - nach einer fehlerhaften Rückkehr des Service-Dispatchers, d.h. der Systemfunktion, die den eigentlichen Dienst startet, in die Datei `stderr.txt` ausgegeben.

28 Warten auf das Ende des Service Thread ...

Nach dem Beenden von openUTM-LU62 als Dienst muss noch abgewartet werden, bis sich der Service-Thread beednet hat, bevor man das Programm verlässt. Um zu verdeutlichen, wie lange dieser Vorgang dauert, wird die Meldung 28 alle 100 Millisekunden wiederholt.

29 Service Thread hat sich beendet.

Diese Meldung - versehen mit einem Zeitstempel - zeigt an, wann sich der Service-Thread endgültig beendet hat.

30 Fehler beim Installieren des Service:
Funktion &FUNK,
Fehlercode &ERRNO (&ERRTEXT)

31 Service &SERVICENAME erfolgreich installiert.

Die Meldungen 30 und 31 erscheinen nach Eingabe des Kommandos
`u62_start -r`

32 Stoppen des Service &SERVICENAME.

Diese Meldung erscheint nach Eingabe des Kommandos `u62_start -r`, wenn openUTM-LU62 gerade läuft. Danach wird jede Sekunde ein Punkt ausgegeben, bis sich der Dienst tatsächlich beendet hat.

33 Service &SERVICENAME ist jetzt gestoppt.

34 Service &SERVICENAME kann nicht gestoppt werden,
Fehlercode &ERRNO (&ERRTEXT)

Hinweis nach dem Kommando `u62_start -u`, ob openUTM-LU62 erfolgreich gestoppt werden konnte. Im Fehlerfall wird der Error-Code &ERRNO der Systemfunktion QueryServiceStatus() ausgegeben.

35 Service &SERVICENAME erfolgreich entfernt!

36 Fehler beim Entfernen des Service &SERVICENAME:
Funktion &FUNKNAME,
Fehlercode &ERRNO (&ERRTEXT)

Hinweis, ob openUTM-LU62 erfolgreich als Dienst entfernt werden konnte. Im Fehlerfall wird der Name der Systemfunktion ausgegeben, die den Fehler verursacht hat (OpenSCManager, OpenService oder DeleteService), dazu der Fehlercode &ERRNO.

37 Fehler bei der Bekanntgabe des Service Handlers:
Funktion &FUNKNAME,
Fehlercode &ERRNO (&ERRTEXT)

Wurde openUTM-LU62 als Dienst gestartet, muss es dem System eine Funktion bekanntgeben, die den Auftrag zum Beenden des Service bearbeitet. Geht dies schief, wird die Fehlermeldung 37 ausgegeben. Der Fehlercode ERRNO der Systemfunktion FUNKNAME gibt Aufschluss ueber die genaue Fehlerursache. openUTM-LU62 beendet sich daraufhin.

38 Fehler beim Setzen des Service-Status:
Funktion &FUNKNAME,
Fehlercode &ERRNO (&ERRTEXT)

Der Dienst openUTM-LU62 gibt dem System zu unterschiedlichen Zeitpunkten Auskunft darüber, in welchem Zustand sich der Dienst im Moment befindet. Geht die entsprechende Systemfunktion &FUNKNAME schief, wird die Meldung 38 ausgegeben. openUTM-LU62 beendet sich daraufhin.

39 Fehler beim Erzeugen eines neuen Thread,
Fehlercode &ERRNO (&ERRTEXT)

Nachdem alle Instanzen von openUTM-LU62 erfolgreich gestartet worden sind, erzeugt `u62_start` Threads, deren Aufgabe lediglich darin besteht, auf das Ende der Instanzen zu warten und danach Aufräumarbeiten durchzuführen. Geht der entsprechende Funktionsaufruf CreateThread() schief, wird diese Meldung ausgegeben und openUTM-LU62 beendet sich.

40 Aufruf:
Zum Start der Instanzen von openUTM-LU62:
u62_start [-l <LU-Name>] [-c|-k] [-t on[,<Trace-Optionen>]]
-l LU-Name: Name der lokalen LU der openUTM-LU62-Instanz
-c: Kaltstart von openUTM-LU62
-k: „Lauwarmer“ Start von openUTM-LU62
-t on: Explizite Spezifikation des Trace durch Angabe
von Optionen moeglich (durch Komma getrennt):
IN[=<Level>]: mit Instanz-Trace
XAP: mit XAP-TP-Provider Trace
Zum (De)Registrieren von openUTM-LU62 als Dienst:
u62_start -r [-l <LU-Name>] zum Registrieren
u62_start -u [-l <LU-Name>] zum Deregistrieren

`u62_start` wurde mit falschen Schaltern oder Parametern aufgerufen. openUTM-LU62 wird nicht gestartet.

42 Service &SERVICENAME erfolgreich gestartet!

Diese Meldung wird beim Start von openUTM-LU62 auf Windows-Systemen ausgegeben.

43 Fehler beim Starten des Service &SERVICENAME:
Funktion &FUNKNAME,
Fehlercode &ERRNO (&ERRTEXT)

Werden auf Windows-Systemen eine oder mehrere openUTM-LU62-Instanzen direkt aus der Kommandozeile gestartet, so wird für jede gestartete Instanz gegebenenfalls zunächst ein System-Dienst mit dem Namen &SERVICENAME eingerichtet und gestartet. Kann der System-Dienst nicht gestartet werden, so erscheint die Meldung 43, wobei &FUNKNAME der Name der fehlgeschlagenen Systemfunktion (OpenSCManager, OpenService, StartService) ist und &ERRNO und &ERRTEXT den Fehlercode angeben.

10.3.2 Meldungen von u62_sta

02 Aufruf:
u62_sta [-l <LU-Name>] [-b]
u62_sta -c [-b]

03 Fehler beim Erzeugen der Input-Pipe &PIPEFILE:
errno &ERRNO (&ERRTEXT)

04 Fehler beim Oeffnen der Input-Pipe &PIPEFILE:
errno &ERRNO (&ERRTEXT)

05 Instanz &INST:
Fehler beim Oeffnen der Output-Pipe &PIPEFILE:
errno &ERRNO (&ERRTEXT)

Vermutlich handelt es sich um einen temporären Mangel an Systemressourcen.

06 Instanz &INST:
Instanz befindet sich noch in der Initialisierungsphase oder bereits in der Endebehandlung.

07 Instanz &INST:
Fehler beim Schreiben in die Output-Pipe &PIPEFILE:
errno &ERRNO (&ERRTEXT)

Diese Meldung sollte nur in Ausnahmesituationen erscheinen, etwa bei einem Mangel an Systemressourcen oder bei einer Kollision mit der Beendigung der openUTM-LU62-Instanz *&INST*.

08 Instanz &INST:
Fehler beim Schreiben in die Output-Pipe &PIPEFILE:
Anzahl geschriebener Bytes fehlerhaft.

09 Instanz &INST:
Instanz &PID (&PID) reagiert nicht in angemessener Zeit.

Nachdem `u62_sta` seine Anfrage an die Instanz $INST von openUTM-LU62 gestellt hat, antwortet das Programm `u62_tp` nicht innerhalb einer fest vorgegebenen Zeit. Diese maximale Wartezeit beträgt 5 Sekunden. `u62_sta` fährt dann mit der Statusanfrage an die nächste openUTM-LU62-Instanz fort. Eine eventuell verspätet eintreffende Quittung von der openUTM-LU62-Instanz *&INST* wird stillschweigend verworfen. Solch drastische Verzögerungen können durchaus bei höherer Last und großen Konfigurationen auftreten.

10 Instanz &INST:
Fehler beim Lesen aus der Input-Pipe &PIPEFILE:
errno &ERRNO (&ERRTEXT)

11 Instanz &INST:
Fehler beim Lesen aus der Input-Pipe &PIPEFILE:
Anzahl gelesener Bytes fehlerhaft.

12 Keine openUTM-LU62-Instanz aktiv.

13 Die angegebene Instanz ist nicht aktiv.

16 Instanz &INST:
Kommando nicht erkannt.

Vermutliche Ursache: `u62_sta` und `u62_tp` gehören aufgrund einer fehlerhaften Installation von openUTM-LU62 zu unterschiedlichen Versionen.

17 Instanz &INST:
Unerwarteter Returncode &RETCODE empfangen.

Vermutliche Ursache: `u62_sta` und `u62_tp` gehören aufgrund einer fehlerhaften Installation von openUTM-LU62 zu unterschiedlichen Versionen.

80 Instanz &INST: Fehler beim Verbindungsaufbau ueber Sockets:
errno &ERRNO (&ERRTEXT)

`u62_sta` kann die Socketverbindung zur Instanz &INST nicht aufbauen. Der Fehlercode ERRNO stammt von der Systemfunktion connect().

81 Instanz &INST: Fehler beim Absenden der Meldung zum Gateway:
errno &ERRNO (&ERRTEXT)

82 Instanz &INST: Fehler beim Absenden der Meldung zum Gateway:
Anzahl geschriebener Bytes fehlerhaft.

Die Systemfunktion send() zum Absenden des Requests an die Instanz, hat nicht das erwartete Ergebnis, nämlich die Länge der Nachricht in Byte, geliefert. Dieses Fehlverhalten führt jedoch nicht zum Abbruch des Administrationsprogramms.

83 Instanz &INST: Fehler beim Abholen der Quittung vom Gateway:
errno &ERRNO (&ERRTEXT)

84 Instanz &INST: Fehler beim Abholen der Quittung vom Gateway:
Anzahl gelesener Bytes fehlerhaft.

Die Systemfunktion recv() zum Abholen der Quittung von der Instanz &INST, hat nicht das erwartete Ergebnis gebracht, was zum Abbruch des Programms führt. Wird als Instanz ??? ausgegeben, so stehen noch mindestens zwei Quittungen von unterschiedlichen Instanzen aus.

10.3.3 Meldungen von u62_adm

Die Meldungen 03 bis 13 sowie die Meldungen 16 und 17 haben jeweils die gleiche Bedeutung wie die entsprechende Meldung des Programms `u62_sta`. Diese Meldungen deuten auf einen Fehler in der Kommunikation zwischen dem Administrationsprogramm (`u62_sta` oder `u62_adm`) und dem Programm `u62_tp` hin.

02 Aufruf:
u62_adm [-l <LU-Name>] -ton [,<Trace-Optionen]
u62_adm [-l <LU-Name>] -tof [,<Trace-Optionen>]
u62_adm [-l <LU-Name>] -tfl [,<Trace-Optionen>]
u62_adm [-l <LU-Name>] -co
u62_adm [-l <LU-Name>] -cs
u62_adm [-l <LU-Name>] -do
u62_adm [-l <LU-Name>] -ds
u62_adm [-l <LU-Name>] -ao
u62_adm [-l <LU-Name>] -as
u62_adm [-l <LU-Name>] -e
u62_adm [-l <LU-Name>] -v
u62_adm -f [-p] -o <Ausgabe> <Trace 1> [<Trace 2> ...]

14 Instanz &INST:
Konfiguration der LU6.2-Software und openUTM-LU62 inkonsistent.

Diese Fehlermeldung ist nur im Zusammenhang mit den Schaltern -cs, -ds und -as möglich, also mit Kommandos, die in `u62_tp` die LU6.2-Schnittstelle zur LU6.2-Basis-Software (z. B. TRANSIT) betreffen. Sie zeigt im Falle von TRANSIT an, dass die lokale LU der Instanz von openUTM-LU62 (Generierungsparameter LOC-LU-ALIAS) oder die ferne LU (Generierungsparameter REM-LU-ALIAS) nicht in der TRANSIT-Konfigurierung enthalten ist oder dass es in der TRANSIT-Generierung keine geeignete PAIR-Anweisung gibt, so dass keine Kommunikation zwischen den beiden LUs möglich ist.

15 Instanz &INST: Systemfehler aufgetreten.

Ein Administrationskommando konnte aufgrund nicht näher beschriebener Gründe nicht ausgeführt werden. Betrifft das Administrationskommando die LU6.2-Schnittstelle von openUTM-LU62 (-cs, -ds, -as), so ist eine mögliche Ursache, dass die LU6.2-Basis-Software (also bei TRANSIT das TRANSIT-Grundsystem oder der LU62Mgr) nicht aktiv ist. Beim Kommando -cs kann es auch sein, dass das Partnersystem den Sessionaufbau abgelehnt hat.

20 Instanz &INST beendet sich.

21 Instanz &INST: Einschalten der Traces erfolgreich.

22 Instanz &INST: Die Traces sind bereits eingeschaltet.

23 Instanz &INST: Einschalten des Instanz-Trace erfolgreich.

24 Instanz &INST: Der Instanz-Trace ist bereits eingeschaltet.

25 Instanz &INST: Einschalten des XAP-TP-Trace erfolgreich.

26 Instanz &INST: XAP-TP-Trace ist bereits eingeschaltet.

27 Instanz &INST: Ausschalten der Traces erfolgreich.

28 Instanz &INST: Die Traces sind bereits ausgeschaltet.

29 Instanz &INST: Ausschalten des Instanz-Trace erfolgreich.

30 Instanz &INST: Der Instanz-Trace ist bereits ausgeschaltet.

31 Instanz &INST: Ausschalten des XAP-TP-Trace erfolgreich.

32 Instanz &INST: XAP-TP-Trace ist bereits ausgeschaltet.

33 Instanz &INST: Flush der Traces erfolgreich.

34 Instanz &INST: Flush des Instanz-Trace erfolgreich.

35 Instanz &INST: Flush des XAP-TP-Trace erfolgreich.

36 Instanz &INST: Aufbau einer neuen Assoziation angestossen.

37 Instanz &INST: Maximale Anzahl von Assoziationen bereits erreicht.

38 Instanz &INST: Aufbau einer neuen Session angestossen.

39 Instanz &INST: Maximale Anzahl von Sessions bereits erreicht.

40 Instanz &INST: Abbau aller freien Assoziationen angestossen.

41 Instanz &INST: Es gibt derzeit keine freien Assoziationen.

42 Instanz &INST: Abbau aller Sessions angestossen.

43 Instanz &INST: Es bestehen derzeit keine Sessions.

44 Instanz &INST: Abbau aller Assoziationen angestossen.

45 Instanz &INST: Es gibt derzeit keine Assoziationen.

50 Fehler beim Oeffnen der Ausgabedatei &FILENAME,
errno &ERRNO (&ERRTXT)

51 Fehler beim Oeffnen des Meldungskatalogs &CATNAME,
errno &ERRNO (&ERRTXT)

Für die mit dem Kommando `u62_adm -f` angestoßene Auswertung der binären Trace-Datei ist der Meldungskatalog *&CATNAME* notwendig, der sich nach einer erfolgreichen Installation von openUTM-LU62 im Verzeichnis `/opt/lib/nls/msg/En` befindet. Wenn der Katalog von `u62_adm` nicht geöffnet werden kann, ist der Katalog in der Regel von Hand gelöscht worden oder es sind die Zugriffsrechte manuell verändert worden. Der Inhalt *&ERRNO* der Variablen `errno` gibt jedenfalls genaueren Aufschluss darüber.

52 Fehler beim Oeffnen der Tracedatei &FILENAME,
errno &ERRNO (&ERRTXT)

53 Fehler beim Lesen der Tracedatei &FILENAME,
errno &ERRNO (&ERRTXT)

Aus der binären Trace-Datei *&FILENAME* konnte ein Trace-Record nicht gelesen werden; die weitere Auswertung der Datei wird abgebrochen. Diese Meldung kommt vor, wenn es sich bei *&FILENAME* um keine Trace-Datei einer Instanz von openUTM-LU62 handelt oder wenn die Trace-Datei nicht vollständig beschrieben ist. Ein vollständiges Beschreiben bewirkt man mit `u62_adm -tfl [,IN]` oder mit `u62_adm -tof [,IN]`. Außerdem wird der Trace bei Beendigung von openUTM-LU62 vollständig weggeschrieben (einzige Ausnahme: kill -9 <PID von `u62_tp`>).

54 Nicht genuegend Speicher vorhanden

Diese Fehlermeldung tritt nur in Verbindung mit dem Schalter -f zur Auswertung von Trace-Dateien auf. Das Programm hat vom Betriebssystem nicht genügend Speicherplatz erhalten. Diese Fehlermeldung kann durchaus vorkommen, wenn der Trace-Level größer oder gleich 4 war, so dass das Programm `u62_tp` auch Trace-Einträge in einer Interrupt-Behandlung geschrieben hat. Wenn das gerade beim Schreiben eines anderen Trace-Eintrags passiert, wird die interne Struktur der Trace-Datei zerstört.

55 Interner Fehler aufgetreten

56 Fehler beim Schliessen der Tracedatei &FILENAME,
errno &ERRNO (&ERRTXT)

Diese Meldung dient nur der Information des Aufrufers und hat sonst keine weitere Bedeutung; die Auswertung ist ja bereits abgeschlossen.

57 Fehler beim Schliessen des Meldungskatalogs &CATNAME,
errno &ERRNO (&ERRTXT)

Diese dient Meldung nur der Information des Aufrufers und hat sonst keine Auswirkungen, da die Auswertung aller angegebenen Trace-Dateien bereits erfolgt ist.

58 Fehler beim Extrahieren des Nachrichtenflusses mittels &FUNCTION
errno &ERRNO (&ERRTXT)

59 Das Extrahieren des Protokoll-Trace kann einige Zeit dauern. Bitte warten ...

Diese Meldung erscheint immer, wenn `u62_adm -f -p` aufgerufen wurde, um aus der Trace-Datei das Protokoll zwischen openUTM-LU62 und seinen Kommunikationspartnern (UTM-Anwendung und LU6.2-Partner, z. B. CICS) zu extrahieren. Da dies als Shell-Skript realisiert ist, lässt die Performance zu wünschen übrig. Es ist also die Geduld der Aufrufers gefordert (nicht die DEL-Taste betätigen!).

60 Der aktuelle User ist nicht zur Administration von openUTM-LU62 berechtigt!

Benutzer des UNIX-Systems (ungleich root), die nicht in der Konfigurationsdatei `u62_users` aufgelistet sind, dürfen - mit Ausnahme einer Trace-Auswertung - keine Administrationskommandos ausführen.

80 Instanz &INST: Fehler beim Verbindungsaufbau ueber Sockets:
errno &ERRNO (&ERRTEXT)

`u62_adm` kann die Socketverbindung zur Instanz &INST nicht aufbauen. Der Fehlercode ERRNO stammt von der Systemfunktion connect().

81 Instanz &INST: Fehler beim Absenden der Meldung zum Gateway:
errno &ERRNO (&ERRTEXT)

82 Instanz &INST: Fehler beim Absenden der Meldung zum Gateway:
Anzahl geschriebener Bytes fehlerhaft.

Die Systemfunktion send() zum Absenden des Requests an die Instanz, hat nicht das erwartete Ergebnis, nämlich die Länge der Nachricht in Byte, geliefert. Dieses Fehlverhalten führt jedoch nicht zum Abbruch des Administrationsprogramms.

83 Instanz &INST: Fehler beim Abholen der Quittung vom Gateway:
errno &ERRNO (&ERRTEXT)

84 Instanz &INST: Fehler beim Abholen der Quittung vom Gateway:
Anzahl gelesener Bytes fehlerhaft.

Die Systemfunktion recv() zum Abholen der Quittung von der Instanz &INST, hat nicht das erwartete Ergebnis gebracht, was zum Abbruch des Programms führt. Wird als Instanz ??? ausgegeben, so stehen noch mindestens zwei Quittungen von unterschiedlichen Instanzen aus.

10.3.4 Meldungen von u62_gen

03 Unbekannter Text in Zeile &LINENO - ab &INVTXT ueberlesen

04 Kein Komma am Anfang der Zeile erlaubt (Fehler in Zeile &LINENO)

05 Kein Komma am Ende einer Anweisung erlaubt (Fehler in Zeile &LINENO)

06 Komma erwartet in Zeile &LINENO1 oder &LINENO2

07 Komma an der falschen Stelle in Zeile &LINENO

08 Zahl erwartet (Fehler in Zeile &LINENO - ab &INVTXT)

09 Unzulaessiger Name (Fehler in Zeile &LINENO - ab &INVTXT)

10 INSTANCE muss allein in einer Zeile stehen.

20 Parameter mehrfach angegeben (Fehler in Zeile &LINENO)

21 Unzulaessige Anzahl von Elementen im OBJECT-IDENTIFIER (Fehler in Zeile &LINENO)

Ein Object Identifier besteht aus mindestens 2 und höchstens 10 ganzen Zahlen. Werden diese Grenzen nicht eingehalten, gibt `u62_gen` diese Fehlermeldung aus.

22 Syntaxfehler in Zeile &LINENO

23 Rest bis Zeile &LINENO ueberlesen

Diese Meldung wird immer zusammen mit Meldung 22 ausgegeben und zeigt an, in welcher Zeile sich der Parser nach einem Syntax-Fehler in der Eingabedatei wieder aufsynchronisert hat und mit der Bearbeitung fortfährt.

24 Name &NAME zu lang (Fehler in Zeile &LINENO)

25 Wert &VAL ausserhalb des Wertebereichs

26 Wert fuer CONNECT zu gross (Fehler in Anweisung vor Zeile &LINENO)

27 Wert fuer CONTWIN zu gross (Fehler in Anweisung vor Zeile &LINENO)

Der Wert von CONNECT bzw. CONTWIN überschreitet den durch den Generierungsparameter ASSOCIATIONS vorgegebenen Maximalwert von parallelen Verbindungen.

28 OSITP-CODE und LU62-CODE inkonsistent:
nur einer von beiden = *NO (Fehler in Anweisung vor Zeile &LINENO)

29 TNS-Name &NAME ist unzulaessig.
(Fehler in Zeile &LINENO)

30 Name &NAME bereits vergeben (Fehler in Zeile &LINENO)

31 Der aus APT und AEQ gebildete lokale AE Title ist bereits vergeben
(Fehler in Anweisung vor Zeile &LINENO)

33 Mode-Name &MODENAME ist unzulaessig (Fehler in Zeile &LINENO)

SNASVCMG ist ein reservierter Name und darf nicht als gewöhnlicher Mode angegeben werden.

34 Pflichtparameter LOC-LU-ALIAS fehlt

35 Pflichtparameter REM-LU-ALIAS fehlt

36 Pflichtparameter MODENAME fehlt

37 Pflichtparameter LOC-APT fehlt

38 Pflichtparameter REM-APT fehlt

39 Pflichtparameter LOC-AEQ fehlt

40 Pflichtparameter REM-AEQ fehlt

41 Die Konfiguration der lokalen OSI-Adresse ist unvollstaendig:
Es fehlt der Parameter LOC-AE bzw. einer der Parameter
LOC-TSEL, LOC-LISTENER-PORT

42 Die Konfiguration der entfernten OSI-Adresse ist unvollstaendig:
Es fehlt der Parameter REM-AE bzw. einer der Parameter
REM-NSEL, REM-TSEL, REM-LISTENER-PORT

43 Pflichtparameter CONTWIN fehlt

47 TNS-Eintraege und Parameter fuer den TNS-losen Betrieb
duerfen nicht gleichzeitig vorkommen.

48 Ungueltige Port-Nummer
Es duerfen nur die Ports 102 und 1025 - 65535 verwendet werden

50 Aufruf:
Generierung: u62_gen [-f <Ausgabedatei>] [<Eingabedatei>]
Reverse : u62_gen -r [<Konfigurationsdatei>]

51 Datei &FILENAME kann nicht geoeffnet werden,
errno &ERRNO (&ERRTXT)

52 Die Version von u62_gen stimmt nicht mit der Datei &FILENAME ueberein (&VERS1 statt &VERS2), oder die angegebene Datei wurde nicht mit u62_gen erzeugt.

Diese Meldung tritt nur bei Verwendung des Schalters -r auf. `u62_gen` hat festgestellt, dass die binäre Konfigurationsdatei *&FILENAME* die falsche Versionskennung *&VERS1* enthält. Diese müsste mit der Version *&VERS2* von `u62_gen` übereinstimmen. Vermutlich wurde lediglich eine neue Version von openUTM-LU62 installiert und anschließend mit `u62_gen -r` versucht, die vorliegende binäre Konfigurationsdatei in ein lesbares Format zu wandeln.

53 INSTANCE-Anweisung fehlt

54 Application Context &CONTEXT nicht definiert.
(Fehler in Zeile &LINENO)

Der angegebene Application Context *&CONTEXT* stimmt mit keinem der zulässigen Werte (UDTSEC, UDTAC, UDTDISAC, UDTCCR) überein.

55 STOP (Fehler)

`u62_gen` hat einen schwerwiegenden Fehler festgestellt und daher die Bearbeitung der Eingabedatei abgebrochen. Eine Konfigurationsdatei wurde nicht erstellt.

56 Nicht genug Speicherplatz vorhanden

57 Interner Fehler aufgetreten

58 Fehler beim Schreiben in die Datei &FILENAME,
errno &ERRNO (&ERRTXT)

Die häufigste Ursache dürften falsche Zugriffssrechte auf das Verzeichnis oder die bereits bestehende Konfigurationsdatei oder aber ein volles Dateisystem sein.

59 Fehler beim Lesen aus der Datei &FILENAME,
errno &ERRNO (&ERRTXT)

60 Generierung in die Datei &FILENAME

`u62_gen` hat keine Fehler in der Eingabedatei festgestellt und die binäre Konfigurationsdatei *&FILENAME* erzeugt. openUTM-LU62 kann nun gestartet werden.

62 Der aktuelle User ist nicht zur Generierung der Konfiguration berechtigt!

Benutzer des UNIX-Systems (ungleich root), die nicht in der Konfigurationsdatei `u62_users` aufgelistet sind, dürfen openUTM-LU62 nicht (um)konfigurieren.

Fachwörter

Adressierbare SNA-Instanz (network adressable unit)
Oberbegriff für *LU*, *PU*, *SSCP*.

Advanced program-to-program communication (APPC)
Anderer Name für das *LU6.2-Protokoll* und die zugrunde liegende Architektur. Wird manchmal auch verwendet für eine Programmschnittstelle zum Zugriff auf das *LU6.2-Protokoll* (APPC application program interface).

Agent
Bei verteilter Transaktionsverarbeitung mit *LU6.2* der Kommunikationspartner, der vom anderen Partner die Aufforderung zum *Sync-Point* (Transaktionsende) erhält.

Alternate Facility
Bezeichnung für einen Kommunikationspartner eines *CICS*-Programms, zu dem es die Kommunikation selbst mittels Allocate aufgebaut hat. Gegenteil: *Principal facility*. Bei openUTM verwendet man stattdessen den Begriff Auftragnehmer.

Anynet
Name einer Familie von IBM-Produkten, die die MPTN-Protokolle realisieren. MPTN steht für Multiprotocol Transport Networking und ermöglicht es, Anwendungen, die für ein bestimmtes Transportprotokoll geschrieben wurden, auf anderen Transportprotokollen ablaufen zu lassen. Wichtigstes Beispiel ist „APPC über TCP/IP“, manchmal auch „SNA über TCP/IP“ oder „SNA/IP“ genannt. Damit können sich zwei *APPC*-Anwendungen über ein TCP/IP-Netz hinweg unterhalten. Um dies zu ermöglichen, muss in beiden Endsystemen ein Anynet-Produkt installiert sein. Auf Windows-Systemen bietet das Produkt IBM Communications Server for Windows diese Funktionalität.

Assoziation (Association)
In der OSI-Begriffswelt eine Kommunikationsbeziehung zwischen zwei Application Entities. In openUTM wird anstelle des Begriffs Assoziation meist der Begriff parallele Verbindung benutzt. Einer Assoziation entspricht bei *SNA* eine *Session*.

Attach Header

Siehe *Function Management Header*.

Back-end Transaction

Im *CICS*-Sprachgebrauch ein Anwendungsprogramm, das von einem anderen Anwendungsprogramm mittels *LU6.1* oder *LU6.2* gestartet wird. Bei openUTM wird statt dessen der Begriff Auftragnehmer-Vorgang verwendet.

Backout

Rücksetzen einer Transaktion.

Basic conversation

Typ einer *LU6.2-Conversation*, bei dem das Anwendungsprogramm die Verpakkung der Daten mittels *GDS* selbst durchführen muss. Gegenteil: *Mapped Conversation*. Basic Conversations werden üblicherweise nicht von normalen Anwendungsprogrammen, sondern nur von Systemprogrammen benutzt. Der Wiederanlauf (*Resync*) im *LU6.2-Protokoll* ist z.B. mit basic conversation realisiert.

Bidder

Synonym für *contention-loser*. Gegenteil: *first-speaker*.

BIND

SNA-Nachricht, die beim Sessionaufbau von der *primary LU* an die *secondary LU* gesendet wird. Die *secondary LU* antwortet mit einem BIND-Response. Mit diesen beiden Nachrichten werden alle Eigenschaften der *Session* festgelegt, unter anderem der Typ der *Session* (z.B. *LU6.1* oder *LU6.2*), Pacing, *RU-Sizes*, *Sync-Level*, *Mode-Name*, *Contention-Winner/Contention-Loser*.

Bracket-Protokoll

Das Bracket-Protokoll in *SNA* ist eine Vereinbarung, wie mehrere Nachrichten innerhalb des Datenflusses einer LU-LU-*Session* zu einer Verarbeitungseinheit zusammengefasst werden. Das Bracket-Protokoll unterstützt den Austausch von transaktionsorientierten Nachrichten mit Richtungswechseln bei einer Auftraggeber/Auftragnehmer-Beziehung.

Chain of RUs

Festlegung innerhalb des *LU6.1-Protokolls*, wie die Benutzerdaten zu transferieren sind. Chain of RUs legt fest, dass Benutzerdaten in Blöcke mit maximal der Größe von *RU-Size* aufzutrennen sind. Der erste Block erhält den Indikator „Begin of Chain“, der letzte „End of Chain“, alle anderen „Middle of Chain“. Alternativ können die Benutzerdaten auch als *VLVB* transferiert werden. Im UTM-Anwendungsprogramm sind diese beiden Alternativen nicht sichtbar, im *CICS*-Anwendungsprogramm sind sie jedoch sichtbar.

Cluster Controller

Synonym für *Typ-2.0-Knoten*, jedoch mehr aus der Hardware-Sicht. Typisches Beispiel ist eine 3174 von IBM.

Common Programming Interface for Communication (CPI-C)

Von IBM und X/Open definierte Programmschnittstelle für die *LU6.2*- und OSI-TP-Kommunikation.

Common Programming Interface for Resource Recovery (CPI-RR)

Von IBM definierte Programmschnittstelle für Transaktionssicherung. CPI-RR wird üblicherweise in Programmen gemeinsam mit der Programmschnittstelle CPI-C genutzt. X/Open hat anstelle der CPI-RR die Programmschnittstelle TX definiert.

Communication Controller

Synonym für *Typ-4-Knoten*, jedoch mehr aus der Hardware-Sicht. Entspricht in etwa einem BS2000/OSD-Vorrechner. Typisches Beispiel ist eine 3745 von IBM.

Compare States

Bei *LU6.2* verwendetes Protokollelement, das im Falle eines Verbindungsabbruchs oder Rechnerabsturzes die gemeinsame Transaktion synchronisiert.

Contention-Winner / Contention-Loser

Einer der beiden Kommunikationspartner einer *Session* wird zum Contention-Winner erklärt, der andere zum Contention-Loser. Der Contention-Winner darf *Brackets* autonom eröffnen. Der Contention-Loser muss beim Contention-Winner die Erlaubnis zur Eröffnung einer *Bracket* einholen. Der Contention-Winner wird manchmal auch *first-speaker* genannt, der Contention-Loser manchmal auch *bidder*.

Conversation

Logische Verbindung zwischen zwei Transaktionsprogrammen über eine *LU6.2-Session*. Eine Conversation beginnt mit dem Allocate (entspricht APRO bei openUTM) und endet mit dem Deallocate. Eine Conversation belegt über ihre gesamte Lebensdauer eine *Session* und sperrt sie damit für andere Benutzer. Zum Belegen der *Session* wird das *Bracket-Protokoll* benutzt.

Customer Information Control System (CICS)

Transaktionsmonitor von IBM. Er ist in der Funktionalität vergleichbar dem openUTM. CICS gibt es auf verschiedenen Plattformen, z.B. CICS Transaction Server for z/OS (für das Betriebssystem z/OS), CICS for iSeries (für das Betriebssystem i5/OS, früher OS/400), TXSeries for Multiplatforms (für die Betriebssysteme AIX, Solaris, HP-UX, Windows).

Data Language/1 (DL/I)

Von IBM definierte Programmschnittstelle für Datenbankzugriffe und Datenkommunikation. DL/I wird ausschließlich in IMS benutzt. Mit DL/I ist eine Kommunikation zwischen IMS und openUTM möglich.

Dependent LU

Siehe *independent LU*.

Dialog (Dialogue)

In der OSI-Begriffswelt eine bestehende Kommunikationsbeziehung zwischen 2 Vorgängen. Ein Dialog beginnt mit einem TP-BEGIN-DIALOGUE (entspricht APRO bei openUTM) und endet mit einem TP-END-DIALOGUE oder TP-ABORT. Ein Dialog belegt über seine gesamte Lebensdauer eine *Assoziation* und sperrt sie damit für andere Benutzer. In der *SNA*-Begriffswelt wird statt dessen der Begriff *Conversation* verwendet. In den openUTM-Handbüchern wird das Wort Dialog oft auch in anderen Zusammenhängen verwendet. Deshalb wird in diesem Handbuch im Zweifelsfall der Begriff OSI-TP-Dialog benutzt.

Distributed Program Link (DPL)

Variante der Programm-Kommunikation, die von CICS-Programmen genutzt werden kann. Die Kommunikation zwischen CICS und openUTM ist nicht möglich.

Distributed Transaction Processing (DTP)

Variante der Programm-Programm-Kommunikation, die von CICS-Programmen genutzt werden kann. Distributed Transaction Processing ermöglicht die Kommunikation zwischen CICS und openUTM.

Enterprise Extender

Von IBM definiertes Protokoll, das es ermöglicht, beliebige SNA-Protokolle in IP-Netzen zu transportieren. Enterprise Extender wird von IBM unter anderem als Nachfolger von *Anynet* gesehen. Enterprise Extender nutzt kein TCP, sondern das verbindungslose Protokoll UDP (User-Datagram-Protokoll). Unter anderem folgende Produkte unterstützen Enterprise Extender: IBM Communications Server (für AIX, Linux und Windows), SNAP-IX, IBM Communications Server for z/OS. TRANSIT unterstützt kein Enterprise Extender.

First-speaker session

Synonym für *contention-winner*. Gegenteil: *bidder*.

Front-end Transaction

Im *CICS*-Sprachgebrauch ein Anwendungsprogramm, das ein anderes mittels *LU6.1* oder *LU6.2* aufruft. Bei openUTM wird statt dessen der Begriff Auftraggeber-Vorgang verwendet.

Function Management Header (FMH)

Function Management Header enthalten Informationen zur Kommunikation und werden als Header vor den Nettodaten in einer *Request Unit* (RU) gesendet. Bei *LU6.1* gibt es folgende FMHs:
* FMH4: Der FMH4 kann zu Beginn einer Nachricht gesendet werden und enthält dann zusätzliche Informationen für die nachfolgende Nachricht. openUTM sendet einen FMH4, wenn beim MPUT KCDF ungleich 0 oder KCMF ungleich Leerzeichen angegeben wurde.
* FMH5 (*Attach Header*): Der FMH5 wird am Anfang einer *Bracket* gesendet und beginnt eine Auftraggeber/Auftragnehmer-Beziehung. Er enthält den Transaktionscode und den Indikator für das Datenformat (*VLVB* oder *chain of RUs*). Er kann auch am Anfang einer Antwortnachricht stehen und enthält dann nur den Indikator für das Datenformat.
* FMH6 (Scheduler FMH): Der FMH6 wird gesendet beim Übertragen eines Asynchronauftrages.
* FMH7 (Error Description): Der FMH7 sendet dem Partner eine Fehlermeldung.
Bei *LU6.2* gibt es folgende FMHs:
* FMH5 (*Attach Header*): Der FMH5 wird am Anfang einer *Conversation* gesendet und beginnt eine Auftraggeber/Auftragnehmer-Beziehung. Er enthält unter anderem Transaktionscode, Userid, Passwort, Already-Verified-Indikator, Indikator für *basic/mapped conversation*, *Sync-Level*, *LUWID*.
* FMH7 (Error Description): Der FMH7 sendet dem Partner eine Fehlermeldung.

Functional Unit Commit

Funktionsgruppe im OSI-TP-Protokoll, die zum Bilden von verteilten Transaktionen erforderlich ist. Ob die Funtional Unit Commit verwendet werden darf, wird beim Aufbau einer *Assoziation* zwischen den beiden Partnern ausgehandelt. Auf einer Assoziation, für die die Functional Unit Commit vereinbart wurde, können OSI-TP-*Dialoge* mit oder ohne Functional Unit Commit laufen. Ob die Functional Unit Commit bei einem OSI-TP-Dialog verwendet wird, entscheidet das UTM-Programm mit dem Parameter KCOF im APRO-Aufruf bzw. das LU6.2-Partnerprogramm. In *SNA* entspricht einem Dialog mit Functional Unit Commit eine *Conversation* mit *Sync-Level* 2.

General Data Stream (GDS)

Festlegung innerhalb des *LU6.2-Protokolls*, wie die Benutzerdaten zu transferieren sind. GDS legt fest, dass Benutzerdaten in Blöcke zu maximal 32763 Byte aufzutrennen sind mit jeweils einem 2 Byte großen Längenfeld und einem 2-Byte-Feld zur Beschreibung des Datentyps.

Half-Session

Bei *LU6.1* verwaltet jede *LU* eine Hälfte einer *Session*, die sogenannte Half-Session. Die Half-Session wird durch einen bis zu 8 Zeichen langen Namen, den Half-Session-Qualifier identifiziert. Diese Namen werden in den Generierungen der entsprechenden Anwendungen festgelegt, bei openUTM durch LSES und RSES. Eine *Session* ist durch die beiden Half-Session-Qualifier eindeutig identifiziert. Bei *LU6.2* gibt es keine Half-Session-Qualifier.

Host

Im *SNA*-Sprachgebrauch Synonym für *Typ-5-Knoten*.

Independent LU / dependent LU

Eine *logical unit* (LU) in einem *Typ-2.0-Knoten* ist immer eine dependent LU. Sie ist abhängig vom *SSCP* (System Service Control Point) im zugehörigen *Typ-5-Knoten*. Eine dependent LU kann höchstens eine *Session* zu einer LU im zugehörigen *Typ-5-Knoten* aufbauen. Sie ist bei dieser *Session* immer *Secondary LU*. Dependent LUs gibt es für alle LU-Typen, also unter anderem auch für *LU6.1* und *LU6.2*. Ein *Typ-2.1-Knoten* kann dagegen dependent und independent LUs beinhalten. Eine independent LU kann nur das *LU6.2-Protokoll* nutzen. Sie kann jedoch zu mehreren Partnern gleichzeitig *Sessions* unterhalten und zu einem Partner bis zu 254 *parallele Sessions*.

Information Management System (IMS)

Transaktionsmonitor und Datenbanksystem von IBM. IMS läuft nur auf dem IBM-Großrechner-Betriebssystem z/OS. IMS besteht aus den beiden zentralen Komponenten

– IMS Database Management System (IMS DB)
– IMS Transaction Manager (IMS TM)

IMS TM ist ein mit openUTM vergleichbarer Transaktionsmonitor.

Initiator / Sync point initiator

Bei verteilter Transaktionsverarbeitung mit *LU6.2* der Kommunikationspartner, der den anderen Partner zum *Sync-Point* (Transaktionsende) auffordert.

Kaltstart (cold start)

Start-Typ einer transaktionsgesicherten Anwendung, bei dem alle *Log-Records* gelöscht werden und somit das Gedächtnis über unvollständige Transaktionen verloren geht.

Logical Unit (LU)

Unter einer logical unit versteht man einen adressierbaren Kommunikationspartner in einem *SNA*-Netz. Eine LU verwaltet eine oder mehrere Kommunikationsbeziehungen (*Sessions*) zu anderen LUs (Terminals, openUTM-, *CICS*- und *IMS*-Anwendungen). Ob nur eine oder mehrere Kommunikationsbeziehungen möglich sind, hängt ab vom Typ der *physical unit* und vom Typ der LU (*independent/dependent LU*). Eine LU in einem *Typ-2.0-Knoten* kann immer nur eine *Session* zu einer anderen LU aufbauen und zwar immer nur mit demselben *SNA*-Protokoll. Je nachdem um welches Protokoll es sich dabei handelt (z.B. LU2, *LU6.1* oder *LU6.2*), bezeichnet man deshalb auch die LUs als LUs vom Typ LU2, *LU6.1* oder *LU6.2*.
Jede LU ist im *SNA*-Netz unter einem eindeutigen, maximal 8 Zeichen langen Namen, dem sogenannten LU-Namen, bekannt. Bei *independent LU6.2* wird dem achtstelligen LU-Namen noch eine ebenfalls achtstellige Netzidentifikation vorangestellt. Beides zusammen bildet dann den sogenannten vollqualifizierten Netzwerknamen der LU. Bei *dependent LUs* in einem *Typ-2.0-Knoten* erfolgt die Adressierung nicht über den LU-Namen sondern über eine sogenannte Loc-Adresse (Zahl zwischen 1 und 255). Einer LU entspricht bei openUTM in etwa ein BCAMAPPL oder ein ACCESS-POINT.

LU6.1-Protokoll

Das LU6.1-Protokoll ist Bestandteil der IBM-Netzwerkarchitektur *SNA* und definiert Methoden zur verteilten Transaktionsverarbeitung. LU6.1 wurde von IBM entwickelt für die verteilte Verarbeitung zwischen *CICS*- und *IMS*-Anwendungen auf *Hostsystemen*. Da es auch in openUTM implementiert ist, kann es auch für die verteilte Verarbeitung zwischen openUTM und *CICS* und zwischen openUTM und *IMS* benutzt werden. IBM betrachtet *LU6.2* als Nachfolger von LU6.1.

LU6.2-Protokoll

Das LU6.2-Protokoll ist Bestandteil der IBM-Netzwerkarchitektur und ist dort das neueste und modernste Protokoll. LU6.2 definiert Methoden für eine Programm-Programm-Kommunikation zwischen Anwendungen in verschiedenen Rechnern. LU6.2 kann mit und ohne Transaktionssicherung benutzt werden. Es bietet vielfältige Security-Mechanismen.

Logical Unit of Work (LUW)

Dieser Begriff wird in der IBM-Literatur verwendet anstelle des bei openUTM üblichen Begriffs Transaktion. Eine logical unit of work kann Rechner-lokal oder Rechner-übergreifend sein.

Logical Unit of Work Identifier (LUWID)

Bei *LU6.2* erhält jede Rechner-übergreifende Transaktion einen netzweit und langfristig eindeutigen Namen, die sogenannte LUWID. Der *Initiator* der Transaktion vergibt diesen Namen. Mit seiner Hilfe können sich im Falle eines Verbindungsverlustes oder Rechnerabsturzes alle an der Transaktion beteiligten Systeme über den Ausgang der Transaktion einigen.

Log-Name

Im *LU6.2-Protokoll* verwendeter Name, der eine Anwendung (z.B. eine UTM- oder *CICS*-Anwendung) über ihre gesamte Lebensdauer, d.h. zwischen zwei *Kaltstarts* charakterisiert. Die Log-Names müssen gemäß *LU6.2* zwischen den beiden *Resync TPs* ausgetauscht werden, bevor nach einem Verbindungsabbruch oder Systemausfall der Wiederanlauf (*Compare States*) gestartet wird. Das Austauschen der Log-Names dient dazu, einen einseitigen *Kaltstart* zu identifizieren.

Log-Record

Auf sicheren Speicher (im allgemeinen Plattenspeicher) geschriebene Information über den Zustand einer offenen Transaktion. Die Log-Records dienen dazu, im Fall eines Rechnerabsturzes die Transaktion wieder aufsetzen zu können. Nach Beendigung der Transaktion werden die Log-Records wieder gelöscht.

Mapped Conversation

Typ einer *LU6.2-Conversation*, bei dem sich das Anwendungsprogramm nicht um die Verpackung der Daten mittels *GDS* kümmern muss. Gegenteil: *Basic Conversation*. Normale Anwendungsprogramme nutzen üblicherweise immer Mapped Conversation.

Mode

Beschreibt die Eigenschaften von Sessions, wie z.B. Maximalzahl von parallelen Sessions, RU-Sizes, Pacing-Werte. Ein Mode hat einen *Mode-Namen*.

Mode-Name

Maximal achtstelliger Name, der beim Sessionaufbau vereinbart wird. Ein Mode-Name ist ein symbolischer Name für eine Liste von *Session*-Eigenschaften, wie z.B. Maximalzahl von *parallelen Sessions*, *RU-Sizes*, Pacing-Werte. Die zu einem Mode-Name gehörenden Eigenschaften müssen üblicherweise in mindestens einem System per Generierung festgelegt werden. Bei *LU6.1* wird der Mode-Name nur am *Host* generiert. Bei *LU6.2* und *independent LUs* müssen die Mode-Namen in allen beteiligten Systemen generiert werden.

Network Control Program (NCP)

Betriebssystem des *Communication Controller*.

Netzwerkname

Siehe *Logical Unit*.

Pacing

Mechanismus zur Datenflusskontrolle bei *SNA*. Beim Pacing gibt der Empfänger dem Sender immer nur die Erlaubnis, eine bestimmte Anzahl von *RUs* zu senden.

Parallele Session

Wenn zwischen zwei *LUs* mehr als eine *Session* besteht, so bezeichnet man diese Sessions als parallele Sessions. Parallele Sessions sind bei *LU6.1* nur möglich zwischen zwei *Typ-5-Knoten*, bei *LU6.2* nur zwischen zwei *Typ-2.1-Knoten*.

Physical Unit (PU)

Jeder Knoten in einem *SNA*-Netz enthält als *adressierbare SNA-Instanz* eine physical unit (PU). Bevor zwei *logical units* (LUs) im *SNA*-Netz eine Kommunikationsbeziehung aufbauen können, muss zuerst eine Kommunikationsbeziehung zwischen den jeweiligen PUs aufgebaut werden.
In *SNA*-Netzen gibt es unterschiedliche Typen von PUs. Dementsprechend bezeichnet man die zugehörigen Systeme als *Typ-5-Knoten*, *Typ-4-Knoten*, *Typ-2.0-Knoten* oder *Typ-2.1-Knoten*.

Primary Logical Unit (PLU) / Secondary Logical Unit (SLU)

Bei den *logical units* (LUs) unterscheidet man zwischen primary logical units (PLU) und secondary logical units (SLU). PLU bzw. SLU-Status zweier LUs werden beim SNA-Sessionaufbau (*BIND*) festgelegt. Mit Hilfe von Generierungsparametern wird eine Voreinstellung bezüglich der primary/secondary-Eigenschaft festgelegt. Bei openUTM geschieht dies im Generierungsparameter SESCHA bei *LU6.1*-Verbindungen. Bei *LU6.2* und *independent LUs* gibt es üblicherweise keine Voreinstellung per Generierung. Die primary logical unit sendet bei Sessioneröffnung Vorschläge für die Sessionparameter im sogenannten *BIND*-Request. Die secondary logical unit akzeptiert diese Vorschläge oder sendet Gegenvorschläge (*BIND*-Response).

Principal Facility

Bezeichnung für den Standard-Kommunikationspartner eines *CICS*-Programms, d.h. den Partner, der das Programm aufgerufen hat. Gegenteil: *Alternate facility*. Bei openUTM verwendet man stattdessen den Begriff Auftraggeber.

RACF (Resource Access Control Facility)

RACF ist ein Produkt der Firma IBM, mit dessen Hilfe im Großrechner-Umfeld der Zugriffsschutz auf alle Anwendungen und Ressourcen gesteuert wird.
RACF ist somit zuständig für

- Identifikation und Verifikation der User mittels Benutzerschlüssel und Passwortprüfung (Authentifizierung)
- Schutz von Ressourcen durch die Verwaltung der Zugriffsrechte (Autorisierung)
- Logging der Zugriffe auf geschützte Ressourcen (Auditing).Request Unit (RU) Nettodatenteil eines SNA-Protokollelements. Siehe auch RU-Size.

Request-Size (RU size)

Maximallänge einer *Request Unit*. Diese Länge muss üblicherweise auf beiden Systemen eingestellt werden. Sie wird dann beim Sessionaufbau ausgehandelt.

Resync (Resynchronisation)

In der *LU6.1*- bzw. *LU6.2*-Protokolldefinition festgelegtes Verfahren, das nach einem Verbindungsabbruch oder Rechnerabsturz offene verteilte Transaktionen zu konsistenten Zuständen führt.

Resync Service Transaction Program (Resync TP)

Teil des *LU6.2-Protokolls*, das im Falle von *Conversations* mit *Sync-Level* 2 benötigt wird. Das LU6.2-Protokoll legt fest, dass vor dem Aufbau einer Conversation mit Sync-Level 2 die beiden Resync TPs der jeweiligen LUs ihre *Log-Namen* austauschen müssen. Im Falle eines Verbindungsverlustes oder Systemabsturzes wird das Resync TP außerdem zum Resynchronsieren der abgebrochenen Transaktionen benötigt. Das Resync TP hat den durch LU6.2 festgelegten TP-Namen X'06F2'.

Secondary Logical Unit (SLU)

Siehe *Primary Logical Unit*.

Session

Unter einer Session versteht man eine Kommunikationsbeziehung zwischen zwei *LUs*, allgemeiner zwischen zwei *adressierbaren SNA-Instanzen*. Zwischen einer LU auf einem System mit *PU* Typ 2 und einer LU auf einem *Hostsystem* (PU Typ 5) kann nur eine Session aufgebaut werden. Man spricht hierbei auch von einer dependent Session, da der *Host* der dominierende Partner in der Session ist. Zwischen zwei LUs auf Hostsystemen oder *PUs* vom Typ 2.1 können gleichzeitig mehrere Sessions (*parallele Sessions*) aufgebaut werden. Man spricht hierbei auch von independent Sessions.
Man beachte, dass der Begriff Session in der OSI-Protokollwelt eine andere Bedeutung als die hier beschriebene hat.

Side-Information

Side-Information bezeichnet bei der Programmschnittstelle CPI-C die für den Ablauf von CPI-C-Programmen notwendige Konfiguration. In openUTM wird die Side-Information mit den Steueranweisungen für KDCDEF beschrieben. Für CPI-C-Programme, die unter IMS ablaufen, muss eine eigene Side-Information-Datei vom Systemverwalter bereitgestellt werden.

Symbolic Destination Name

Unter Symbolic Destination Name versteht man bei der Programmschnittstelle CPI-C einen symbolischen Namen, mit dem man aus dem Programm heraus eine ferne Transaktion adressieren kann. Über die Side-Information wird diesem symbolischen Namen ein konkreter Transaktionscode in einer konkreten fernen Anwendung zugeordnet. Das Konzept ist vergleichbar dem von openUTM verwendeten LTAC.

Sync-Level (Synchronisation level)

Bezeichnung bei *LU6.2*, die die Transaktionssicherheit bei verteilter Verarbeitung charakterisiert:
Bei Sync-Level 0 (None) dürfen nur Nettodaten und Fehlermeldungen gesendet werden. Quittungen sind nicht zulässig. Dies entspricht bei openUTM KCOF=B beim APRO.
Bei Sync-Level 1 (Confirm) sind neben Nettodaten und Fehlermeldungen auch einfache Quittungen zulässig. Dies entspricht bei openUTM KCOF=H beim APRO.
Bei Sync-Level 2 (Syncpoint) ist die volle Transaktionssicherheit für verteilte Transaktionen eingeschaltet. Dies entspricht bei openUTM KCOF=C beim APRO.

Sync-Point (Synchronization point)

Stelle innerhalb eines Ablaufs einer verteilten Verarbeitung, an der die gemeinsamen Ressourcen in einen definierten Zustand gebracht werden. Bei openUTM verwendet man statt dessen den Begriff Transaktionsende oder Sicherungspunkt.

Systems Network Architecture (SNA)

SNA ist die Bezeichnung für eine Reihe von IBM definierten Kommunikationsprotokollen.

System Services Control Point (SSCP)

Adressierbare SNA-Instanz in einem *Typ-5-Knoten*. Der SSCP verwaltet die Verbindungen zu allen *Typ-4-* und *Typ-2.0-Knoten* in seinem Netz und alle *Sessions* zwischen dem *Typ-5-Knoten* und einem dieser untergeordneten Knoten.

Transaction Program (TP)

Im IBM-Sprachgebrauch bezeichnet man jedes Programm, das eine Programm-Programm-Kommunikation zu anderen Programmen im Netz betreibt, als Transaction Program, und zwar unabhängig davon, ob eine echte verteilte Transaktionsverarbeitung stattfindet oder nicht. Erfolgt die Programm-Programm-Kommunikation über *LU6.2*, so wird jedes Transaction Program durch einen TP-Namen identifiziert. Dieser TP-Name entspricht dem Transaktionscode von openUTM oder *CICS*.

Transmission Header (TH)

Protokoll-Header der SNA-Schicht 3. openUTM verwendet das FID1-Format. Hierbei ist der Transmission Header 10 Bytes lang. Zwischen TRANSIT und dem IBM-Partnersystem wird das FID2-Format für den Transmission Header verwendet. Dabei ist der Transmission Header 6 Bytes lang.

Typ-2.0-Knoten

Bezeichnung für einen Knoten im *SNA*-Netz, der nur *Sessions* zu seinem übergeordneten *Typ-4-* oder *Typ-5-Knoten* aufbauen kann. Eine *LU* in einem Typ-2.0-Knoten ist immer eine *dependent LU*.

Typ-2.1-Knoten

Bezeichnung für einen Knoten im *SNA*-Netz, der einerseits die Funktionen eines *Typ-2.0-Knoten* enthält, andererseits aber auch *Sessions* zu anderen Typ-2.1-Knoten betreiben kann. Auf den *Sessions* zu anderen Typ-2.1-Knoten können nur *independent LUs* und das Protokoll *LU6.2* benutzt werden. Das Produkt TRANSIT-SERVER emuliert einen Typ-2.1-Knoten.

Typ-4-Knoten

Bezeichnung für einen Knoten, der Routingfunktionen im herkömmlichen *SNA*-Netz übernimmt. Er unterhält Verbindungen zu einem oder mehreren *Typ-5-Knoten* und einem oder mehreren *Typ-2.0/2.1-Knoten*.

Typ-5-Knoten

Bezeichnung für einen Knoten im *SNA*-Netz, der *Host*-Funktionen übernimmt, d.h. einen *SSCP* enthält. Ein Typ-5-Knoten kann gleichzeitig auch die Funktionen eines *Typ-2.1-Knotens* abdecken. Das frühere Produkt TRANSIT-CD emuliert einen *Typ-4-* und Typ-5-Knoten.

UDP (User-Datagram-Protokoll)

UDP ist ein Transportprotokoll und unterstützt den verbindungslosen Datenaustausch zwischen Rechnern.
Das UDP wurde definiert, um auch Anwendungsprozessen die direkte Möglichkeit zu geben, Datagramme zu versenden und damit die Anforderungen transaktionsorientierten Verkehrs zu erfüllen. UDP baut direkt auf dem darunter liegenden IP-Protokoll auf.
UDP hat einen minimalen Protokollmechanismus und garantiert weder die Ablieferung eines Datagrammes beim Zielpartner, noch sind Vorkehrungen gegen eine Duplizierung oder eine Reihenfolgevertauschung getroffen.

Variable length variable blocked (VLVB)

Festlegung innerhalb des *LU6.1-Protokolls*, wie die Benutzerdaten zu transferieren sind. VLVB legt fest, dass Benutzerdaten in Blöcke zu maximal 32765 Byte aufzutrennen sind mit jeweils einem 2 Byte großen Längenfeld davor. Alternativ können die Benutzerdaten auch als *chain of RUs* transferiert werden. Im UTM-Anwendungsprogramm sind diese beiden Alternativen nicht sichtbar, im *CICS*-Anwendungsprogramm sind sie jedoch sichtbar.

Virtual Telecommunications Access Method (VTAM)

Komponente in einem IBM-*Hostsystem*, die für die Datenfernverarbeitung zuständig ist. Manchmal wird stattdessen auch ACF/VTAM verwendet.

Abkürzungen

AAID	Atomic Action Identifier
ACF	Advanced Communications Function
ACSE	Association Control Service Element
AE	Application Entity
AEQ	Application Entity Qualifier
APDU	Application Protocol Data Unit
APPC	Advanced Program to Program Communication
APT	Appliction Process Title
ASCII	American Standard Code for Information Interchange
BB	Begin Bracket
BCAM	Basic Communication Access Method
CCR	Commitment, Concurrency and Recovery
CD	Change Direction
CEB	Conditional End Bracket
CICS	Customer Information Control System
CMX	Communication Manager in UNIX
CPI-C	Common Programming Interface for Communication
CPI-RR	Common Programming Interface for Resource Recovery
DPL	Distributed Program Link
DPN	Destination Process Name
DRI	Define Response Indicator
DTP	Distributed Transaction Processing
EB	End Bracket
EBCDIC	Extended Binary-Coded Decimal Interchange Code
ECI	External Call Interface
EDI	Enciphered Data Indicator
EIB	Exec Interface Block

ERI	Exception Response Indicator
ESA	Enterprise System Architecture
ET	End of Transaction
FDDI	Fiber Distributed Data Interface
FID	Format Identifier
FMH	Function Management Header
FSM	Finite State Machine
GDS	General Data Stream
IMS	Information Management System
ISC	Intersystem Communication
ISO	International Organization for Standardization
KDCS	Kompatible Datenkommunikationsschnittstelle
LU	Logical Unit
LUW	Logical Unit of Work
LUWID	Logical Unit of Word Identifier
MFS	Message Format Service
MPTN	Multiprotocol Transport Networking
MVS	Multiple Virtual Storage System
NCP	Network Control Program
OSI	Open Systems Interconnection
OSI-TP	Open Systems Interconnection - Distributed Transaction Processing
OSS	Open Systems Interconnection Services
PCMX	Portable Communication Manager in UNIX
PDI	Padded Data Indicator
PDN	Programmsystem für Datenfernverarbeitung und Netzsteuerung
PET	Preliminary End of Transaction
PID	Process Identifier
PIP	Program Initialization Parameter
PLU	Primary Logical Unit
PRN	Primary Resource Name
QRI	Queued Response Indicator
RACF	Resource Access Control Facility
RDO	Resource Definition Online

RDPN	Return Destination Process Name
RPRN	Return Primary Resource Name
RLU	Remote Logical Unit
RTI	Response Type Indicator
RTR	Ready to Receive
RU	Request Unit
SDI	Sense Data Indicator
SDLC	Synchronous Data Link Control
SSCP	System Services Control Point
SIT	System Initialization Table
SLU	Secondary Logical Unit
SNA	System Network Architecture
SNI	SNA Network Interconnection
TNSX	Transport Name Service UNIX
TP	Transaction Processing, Transaction Program
TPSU	Transaction Processing Service User
TX	Transaction Demarcation (X/Open)
UDT	Unstructered Data Transfer
UTM	Universeller Transaktionsmonitor
VLVB	Variable Length Variable Blocked
VM	Virtual Machine
VTAM	Virtual Telecommuncations Access Method
XAP	X/Open ACSE/Presentation Services
XAP-TP	X/Open ACSE/Presentation Services - Transaction Processing
XLN	Exchange Log Name

Literatur

Zu openUTM gibt es auch eine CD-ROM. Diese enthält PDF-Dateien von allen openUTM-Handbüchern.
Die Handbücher sind online unter *http://manuals.ts.fujitsu.com* zu finden oder in gedruckter Form gegen gesondertes Entgelt unter *http://manualshop.ts.fujitsu.com* zu bestellen.

Dokumentation zu openUTM

openUTM
Konzepte und Funktionen
Benutzerhandbuch

Zielgruppe
Alle, die sich einen Überblick über die Funktionsbreite und Leistungsfähigkeit von openUTM verschaffen wollen

openUTM
Anwendungen programmieren mit KDCS für COBOL, C und C++
Basishandbuch

Zielgruppe
Programmierer, die für die Programmierung von UTM-Anwendungen die Programmschnittstelle KDCS nutzen wollen

openUTM
Anwendungen generieren
Benutzerhandbuch

Zielgruppe
Anwendungsplaner, Anwendungsentwickler und Betreuer von UTM-Anwendungen

openUTM
Einsatz von openUTM-Anwendungen unter BS2000/OSD
Benutzerhandbuch

Zielgruppe
Anwendungsplaner, Anwendungsentwickler, Anwender und Betreuer von UTM-Anwendungen

openUTM
Einsatz von openUTM-Anwendungen unter UNIX- und Windows-Systemen
Benutzerhandbuch

Zielgruppe
Anwendungsplaner, Anwendungsentwickler, Anwender und Betreuer von UTM-Anwendungen

openUTM
Anwendungen administrieren
Benutzerhandbuch

Zielgruppe
Administratoren und Programmierer von Administrationsprogrammen

openUTM
Meldungen, Test und Diagnose in BS2000/OSD
Benutzerhandbuch

Zielgruppe
Anwender, Administratoren und Programmierer von UTM-Anwendungen

openUTM
Meldungen, Test und Diagnose in UNIX- und Windows-Systemen
Benutzerhandbuch

Zielgruppe
Anwender, Administratoren und Programmierer von UTM-Anwendungen

openUTM (BS2000/OSD, UNIX-Systeme, Windows NT)
Anwendungen erstellen mit X/Open-Schnittstellen
Basishandbuch

Zielgruppe
Programmierer, die für die Programmierung von UTM-Anwendungen die X/Open-Schnittstellen CPI-C, XATMI und TX nutzen wollen

openUTM
openUTM Data Marshalling mit XML
Online-Beschreibung (im Internet verfügbar)

Zielgruppe
Programmierer von UTM-Anwendungen

openUTM-Client (UNIX-Systeme)
für Trägersystem OpenCPIC
Client-Server-Kommunikation mit openUTM
Benutzerhandbuch

Zielgruppe
Organisatoren, Einsatzplaner, Programmierer von CPI-C- und XATMI-Programmen

openUTM-Client
für Trägersystem UPIC
Client-Server-Kommunikation mit openUTM
Benutzerhandbuch

Zielgruppe
Organisatoren, Einsatzplaner, Programmierer von CPI-C- und XATMI-Programmen

openUTM WinAdmin
Grafischer Administrationsarbeitsplatz für openUTM
Online-Beschreibung (im Internet verfügbar) und Online-Hilfe

Zielgruppe
Organisatoren und Administratoren von UTM-Anwendungen

openUTM (BS2000/OSD)
Anwendungen programmieren mit KDCS für Assembler
Ergänzung zum Basishandbuch
nur online verfügbar: Internet

Zielgruppe
Programmierer von UTM-Assembler-Anwendungen

openUTM (BS2000/OSD)
Anwendungen programmieren mit KDCS für Fortran
Ergänzung zum Basishandbuch
nur online verfügbar: Internet

Zielgruppe
Programmierer von UTM-Fortran-Anwendungen

openUTM (BS2000/OSD)
Anwendungen programmieren mit KDCS für Pascal-XT
Ergänzung zum Basishandbuch
nur online verfügbar: Internet

Zielgruppe
Programmierer von UTM-Pascal-XT-Anwendungen

openUTM (BS2000/OSD)
Anwendungen programmieren mit KDCS für PL/I
Ergänzung zum Basishandbuch
nur online verfügbar: Internet

Zielgruppe
Programmierer von UTM-PL/I-Anwendungen

WS4UTM (UNIX- und Windows-Systeme)
Web-Services für openUTM
nur online verfügbar: Internet

Zielgruppe
Anwendungsplaner, Anwendungsentwickler und Betreuer von UTM-Anwendungen

openUTM
Masterindex
nur online verfügbar: Internet

Zielgruppe
Alle, die mit der openUTM-Dokumentation arbeiten möchten.

Dokumentation zum openSEAS-Produktumfeld

BeanConnect for openUTM
Benutzerhandbuch

Zielgruppe
BeanConnect Entwickler und Administratoren, Administratoren von Application Servern wie z.B. OracleAS/OC4J, Deployer, EJB-Entwickler und openUTM-Administratoren.

BeanConnect for CICS
Benutzerhandbuch

Zielgruppe
BeanConnect Entwickler und Administratoren, Administratoren von Application Servern wie z.B. OracleAS/OC4J, Deployer, EJB-Entwickler und CICS-Administratoren.

Biz*Transactions*
Anwendungsintegration mit Business Objekten
Online-Beschreibung (im Internet verfügbar)

Zielgruppe
Alle, die mit Business Objekten openUTM-, MVS- oder OSD-Anwendungen in Java, Windows oder openSEAS integrieren wollen.

JConnect V2.0
Verbindung von Java-Clients zu openUTM
Benutzerdokumentation und Java-Docs
(nur online verfügbar: Internet)

Zielgruppe
Programmierer von Java-Anwendungen

Web*Transactions*
Konzepte und Funktionen
Online-Beschreibung (im Internet verfügbar)

Zielgruppe
Alle, die UTM-, MVS- oder OSD-Anwendungen ans Web anbinden wollen

Web*Transactions*
Template-Sprache
Online-Beschreibung (im Internet verfügbar)

Zielgruppe
Alle, die UTM-, MVS- oder OSD-Anwendungen ans Web anbinden wollen

Web*Transactions*
Anschluss an openUTM-Anwendungen über UPIC
Online-Beschreibung (im Internet verfügbar)

Zielgruppe
Alle, die UTM-Anwendungen über UPIC ans Web anbinden wollen

Web*Transactions*
Anschluss an MVS-Anwendungen
Online-Beschreibung (im Internet verfügbar)

Zielgruppe
Alle, die MVS-Anwendungen ans Web anbinden wollen

Web*Transactions*
Anschluss an OSD-Anwendungen
Online-Beschreibung (im Internet verfügbar)

Zielgruppe
Alle, die OSD-Anwendungen ans Web anbinden wollen

Dokumentation zum BS2000/OSD-Umfeld

AID (BS2000/OSD)
Advanced Interactive Debugger
Basishandbuch
Benutzerhandbuch

Zielgruppe
Programmierer im BS2000

BCAM (BS2000/OSD)
BCAM Band 1/2
Benutzerhandbuch

Zielgruppe
Netzplaner, -generierer und -verwalter, die in BS2000-Systemen BCAM betreiben

BINDER (BS2000/OSD)
Benutzerhandbuch

Zielgruppe
Software-Entwickler

BINDER (BS2000/OSD)
Taschenbuch

Zielgruppe
Software-Entwickler, die im Umgang mit dem BINDER geübt sind

BS2000/OSD
Makroaufrufe an den Ablaufteil
Benutzerhandbuch

Zielgruppe
BS2000/OSD-Assembler-Programmierer

BS2000/OSD-BC
BLSSERV
Bindelader-Starter
Benutzerhandbuch

Zielgruppe
Software-Entwickler und geübte BS2000/OSD-Benutzer

DCAM (BS2000/OSD, TRANSDATA)
COBOL-Aufrufe
Benutzerhandbuch

Zielgruppe
Programmierer von DCAM-COBOL-Programmen

DCAM (BS2000/OSD, TRANSDATA)
Makroaufrufe
Benutzerhandbuch

Zielgruppe
Programmierer von DCAM-Assembler-Programmen

DCAM (BS2000/OSD, TRANSDATA)
Programmschnittstellen
Beschreibung

Zielgruppe
Organisatoren, Einsatzplaner, Programmierer, Systemverwalter und Netzadministratoren

FHS (BS2000/OSD, TRANSDATA)
Formatierungssystem für openUTM, TIAM, DCAM
Benutzerhandbuch

Zielgruppe
Programmierer

IFG für FHS (TRANSDATA)
Benutzerhandbuch

Zielgruppe
Datenstationsbenutzer, Anwendungsdesigner und Programmierer

FHS-DOORS (BS2000/OSD, MS-Windows)
Grafische Oberfläche für BS2000/OSD-Anwendungen
Benutzerhandbuch

Zielgruppe
End-Anwender von FHS-DOORS und Administratoren, die Masken konvertieren und diese bereitstellen

HIPLEX AF (BS2000/OSD)
Hochverfügbarkeit von Anwendungen in BS2000/OSD
Produkthandbuch

Zielgruppe
Systemverwalter und Organisatoren in Rechenzentren

IMON (BS2000/OSD)
Installationsmonitor
Benutzerhandbuch

Zielgruppe
Systembetreuung des Betriebssystems BS2000/OSD.

MT9750 (MS Windows)
9750-Emulation unter Windows
Produkthandbuch

Zielgruppe
PC-Benutzer, der mit Hilfe des Produktes MT9750 mit Anwendungen auf einem BS2000-Host kommunizieren will.

OMNIS/OMNIS-MENU (TRANSDATA, BS2000)
Funktionen und Kommandos
Benutzerhandbuch

Zielgruppe
OMNIS-Administrator und OMNIS-Anwender

OMNIS/OMNIS-MENU (TRANSDATA, BS2000)
Administration und Programmierung
Benutzerhandbuch

Zielgruppe
OMNIS-Administrator und Programmierer

OMNIS-MENU (TRANSDATA, BS2000/OSD)
Benutzerhandbuch

Zielgruppe
OMNIS-MENU-Benutzer und OMNIS-MENU-Administrator

OSS (BS2000/OSD)
OSI Session Service
User Guide

Zielgruppe
OSI TP-Anwender

RSO (BS2000/OSD)
Remote SPOOL Output
Benutzerhandbuch

Zielgruppe
Nichtprivilegierte Benutzer, RSO-Geräteverwalter, SPOOL-Verwalter und Systembetreuer des BS2000/OSD.

SECOS (BS2000/OSD)
Security Control System
Benutzerhandbuch

Zielgruppe
BS2000-Systemverwalter und BS2000-Anwender, die den erweiterten Zugriffsschutz für Dateien nutzen

SECOS (BS2000/OSD)
Security Control System
Tabellenheft

Zielgruppe
Erfahrene SECOS-Anwender

openSM2 (BS2000/OSD)
Software Monitor
Band 1: Verwaltung und Bedienung

Zielgruppe
Anwender und Systembetreuung

TIAM (BS2000/OSD, TRANSDATA)
Benutzerhandbuch

Zielgruppe
Nicht privilegierte BS2000-Anwender und Programmierer

Unicode im BS2000/OSD
Übersichtshandbuch

Zielgruppe
Anwendungsprogrammierer und Systemverwalter, die sich einen Überblick verschaffen wollen, welche Unicode-Unterstützung ihnen im BS2000/OSD geboten wird, und welche BS2000-Komponenten sie dazu benötigen.

VTSU (BS2000/OSD, TRANSDATA)
Virtual Terminal Support
Benutzerhandbuch

Zielgruppe
Anwender der Zugriffsmethoden DCAM, TIAM und UTM sowie System- und Netzverwalter

XHCS (BS2000/OSD)
8-bit-Code- und Unicode-Unterstützung im BS2000/OSD
Benutzerhandbuch

Zielgruppe
Anwender der Zugriffsmethoden DCAM, TIAM und UTM sowie Systemverwalter, Anwender, die von EHCS auf XHCS umstellen

Dokumentation zum Umfeld von UNIX-Systemen

CMX V6.0 (Solaris)
Betrieb und Administration
Benutzerhandbuch

Zielgruppe
Systemverwalter für Solaris

CMX V6.0 (Unix-Systeme)
Betrieb und Administration
Benutzerhandbuch

Zielgruppe
Systemverwalter für UNIX-Systeme

CMX V6.0
CMX-Anwendungen programmieren
Programmierhandbuch

Zielgruppe
Programmierer von Kommunikationsanwendungen auf UNIX-Systemen

OSS (SINIX)
OSI Session Service
User Guide

Zielgruppe
OSI TP-Anwender

PRIMECLUSTER*TM*
Konzept (Solaris, Linux)
Benutzerhandbuch

Zielgruppe
Benutzer und Systemadministratoren

Dokumentation zu TRANSIT

Die TRANSIT-Handbücher sind online zu finden unter
http://manuals.fujitsu-siemens.com/softbooks/software/de/prevtsit.htm.

TRANSIT-SERVER
Administration von TRANSIT
Basishandbuch Teil 1 und 2.

Zielgruppe
Für openUTM-LU6.2-Kopplung und openUTM-LU6.1-Kopplung, Systemverwalter, die für den Einsatz von TRANSIT unter UNIX-Systemen verantwortlich sind.

TRANSIT-CLIENT
Programmierschnittstelle LU0, LU0-API, FTX-LU, UTMGW1
Programmierhandbuch

Zielgruppe

Für openUTM-LU6.1-Kopplung, Anwendungsprogrammierer und TRANSIT-Adminstratoren.

Dokumentation zu CICS und IMS

Die folgende Liste gibt eine unverbindliche Übersicht über die IBM-Literatur zu CICS und IMS. Die Literaturliste bezieht sich auf die Versionen CICS Transaction Server for z/OS V3.1 und IMS Version 9.

CICS
Intercommunication Guide
Das Dokument beschreibt die unterschiedlichen Möglichkeiten in CICS zur Kommunikation mit anderen Anwendungen. Unter anderem enthält es die grundlegenden Konzepte des Distributed Transaction Processing hinsichtlich Installation, Konfiguration und Programmierung.

CICS
Distributed Transaction Programming Guide
Dieses Dokument beschreibt alle Details bei der Erstellung von CICS-Programmen mit Distributed Transaction Processing.

CICS
Application Programming Guide
Das Dokument beschreibt die Konzepte der CPI-C-Anwendungsprogrammierung. Unter anderem wird auch Distributed Transaction Processing erwähnt.

CICS
Application Programming Reference
Das Dokument enthält die vollständige Beschreibung der CICS-Programmschnittstelle.

CICS
Supplied Transactions
Beschreibung der CICS-Administrationskommandos

CICS
Resource Definition Guide
Beschreibung der Parameter zur CICS-Definition, z.B. Connection- und Session-Definition

IMS Version 9
Application Programming: Design Guide
Das Kapitel 7 beschreibt das Design von LU6.2-Anwendungsprogrammen.

IMS Version 9
Application Programming: Transaction Manager
Das Dokument enthält die Details zur Anwendungsprogrammierung mit DL/I.

IMS Version 9
Installation Volume 2: System Definition and Tailoring
Das Dokument enthält die Beschreibung der IMS-Systemdefinition, z.B. die Makros zur Definition von Transaktionscodes.

IMS Version 9
Administration Guide: Transaction Manager
In den Kapiteln 19 und 20 wird das Einrichten von LU6.2-Kopplungen beschrieben.

IMS Version 9
Command Reference
Beschreibung der IMS-Administrationskommandos

Dokumentation zu SNAP-IX und IBM Communications Server

Die folgende Liste gibt eine unverbindliche Übersicht über die Literatur der Firma Data Connection Limited zu SNAP-IX und über die IBM-Literatur zu den IBM-Communications-Server-Produkten. Die Literaturliste bezieht sich auf die Versionen

- SNAP-IX Version 7
- IBM Communications Server for Windows Version 6.1.2
- IBM Communications Server for Linux Version 6.2.2
- IBM Communications Server for AIX Version 6.3.

SNAP-IX
General Information
Das Dokument enthält allgemeine Informationen zu SNAP-IX.

SNAP-IX
Installation
Das Dokument enthält die Anleitung zur Installation von SNAP-IX auf Solaris-Systemen.

SNAP-IX
Administration Guide
Das Dokument enthält die Beschreibung der Administration von SNAP-IX, sortiert nach Aufgabenbereichen.

SNAP-IX
Administration Command Reference
Das Dokument enthält die Beschreibung der Administrations-Kommandos von SNAP-IX, alphabetisch nach dem Name der Kommandos sortiert. Es beschreibt außerdem die Generierungsparameter.

SNAP-IX
Diagnostic Guide
Das Dokument enthält die Beschreibung der Diagnosemöglichkeiten in SNAP-IX.

IBM Communications Server for Windows
Quick Beginnings
Das Dokument enthält die Einführung in das Produkt und die Installationsanleitung.

IBM Communcations Server for Windows
Network Administration Guide
Das Dokument enthält die Beschreibung der Administration, sortiert nach Aufgabenbereichen.

IBM Communications Server for Windows
Configuration File Refernce
Das Dokument enthält die Beschreibung der Generierungsparameter.

IBM Communications Server for Linux
Quick Beginnings
Das Dokument enthält die Einführung in das Produkt und die Installationsanleitung.

IBM Communications Server for Linux
Administration Guide
Das Dokument enthält die Beschreibung der Administration, sortiert nach Aufgabenbereichen.

IBM Communications Server for Linux
Administration Command Reference
Das Dokument enthält die Beschreibung der Administrations-Kommandos, alphabetisch nach dem Namen der Kommandos sortiert. Es beschreibt außerdem die Generierungsparameter.

IBM Communications Server for Linux
Diagnostics Guide
Das Dokument enthält die Beschreibung der Diagnosemöglichkeiten.

IBM Communications Server for AIX
Quick Beginnings
Das Dokument enthält die Einführung in das Produkt und die Installationsanleitung.

IBM Communications Server for AIX
Administration Guide
Das Dokument enthält die Beschreibung der Administration, sortiert nach Aufgabenbereichen.

IBM Communications Server for AIX
Administration Command Reference
Das Dokument enthält die Beschreibung der Administrations-Kommandos, alphabetisch nach dem Namen der Kommandos sortiert. Es beschreibt außerdem dieGenerierungsparameter.

IBM Communications Server for AIX
Diagnostics Guide
Das Dokument enthält die Beschreibung der Diagnosemöglichkeiten.

Sonstige Literatur

CPI-C (X/Open)
Distributed Transaction Processing
X/Open CAE Specification, Version 2
ISBN 1 85912 135 7

Reference Model Version 2 (X/Open)
Distributed Transaction Processing
X/Open Guide
ISBN 1 85912 019 9

TX (Transaction Demarcation) (X/Open)
Distributed Transaction Processing
X/Open CAE Specification
ISBN 1 85912 094 6

XATMI (X/Open)
Distributed Transaction Processing
X/Open CAE Specification
ISBN 1 85912 130 6

XML
Spezifikation des W3C (www – Konsortium)
Web page: http://www.w3.org/XML

Stichwörter

FUJITSU THE POSSIBILITIES ARE INFINITE

MANUALS
BUSINESS Products
- Industry standard servers
- BS2000/OSD mainframes
- UNIX OS based servers
- Mission critical IA64 servers
- Storage Solutions
- IT infrastructures solutions
- Software
- Deskbound devices
- Mobile devices
- Professional accessories & mainboards
- Displays & projectors

HOME & HOME OFFICE
MANUAL SHOP

Online manuals

If you are looking for Fujitsu Technology Solutions product manuals you are in the right spot. Simply choose your product and download the document in pdf format.

QUICK ACCESS

Search by product

The Quick Access search locates all available documents for your selected product. If JavaScriptâ„¢ has been enabled, suggestions will be offered as you type. Use the mouse to browse the results, left-click to insert the highlighted suggestion. **more**

ONLINE SHOP

manualshop

Most of the manuals are also available in **printed form**. To order your copies, please visit our Manual Shop at http://manualshop.ts.fujitsu.com.

COGNITAS

cognitas logo

The documents on the following internet pages have been created by our partner **cognitas Gesellschaft fÃ¼r Technik Dokumentation mbH**.

ADOBE READER

adobe logo

In order to view and print the online document in pdf format **Adobe Reader is required** (version 5 or later). To download the latest version of Adobe Reader ™, click here.

FUJITSU THE POSSIBILITIES ARE INFINITE

MANUALS

BUSINESS Products

- Industry standard servers
- BS2000/OSD mainframes
- UNIX OS based servers
- Mission critical IA64 servers
- Storage Solutions
- IT infrastructures solutions
- Software
  - Cluster Technology
  - Communications and Networking
  - Compiler
  - Document Printing and Spools
  - Management
  - Storage software
  - openSEAS
  - Transaction processing and Databases
- Deskbound devices
- Mobile devices
- Professional accessories & mainboards
- Displays & projectors

HOME & HOME OFFICE

MANUAL SHOP

SELECT

please choose