Deutsch

FUJITSU Software

openUTM-Client V4.0 für Trägersystem OpenCPIC

Client-Server-Kommunikation mit openUTM

Benutzerhandbuch

Ausgabe Februar 2017

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 9001:2008

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 9001:2008 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, ...)
- 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.

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

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

Eine UTM-Anwendung kann auf einem einzelnen Rechner als stand-alone UTM-Anwendung oder auf mehreren Rechnern gleichzeitig als UTM-Cluster-Anwendung betrieben werden.

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

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

1.1 Kurzbeschreibung des Produkts

Das Produkt “openUTM-Client V4.0, Trägersystem OpenCPIC” (kurz: OpenCPIC) ermöglicht die Client-Server-Kommunikation und gesicherte Transaktionsverarbeitung in verteilten Systemen auf der Grundlage des X/Open DTP Referenzmodells.

In diesem Handbuch wird als Produktbezeichnung immer der Kurzname OpenCPIC verwendet.

OpenCPIC bietet eine Implementierung der Programmschnittstellen

– X/Open CPI-C® (Common Programming Interface for Communication) Version 2.0

– X/Open XATMI® (X/Open Application Transaction Manager Interface)

– X/Open TX® (Transaction Demarcation Interface)

Die Implementierung basiert auf dem Protokoll OSI-TP (Open Systems Interconnection Distributed Transaction Processing).

OpenCPIC ermöglicht die Erstellung und den Betrieb von Transaktionsanwendungen in verteilten Systemen wobei Transaktionssicherung, Konsistenz der Anwenderdaten bei Datenbankzugriffen sowie Wiederanlauf unterstützt werden.

1.2 Zielsetzung und Zielgruppen des Handbuchs

Dieses Handbuch wendet sich sowohl an den Systemverwalter, der den Einsatz von OpenCPIC vorbereitet und durchführt, als auch an den Entwickler von Anwendungsprogrammen.

Es werden Kenntnisse in Unix- und Linux-Systemen und in CMX vorausgesetzt. Für die Konfigurierung der Anwendungen sind grundlegende Kenntnisse des OSI-TP Kommunikationsmodells von Vorteil.

Als Entwickler von Anwendungsprogrammen für OpenCPIC sollten Sie mit der Schnittstelle X/Open CPI-C V2.0 bzw. X/Open XATMI vertraut sein. Wenn Sie gesicherte Transaktionen durchführen wollen, sind zudem Kenntnisse der Schnittstelle X/Open TX erforderlich.

Für Datenbankzugriffe benötigen Sie Kenntnisse der entsprechenden Programmschnittstelle Ihres Datenbanksystems.

Grundsätzlich ist für das Verständnis der vorliegenden Dokumentation die Kenntnis des X/Open DTP Referenzmodells hilfreich. In Kapitel 2 wird eine kurze Einführung gegeben, die sich jedoch auf grundlegende Aspekte des Modells und die Einbettung von OpenCPIC beschränkt.

Im Literaturverzeichnis am Ende des Handbuchs finden Sie eine Liste aller für OpenCPIC relevanten X/Open-Dokumente.

1.3 Wegweiser durch die Dokumentation zu openUTM

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

1.3.1 openUTM-Dokumentation

Die openUTM-Dokumentation besteht aus Handbüchern, den Online-Hilfen für den grafischen Administrationsarbeitsplatz openUTM WinAdmin und das grafische Administrationstool WebAdmin 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-Systeme bzw. für Unix-, Linux- und Windows-Systeme gelten.

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

http://manuals.ts.fujitsu.com

Geben Sie dort in das Feld **Produktsuche** den Suchbegriff "openUTM" ein, um sich alle openUTM-Handbücher der jeweiligen Version anzeigen zu lassen.

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

Einführung und Überblick

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

Programmieren

- Zum Erstellen von Server-Anwendungen über die KDCS-Schnittstelle benötigen Sie das Handbuch **Anwendungen programmieren mit KDCS für COBOL, C und C++**, in dem die KDCS-Schnittstelle in der für COBOL, C und C++ gültigen Form 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-Spezifikation für die jeweilige X/Open-Schnittstelle.
- Wenn Sie Daten auf Basis von XML austauschen wollen, benötigen Sie das Dokument **XML für openUTM**. Darin werden die C- und COBOL-Aufrufe beschrieben, die zum Bearbeiten von XML-Dokumenten benötigt werden.
- Für BS2000-Systeme gibt es Ergänzungsbände für die Programmiersprachen Assembler, Fortran, Pascal-XT und PL/1.

Konfigurieren

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

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

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

Binden, Starten und Einsetzen

Um UTM-Anwendungen einsetzen zu können, benötigen Sie für das betreffende Betriebssystem (BS2000- bzw. Unix-, Linux- oder Windows-Systeme) das Handbuch **Einsatz von 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. Zudem wird ausführlich auf die Punkte eingegangen, die beim Betrieb von UTM-Cluster-Anwendungen zu beachten sind.

Administrieren und Konfiguration dynamisch ändern

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

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

Testen und Fehler diagnostizieren

Für die o.g. Aufgaben benötigen Sie außerdem die Handbücher **Meldungen, Test und Diagnose** (jeweils ein Handbuch für Unix-, Linux- und Windows-Systeme und für BS2000-Systeme). Sie beschreiben das Testen einer UTM-Anwendung, den Inhalt und die Auswertung eines UTM-Dumps, das 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. Das Handbuch umfasst im Wesentlichen die Beschreibung der Schnittstellen CPI-C und XATMI.
- 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.
- Das Handbuch **BizXML2Cobol** beschreibt, wie Sie bestehende Cobol-Programme einer UTM-Anwendung so erweitern können, dass sie als Standard-Web-Service auf XML-Basis genutzt werden können. Die Arbeit mit der grafischen Bedienoberfläche ist in der zugehörigen **Online-Hilfe** beschrieben.
- 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. Die Arbeit mit der grafischen Bedienoberfläche ist in der zugehörigen **Online-Hilfe** beschrieben.

Kopplung mit der IBM-Welt

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

Dokumentation zu PCMX

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

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

1.3.2 Dokumentation zum openSEAS-Produktumfeld

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

Integration von Java EE Application Servern und UTM-Anwendungen

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

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

  Die Handbücher zum Application Server von Oracle sind bei Oracle beziehbar.

Web-Anbindung und Anwendungsintegration

Zum Anschließen neuer und bestehender UTM-Anwendungen an das Web mit dem Produkt WebTransactions benötigen Sie die Handbücher zu **WebTransactions**.

Die Dokumentation wird durch JavaDocs ergänzt.

1.3.3 Readme-Dateien

Funktionelle Änderungen und Nachträge der aktuellen Produktversion zu diesem Handbuch entnehmen Sie bitte ggf. den Produkt-spezifischen Readme-Dateien. Readme-Dateien stehen Ihnen online bei dem jeweiligen Produkt zusätzlich zu den Produkthandbüchern unter *http://manuals.ts.fujitsu.com* zur Verfügung.

1.4 Darstellungsmittel

In diesem Handbuch werden folgende typographische Darstellungsmittel verwendet:

	Dieses Symbol weist auf wichtige Informationen hin, die unbedingt beachtet werden sollen.
Kursivschrift	kennzeichnet Kommandos, Variablen, Konstanten, Pfadnamen, Datei- und Programmnamen im Fließtext.
`dicktengleich`	kennzeichnet Meldungen und Bildschirmausgaben sowie Beispiele für Programme und Dateieinträge
Normalschrift halbfett	kennzeichnet in Syntaxdarstellungen Syntaxelemente, die so eingegeben werden müssen
Normalschrift	kennzeichnet in Syntaxdarstellungen variable Syntaxelemente. Sie sind Stellvertreter (Platzhalter) für andere Zeichen oder Zeichenketten, die an deren Stelle eingegeben werden müssen.
[ ]	Eckige Klammern dienen in Syntaxdarstellungen zur Kennzeichnung von optionalen Werten. Die Klammern sind nicht Element der Syntax und dürfen nicht miteingegeben werden. Im Fließtext verweisen Zahlen in eckigen Klammern auf Literaturhinweise am Ende des Buches.
{}	Geschweifte Klammern fassen in Syntaxdarstellungen Alternativen von Parameter oder Werten zusammen. Es darf nur ein Wert oder Parameter innerhalb der Klammern eingegeben werden.
|	Ein senkrechter Strich innerhalb von Syntaxdarstellungen trennt alternative Eingaben, von denen eine auszuwählen ist.
openCPIC-pfad	Bezeichnet das Installationsverzeichnis von OpenCPIC.

2 Einführung in das Produkt OpenCPIC

Dieses Kapitel beschreibt nach einer kurzen Einführung in das Thema der verteilten Transaktionsverarbeitung die Einbettung von OpenCPIC in das X/Open DTP Referenzmodell. Des Weiteren erhalten Sie einen Überblick über die Struktur von OpenCPIC und das Zusammenwirken der verschiedenen Komponenten.

2.1 Verteilte Verarbeitung mit globaler Transaktionssicherung

Eine Transaktion besteht aus einer Folge von Aktionen, die notwendig sind für die Ausführung einer fest umrissenen, anwendungsspezifischen Verarbeitungseinheit (unit of work). Zum Beispiel bildet eine Überweisung zwischen zwei Konten, bestehend aus einer Lastschrift von dem einen und einer Gutschrift auf das andere Konto eine Transaktion.

Transaktionen sind durch vier grundlegende Merkmale, auch als ACID-Kriterien (Atomicity, Consistency, Isolation, Durability) bekannt, gekennzeichnet:

– Atomicity
  Eine Transaktion wird entweder vollständig oder gar nicht ausgeführt.

– Consistency
  Beim Abbruch (Rollback) einer Transaktion werden alle beteiligten Ressourcen in ihren ursprünglichen Zustand zurückgeführt.

– Isolation
  Die Resultate der beteiligten Aktionen sind bis zum erfolgreichen Abschluss der Transaktion (Commit) nach außen nicht sichtbar.

– Durability
  Die Ergebnisse einer Transaktion sind dauerhaft.

Bei Beendigung einer Transaktion erfolgt entweder eine Bestätigung (Commit) oder das Zurücksetzen (Rollback) aller durchgeführten Aktionen. Ein Rollback ist nötig, falls mindestens eine der Aktionen nicht durchgeführt werden konnte oder zu einem Fehler führte.

Bei der verteilten Verarbeitung mit globaler Transaktionssicherung sind die an einer Transaktion beteiligten Programme und Ressourcen (z. B. Datenbanken) über verschiedene Systeme in einem Netz verteilt.

Bei globalen Transaktionen ist es notwendig, die Interaktionen zwischen den an der Transaktion beteiligten Programmen und verteilten Ressourcen zu koordinieren und abzustimmen. Diese Abstimmung erfolgt mit Hilfe eines sogenannten Zwei-Phasen-Commit-Protokolls, mit dem sichergestellt wird, dass alle getroffenen Entscheidungen konsistent sind.

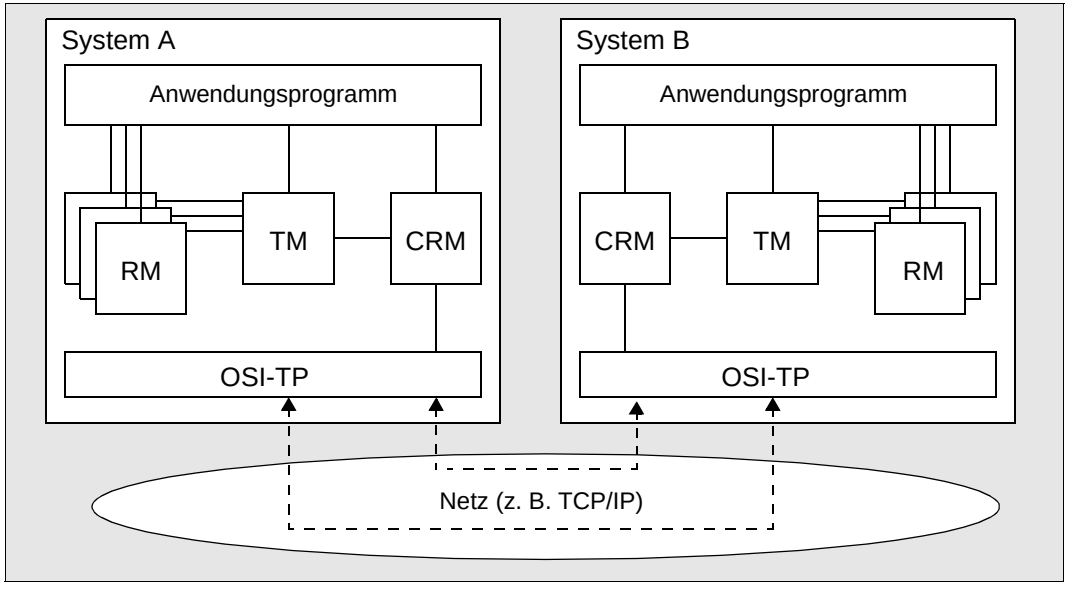
2.2 OpenCPIC und das X/Open DTP Referenzmodell

Durch die immer weiter fortschreitende Vernetzung von Computersystemen müssen zunehmend Software-Komponenten zusammenarbeiten, die von unterschiedlichen Herstellern entwickelt wurden und auf unterschiedlichen Rechnertypen mit unterschiedlichen Betriebssystemen ablaufen. Dies wird ermöglicht durch offene Systeme. Offene Systeme gewährleisten das Zusammenwirken und die Portierbarkeit von Anwendungen und Daten in heterogenen Systemumgebungen durch die Berücksichtigung internationaler Standards für Schnittstellen und Datenformate.

Für die verteilte Verarbeitung in offenen Systemen hat X/Open ein Referenzmodell entwickelt: das Distributed Transaction Processing Reference Model V2/11.93.

Das X/Open DTP Referenzmodell nutzt verschiedene standardisierte Schnittstellen für die Kommunikation zwischen Client- und Serveranwendungen sowie für die verteilte Verarbeitung.

Das folgende Bild zeigt die Bestandteile des DTP Referenzmodells.



X/Open DTP Referenzmodell

Anwendungsprogramm

Die beiden Anwendungsprogramme stellen die Endpunkte einer Programm-Programm-Kommunikation dar. Ein Anwendungsprogramm oder Transaktionsprogramm ist z. B. ein Server oder Client in einem verteilten Verarbeitungssystem.

CRM Die Kommunikation zwischen den Anwendungsprogrammen wird von den Communication Resource Managern (CRM) gesteuert. Die CRMs auf unterschiedlichen Systemen kommunizieren über das von ISO definierte Transaktionsprotokoll OSI-TP, dem unterschiedliche Netzprotokolle, wie z. B. TCP/IP oder X.25 zugrunde liegen können.
Als Schnittstelle zwischen Anwendungsprogramm und CRM wurden von X/Open CPI-C, XATMI und TxRPC definiert.

RM Während Anwendungsprogramme über CRMs mit entfernten Anwendungen kommunizieren, greifen Sie über lokale Resource Manager (RM) auf lokale Datenbestände zu. Die hierfür genutzten Schnittstellen sind durch den jeweiligen RM festgelegt. X/Open definiert verschiedene Schnittstellen zwischen Anwendungsprogramm und RM, z. B. SQL oder ISAM.

TM Der Transaction Manager (TM) überwacht die Durchführung von gesicherten Transaktionen. Er steuert Beginn und Abschluss einer Transaktion. Über das Zwei-Phasen-Commit-Protokoll erfolgt die Koordinierung mit lokalen RM und entfernten TM. Damit wird die Konsistenz der Datenmodifikationen aller beteiligten Resource Manager gewährleistet. Der TM führt außerdem Wiederanlaufmaßnahmen (Recovery) nach Fehlerereignissen durch.
X/Open definiert für die Kommunikation zwischen TM und RM die Schnittstelle XA und für die Kommunikation zwischen AP und TM die Schnittstelle TX (Transaction Demarcation Interface). Für die Kommunikation zwischen CRM und TM wurde die Schnittstelle XA+ definiert.

OSI-TP
Die CRM und TM auf unterschiedlichen Systemen kommunizieren über das von ISO definierte Transaktionsprotokoll OSI-TP (Open Systems Interconnection - Transaction Processing), dem unterschiedliche Netzprotokolle, wie z. B. TCP/IP oder X.25 zugrunde liegen können. OSI-TP ist das Protokoll für die oberste Schicht (Application Layer) des OSI-Protokoll-Stacks.
X/Open hat die Schnittstelle XAP-TP für den standardisierten Zugriff von Anwendungen auf OSI-TP definiert. XAP-TP stellt eine Erweiterung der X/Open Schnittstelle XAP dar, welche einen Zugriff auf die Schichten ACSE und Presentation Layer des OSI-Protokoll-Stacks ermöglicht.

Das Produkt OpenCPIC stellt eine Implementierung folgender Bestandteile des X/Open DTP Referenzmodells dar:

- Communication Resource Manager (CRM)
- Transaction Manager (TM)
- Schnittstelle CPI-C zwischen Anwendungsprogramm und CRM
- Schnittstelle XATMI zwischen Anwendungsprogramm und CRM
- Schnittstelle TX zwischen Anwendungsprogramm und TM
- Schnittstelle XAP-TP zwischen CRM und OSI-TP

Die Kommunikation zwischen CRM und TM erfolgt in OpenCPIC nicht über XA+, sondern über interne Schnittstellen.

Von zentraler Bedeutung sind im Folgenden die Begriffe Anwendung und Anwendungsprogramm:

Anwendung

Alle lokalen Anwendungsprogramme auf einem System bilden zusammen mit dem lokalen Communication Resource Manager (CRM) und dem lokalen Transaction Manger (TM) eine Anwendung. In der Terminologie von OSI (Open Systems Interconnection) sagt man statt Anwendung auch Application Entity (AE). Eine Anwendung im Netz wird über eine Transportadresse und einen Application Entity Title (AE Title) angesprochen. Sowohl die Transportadresse als auch der AE Title müssen netzweit eindeutig sein.
Bei OpenCPIC bildet der OpenCPIC-Manager auf einem System zusammen mit den lokalen Anwendungsprogrammen eine Anwendung. Typische Anwendungen sind neben OpenCPIC z. B. UTM-Anwendungen oder CICS-Anwendungen.

Anwendungsprogramm

Ein Anwendungsprogramm ist Teil einer Anwendung. Gleichbedeutend damit sind die Begriffe Transaktionsprogramm (TP, insbesondere bei CPI-C verwendet), Teilprogramm (bei openUTM) oder Transaction Processing Service User (TPSU, bei OSI-TP). In diesem Handbuch wird immer der Begriff Anwendungsprogramm verwendet.
Ein Anwendungsprogramm wird durch einen lokalen Namen identifiziert. Dieser Name ist nur innerhalb einer Anwendung eindeutig. Zur eindeutigen Adressierung eines Anwendungsprogramms im Netz ist daher auch die Transportadresse der Anwendung notwendig. Bei CPI-C verwendet man dafür auch den Begriff Transaction Program Name (TP Name), bei openUTM entspricht dies dem Transaktionscode, bei OSI-TP dem TPSU Title.
Anwendungsprogramme können, wie in OpenCPIC, eigenständige Prozesse sein, die sich mittels Interprozesskommunikation (IPC) mit dem lokalen CRM bzw. TM verständigen. Sie können aber auch, wie z. B. bei openUTM, mit CRM, TM und anderen Anwendungsprogrammen zu einer Anwendung (siehe oben) gebunden sein.

2.3 Einsatzgebiete von CPI-C, XATMI und TX

In der folgenden Tabelle ist dargestellt, auf welchen Gebieten Sie die Schnittstellen schwerpunktmäßig einsetzen können:

Schnittstelle	Einsatzgebiete
CPI-C	Ist eine Schnittstelle zur Programm-Programm-Kommunikation. Sie ist geeignet zum Anschluss von Präsentations-Clients (PC, Workstation) an UTM-Anwendungen und zur Kommunikation zwischen Server-Anwendungen. Da CPI-C auch im Rahmen von IBM SAA definiert ist, eignet sich CPI-C besonders für Anwendungen, die in IBM-Umgebung eingesetzt werden sollen.
XATMI	Ist eine Schnittstelle zur Programm-Programm-Kommunikation. Da XATMI auch von anderen Transaktionsmonitoren unterstützt wird, eignet sich XATMI besonders für Anwendungen, die mit Anwendungen anderer Transaktionsmonitore, wie z.B. TUXEDO, kommunizieren sollen.
TX	Ist eine Schnittstelle zwischen Programm und Transaktionsmonitor, die der Steuerung von globalen Transaktionen dient. Sie eignet sich besonders für die Kopplung von OpenCPIC-Anwendungen mit UTM-Anwendungen.

2.4 Komponenten von OpenCPIC

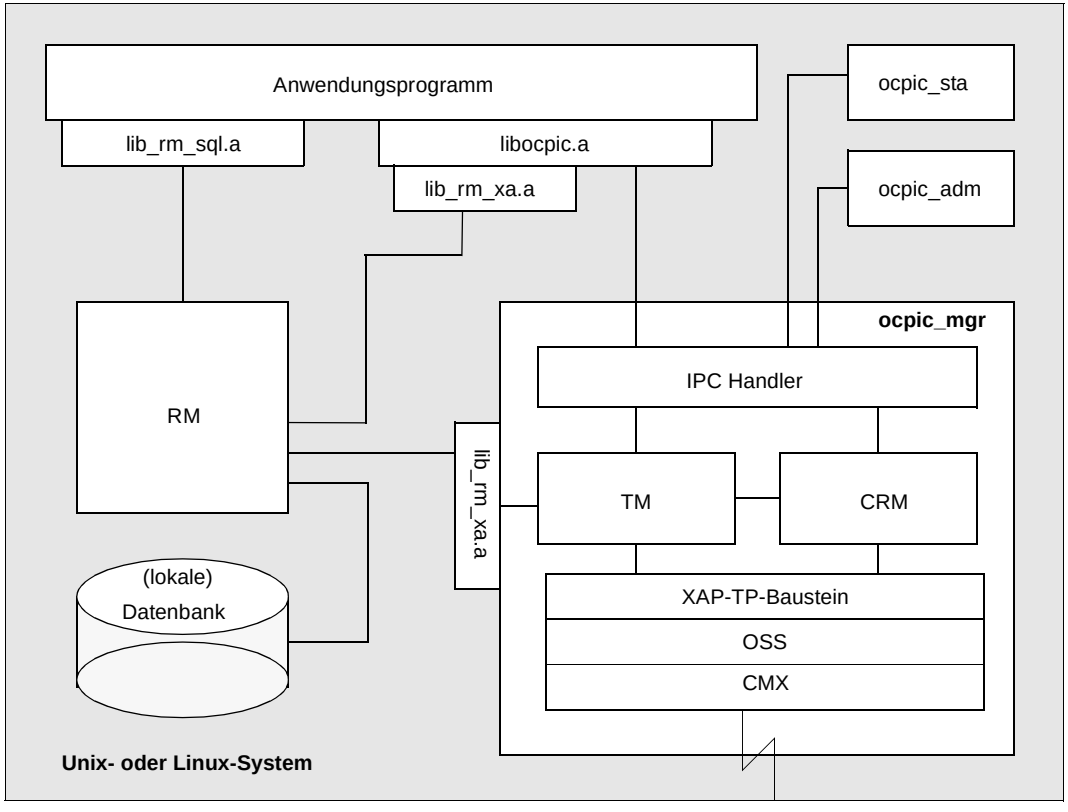
Das Produkt OpenCPIC besteht aus den folgenden Komponenten:

– Das Programm *ocpic_mgr*, im Folgenden als OpenCPIC-Manager bezeichnet.
Dieses Programm übernimmt als zentraler Managerprozess die Funktion des Communication Resource Manager (CRM) und des Transaction Manager (TM) für alle Anwendungsprogramme einer lokalen Anwendung.

– Die Bibliothek *libocpic.a*, in der die Schnittstellenfunktionen von CPI-C und TX implementiert sind.
libocpic.a ist eine statische Bibliothek, die zu jedem OpenCPIC-Anwendungsprogramm dazugebunden werden muss.

– Dienstprogramme z. B. für Generierung (*ocpic_gen*) und Administration (*ocpic_adm*, *ocpic_sta*)

Die folgende Abbildung zeigt eine einfache OpenCPIC-Anwendung in einem Unix- oder Linux-System, bestehend aus dem OpenCPIC-Manager (*ocpic_mgr*) und einem lokalen Anwendungsprogramm, welches über einen lokalen Resource Manager (RM) Zugriff auf eine lokale Datenbank hat. Wenn der Resource Manager dies unterstützt, ist auch der Zugriff auf eine verteilte Datenbank möglich. Das Bild veranschaulicht die Struktur einer OpenCPIC-Anwendung und das Zusammenwirken der einzelnen Komponenten von OpenCPIC auf einem lokalen System.



Komponenten einer OpenCPIC-Anwendung

Das Programm *ocpic_mgr* bildet den zentralen Managerprozess der Anwendung. In ihm sind die Komponenten CRM und TM des X/Open DTP Referenzmodells realisiert. Die Interprozesskommunikation des OpenCPIC-Managers mit den lokalen Anwendungsprogrammen sowie den Administrationsprogrammen (*ocpic_sta*, *ocpic_adm*) erfolgt über Named Pipes und wird im OpenCPIC-Manager durch die Komponente IPC-Handler gesteuert. Die CPI-C- und TX-Funktionen der Bibliothek *libocpic.a* senden und empfangen, ebenso wie die Administrationsprogramme, Nachrichten über spezielle Named Pipes. Der IPC-Handler leitet die Nachrichten an die zuständige Komponente (CRM oder TM) weiter und übernimmt das Zurücksenden der Antwortnachrichten.

Das Umsetzen der CPI-C- und TX-Funktionen auf das OSI-TP-Protokoll erfolgt durch eine Komponente namens XAP-TP-Baustein. Der XAP-TP-Baustein sendet die OSI-TP-Protokollelemente über das Netz an die Partneranwendung und reicht vom Partner empfangene Daten an den CRM bzw. TM weiter. Er setzt dazu auf dem Kommunikations-Produkt CMX auf.

Das Anwendungsprogramm in diesem Beispiel kommuniziert mit dem lokalen Resource Manager über eine SQL-Schnittstelle. Diese wird vom lokalen Datenbanksystem in Form einer Schnittstellenbibliothek mit dem Namen *lib_rm_sql.a* zur Verfügung gestellt.
In der Bibliothek *lib_rm_xa.a*, die ebenfalls Bestandteil des lokal installierten Datenbankproduktes ist, ist die XA-Schnittstelle realisiert. Da sowohl die TX-Funktionen der Bibliothek *libocpic.a*, als auch der Transaction Manager XA-Funktionen aufrufen, müssen beide Programme, also das Anwendungsprogramm und das Programm *ocpic_mgr*, mit dieser Bibliothek gebunden werden.

2.5 Wichtige Begriffe

Folgende Begriffe werden in diesem Handbuch häufig verwendet bzw. sind für das Verständnis erforderlich:

Association

Associations sind Verbindungen zwischen Application Entities, d.h. Verbindungen der Schicht 7 im ISO-Referenzmodell. Für jeden Dialog zwischen zwei Anwendungsprogrammen muss zunächst eine Association zwischen den beiden Partneranwendungen aufgebaut werden.

Dialog und Conversation

Als Dialog wird in OSI-TP eine logische Verbindung zwischen zwei TPSU (Transaction Processing Service User) bezeichnet.

Eine Conversation ist eine logische Verbindung zwischen zwei CPI-C-Anwendungsprogrammen. In OpenCPIC wird für jede Conversation zwischen zwei Anwendungsprogrammen je ein OSI-TP-Dialog durch die zugehörigen Partneranwendungen eröffnet. Das heißt, jeder Conversation ist genau ein Dialog zugeordnet und umgekehrt.

Superior und Subordinate

Als Superior oder Initiator einer Conversation wird derjenige Partner bezeichnet, der die Conversation mittels *Allocate (CMALLC)* eröffnet hat. Der Subordinate oder Akzeptor ist derjenige Partner, welcher die Conversation mittels *Accept (CMACCP)* oder *Accept_Incoming (CMACCI)* akzeptiert hat. Auf OSI-TP-Ebene ist dementsprechend der Superior eines Dialoges derjenige Partner, welcher den Dialog durch einen *TP-BEGIN-DIALOGUE-request* eröffnet hat und der Subordinate der Partner, welcher den Dialogaufbauwunsch in Form einer *TP-BEGIN-DIALOGUE-indication* zugestellt bekommt.

Transaktionsknoten, Root-, Intermediate- und Leaf-Knoten einer Transaktion

In OSI-TP sind verteilte Transaktionen in Form von Bäumen organisiert (siehe [32]). Die beteiligten TPSU (Transaction Processing Service User), d. h. die Anwendungsprogramme, die die Dialoge bzw. Conversations betreiben, bilden die Knoten des Transaktionsbaumes (Transaktionsknoten).

Der Wurzelknoten des Transaktionsbaumes wird als Root-Knoten bezeichnet. Er hat als einziger keinen Superior. Jeder Knoten kann höchstens einen Superior, aber mehrere Subodinates besitzen. Ein Knoten, der sowohl einen Superior als auch mindestens einen Subordinate besitzt, heißt Intermediate-Knoten. Knoten, die keinen Subordinate haben, heißen Leaf-Knoten der Transaktion.

Protected Conversations

Als Protected Conversations werden Conversations bezeichnet, die momentan an einer Transaktion beteiligt sind. Das bedeutet, der der Conversation zugrunde liegende Dialog ist Teil des Transaktionsbaums. In OSI-TP heißt ein solcher Dialog “Coordinated” (siehe [32]). Das Anwendungsprogramm, welches die Protected Conversation betreibt, stellt einen Knoten im Transaktionsbaum dar.

Precommit-Phase, Commit-Phase

Dies sind die beiden Phasen des Zwei-Phasen-Commit-Protokolls. Der Ablauf dieses Protokolls wird durch die beteiligten Transaction Manager gesteuert.

In der Precommit-Phase werden alle beteiligten Transaktionsknoten aufgefordert, ihre jeweiligen lokalen Ressourcen in einen Zustand zu bringen, der einen erfolgreichen Abschluss der Transaktion ermöglicht (“Prepare to Commit”). Dies erfolgt durch das Versenden eines entsprechenden CCR-Protokollelements, ausgehend vom Root-Knoten durch den gesamten Transaktionsbaum. Wenn alle Knoten ihre Bereitschaft zum Transaktionsabschluss (“Ready to Commit”) angezeigt haben, werden alle Knoten, wiederum ausgehend vom Root-Knoten, aufgefordert, die Modifikationen ihrer lokalen Ressourcen jetzt durchzuführen. Wenn in dieser Phase irgendein Knoten anzeigt, dass seine lokalen Ressourcen nicht zum Commit-Abschluss bereit sind, oder irgendein Fehlerereignis, z. B. Ausfall eines Transaktionsknotens, eintritt, werden alle Transaktionsknoten aufgefordert, einen Rollback durchzuführen, d. h. ihre lokalen Ressourcen wieder in den Ursprungszustand zu bringen.

In der Commit-Phase wird die in der Precommit-Phase getroffene Entscheidung (Commit oder Rollback) durchgeführt. Im Commit-Fall wird die Transaktion zu einem erfolgreichen Abschluss gebracht, alle Modifikationen der beteiligten Ressourcen sind danach nicht mehr rückgängig zu machen. Im Rollback-Fall werden alle beteiligten Ressourcen wieder in ihren Ursprungszustand gebracht.

Heuristische Entscheidungen

Unter bestimmten Umständen (z. B. Ausfall des Transaction Managers) können lokale Resource Manager während der Commit-Phase sogenannte heuristische Entscheidungen treffen.

Das bedeutet, ein lokaler Resource Manager entscheidet unabhängig vom Transaction Manager, den lokalen Transaktionszweig (transaction branch) durch Commit oder Rollback zu beenden. Wenn der Transaction Manager zu einem späteren Zeitpunkt dem lokalen Resource Manager seine Entscheidung mitteilt, meldet dieser dem Transaction Manager, dass er den lokalen Transaktionszweig bereits beendet hat. Dabei kann es zu Inkonsistenzen im Transaktionsbaum kommen, wenn die heuristische Entscheidung eines lokalen Resource Managers nicht mit der Entscheidung des Transaction Managers übereinstimmt.

Diese Situation wird als “Heuristic Mix Condition” oder “Heuristic Hazard Condition” bezeichnet. Man spricht dann auch von einem heuristischen Transaktionsende (“Heuristic Outcome”). Die Maßnahmen zur Behebung solcher Inkonsistenzen sind abhängig vom Resource Manager Produkt und i. a. durch den Systemadministrator durchzuführen.

3 OpenCPIC generieren

Der OpenCPIC-Manager benötigt zum Start bestimmte Konfigurierungsinformationen, die Sie als Systemverwalter in Form einer Konfigurationsdatei bereitstellen müssen. Das Erzeugen der Konfigurationsdatei erfolgt in zwei Schritten:

1. Erstellen einer sogenannten Generierungsdatei mit einem Texteditor

   In der Generierungsdatei geben Sie alle notwendigen Informationen in lesbarer Form (ASCII-Text) an. Der Aufbau der Generierungsdatei ist im nachfolgenden Abschnitt „Aufbau der Generierungsdatei“ auf Seite 30 beschrieben.

2. Erzeugen der binären Konfigurationsdatei *conffile* aus der Generierungsdatei

   Für den OpenCPIC-Manager müssen die Konfigurierungsinformationen in einem bestimmten internen Format vorliegen. Dazu wird die Generierungsdatei mit dem Kommando *ocpic_gen* aufbereitet. Dieses Kommando ist im Abschnitt „Generierungsprogramm starten“ auf Seite 50 beschrieben.

3.1 Aufbau der Generierungsdatei

Die Generierungsdatei besteht aus einer Folge von Anweisungen, wobei jede Anweisung nach folgendem allgemeinen Format aufgebaut ist:

Anweisungstyp Name [,Parameter$_1$[,..., Parameter$_n$]]

Anweisungstyp
: gibt den Typ der Anweisung an. Folgende Anweisungstypen sind möglich:

  – LOCAPPL (siehe Seite 32)
  – PARTAPPL (siehe Seite 34)
  – LOCAPRO (siehe Seite 39)
  – SYMDEST (siehe Seite 43)
  – APPLICATION-CONTEXT (siehe Seite 48)

Name
: ist der Name des Objekts, das mit dieser Anweisung generiert werden soll. Der Name muss in derselben Zeile wie Anweisungstyp stehen. Er ist frei wählbar und darf nur einmal innerhalb der Generierungsdatei vorkommen.

Parameter
: besteht aus einem Schlüsselwort, dem Zeichen ‘=’ und einem Wert.

  Beispiel: **ASSOCIATIONS = 2**

  Leere Wertzuweisungen (z. B. **ASSOCIATIONS** =) sind nicht erlaubt.

  Mehrere zu einer Anweisung gehörenden Parameter müssen durch Kommas getrennt werden. Dem letzten Parameter einer Anweisung darf kein Komma folgen.

  Die für jede Anweisung zulässigen Parameter und ihr Wertebereich sind im jeweiligen Unterabschnitt beschrieben. Parameter in eckigen Klammern sind optional. Für fehlende optionale Parameter wird der Standardwert angenommen.

Es gelten folgende Regeln:

- Eine Generierungsdatei muss genau eine LOCAPPL-Anweisung und mindestens eine PARTAPPL-Anweisung enthalten. Die übrigen Anweisungen sind optional.
- Die Reihenfolge der Anweisungen in der Generierungsdatei sowie die Reihenfolge der Parameter innerhalb einer Anweisung sind beliebig.
- Jede Anweisung muss auf einer neuen Zeile beginnen und kann sich über mehrere Zeilen erstrecken. Allerdings muss ein Parameter immer innerhalb einer Zeile stehen.
- Zeilen, die mit einem Stern (*) oder Nummernkreuz (#) beginnen, werden als Kommentar behandelt und von *ocpic_gen* überlesen.

Allgemeine Syntaxregeln für Namen

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

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

Allgemeine Syntaxregeln für Objektbezeichner

Ein Objektbezeichner (Object Identifier) ist ein spezieller, in ISO 8824 [29] definierter ASN.1-Datentyp. Objektbezeichner bestehen i. a. aus einer Liste von Elementen, die in Form von ganzen Zahlen oder Namen angegeben werden können.

In OpenCPIC muss ein Objektbezeichner immer in Form einer Liste von mindestens zwei bis maximal zehn nichtnegativen ganzen Zahlen angegeben werden, die jeweils durch Kommas getrennt sind. Die Liste wird durch runde Klammern () begrenzt.

Beispiel: **(1, 5 ,6）**

Beachten Sie, dass die Anzahl der Elemente in der Liste relevant ist. So sind z. B. (1, 5, 6) und (1, 5, 6, 0) zwei unterschiedliche Objektbezeichner.

Für die Elemente eines Objektbezeichners gelten folgende Wertebereiche:

1. Element:	0 - 2 (siehe Tabelle auf Seite 31)
2. Element:	falls das erste Element 0 oder 1 ist: 0 - 39, andernfalls 0 - 67108863
3. bis 10. Element:	0 - 67108863

Objektbezeichner müssen von einer autorisierten Behörde registriert werden. Das erste Element eines Objektbezeichners identifiziert die Registrierungsbehörde, welche für die Registrierung des Objektbezeichners (d. h. der nachfolgenden Elemente) autorisiert ist. Dabei gelten die folgenden Zuordnungen (siehe auch ISO 8824 [29], Annex B):

Wert des ersten Elements	**Autorisierte Registrierungsbehörde**
0 (ccitt)	CCITT
1 (iso)	ISO
2 (joint-iso-ccitt)	CCITT und ISO

3.1.1 LOCAPPL - Lokale Anwendung definieren

Syntax

LOCAPPL	locappl_name**,**
APT	**=** apt_objid**,**
AEQ	**=** aeq_num

Beschreibung

Mit dieser Anweisung definieren Sie Ihre lokale Anwendung, d. h. den OpenCPIC-Manager auf dem lokalen System, als lokalen Zugriffspunkt für die Kommunikation über das OSI-TP-Protokoll. Alle CPI-C-Conversations, die von Ihren lokalen Anwendungsprogrammen betrieben werden sollen, werden über diesen einen Zugriffspunkt gesteuert. In einer Generierungsdatei muss es daher genau eine LOCAPPL-Anweisung geben.

locappl_name

ist der Name, unter dem sich der OpenCPIC-Manager bei CMX anmelden soll. Er muss im TNSX (Transport Name Service im Unix- oder Linux-System) als *Globaler Name* generiert sein. Wie Sie Einträge im TNSX vornehmen, ist im CMX-Benutzerhandbuch [22] beschrieben.

Der *locappl_name* besteht aus fünf Namensteilen, jeweils durch Punkte getrennt:

NP5.NP4.NP3.NP2.NP1

Sind Namensteile leer, können die Punkte am Namensende entfallen. Die Maximallängen der einzelnen Namensteile sind wie folgt:

NP1	2	Zeichen
NP2	16	Zeichen
NP3	16	Zeichen
NP4	10	Zeichen
NP5	30	Zeichen

Genaueres zur Syntax des *Globalen Namens* finden Sie im CMX-Benutzerhandbuch [23] (Beschreibung des Programmaufrufs *t_getaddr*).

Hinweis: Im TNSX müssen neben einem T-Selektor für die Schicht 4 auch ein P- und ein S-Selektor für die Schichten 5 und 6 eingetragen sein. Es empfiehlt sich, Leereinträge für S- und P-Selektoren zu wählen. (siehe dazu CMX-Benutzerhandbuch [22]).

apt_objid
: gibt den Application Process Title (APT) in Form eines Objektbezeichners an (siehe Seite 31). Der Application Process Title bildet zusammen mit dem Application Entity Qualifier (AEQ) den Application Entity Title (AET) der lokalen Anwendung.
Wie auf Seite 31 beschrieben, müssen Objektbezeichner von einer autorisierten Behörde registriert werden. Dies gilt auch für den Application Process Title, insbesondere dann, wenn Sie in einem offenen Netz arbeiten.
In einem privaten, abgeschlossenen Netz ist der Netzbetreiber selbst für die Eindeutigkeit der Application Process Title zuständig, weshalb in diesem Fall auf eine offizielle Registrierung verzichtet werden kann.

aeq_num
: ist eine ganze Zahl zwischen 1 und 67108863 und gibt den Application Entity Qualifier (AEQ) an. Er ist frei wählbar und bildet zusammen mit dem Application Process Title (APT) den Application Entity Title (AET) der lokalen Anwendung.

Anmerkungen

1. Es müssen sowohl APT = *apt_name* als auch AEQ = *aeq_name* angegeben sein.
2. Mit dem aus APT und AEQ gebildeten Application Entity Title (AET) muss der lokale OpenCPIC-Manager im Netz eindeutig adressierbar sein. Der AET wird als Parameter im OSI-TP-Protokollelement *TP-BEGIN-DIALOGUE-request* mitgegeben.

3.1.2 PARTAPPL - Partneranwendung definieren

Syntax

PARTAPPL	partappl_name**,**
APT	**=** apt_objid**,**
AEQ	**=** aeq_num [**,**
APPL-CONTEXT	**=** context_name][**,**
CCR	**=** ccr_option][**,**
ASSOCIATIONS	**=** associations_number][**,**
CONNECT	**=** connect_number][**,**
CONTWIN	**=** (contwin_min contwin_max)][**,**
IDLETIME	**=** idle_number]

Beschreibung

Mit dieser Anweisung generieren Sie eine Partneranwendung, z. B. einen OpenCPIC-Manager oder eine UTM-Anwendung auf einem entfernten System. Jede Partneranwendung, die im Netz erreicht werden soll, muss mit einer separaten PARTAPPL-Anweisung generiert werden. Zu nicht generierten Partneranwendungen kann keine Verbindung aufgebaut werden. Die PARTAPPL-Anweisung ist somit auch ein Hilfsmittel für den Datenschutz.

partappl_name
: ist der Name der Partneranwendung im entfernten System. Es gelten dieselben Syntaxregeln wie für den *locappl_name* in der LOCAPPL-Anweisung (siehe Seite 32).

  Der *partappl_name* muss im TNSX als *Globaler Name* generiert sein. Wie Sie Einträge im TNSX vornehmen, ist im CMX-Benutzerhandbuch [22] beschrieben.

  Hinweis: Im TNSX müssen neben einem T-Selektor für die Schicht 4 auch ein P- und ein S-Selektor für die Schichten 5 und 6 eingetragen sein. Es empfiehlt sich, Leereinträge für S- und P-Selektoren zu wählen. (siehe dazu CMX-Benutzerhandbuch [22]).

apt_objid

gibt den Application Process Title (APT) in Form eines Objektbezeichners (siehe Seite 31) an. Der Application Process Title bildet zusammen mit dem Application Entity Qualifier (AEQ) den Application Entity Title (AET) der Partneranwendung. Wie auf Seite 31 beschrieben, müssen Objektbezeichner von einer autorisierten Behörde registriert werden. Dies gilt auch für den Application Process Title, insbesondere dann, wenn Sie in einem offenen Netz arbeiten.
In einem privaten, abgeschlossenen Netz ist der Netzbetreiber selbst für die Eindeutigkeit der Application Process Title zuständig, weshalb in diesem Fall auf eine offizielle Registrierung verzichtet werden kann.

aeq_num

ist eine ganze Zahl zwischen 1 und 67108863 und gibt den Application Entity Qualifier (AEQ) an. Er bildet zusammen mit dem Application Process Title (APT) den Application Entity Title (AET) der Partneranwendung.

context_name

ist der Name des Application Context der für die Verbindung zur Partneranwendung vereinbart wird. Sie können hier entweder einen der nachfolgend beschriebenen, vordefinierten Application Context Namen angeben oder den Namen eines in der Generierungsdatei von Ihnen definierten Application Context (siehe Seite 48, APPLICATION-CONTEXT-Anweisung). Im Falle eines selbst definierten Application Context, muss die APPLICATION-CONTEXT-Anweisung vor der PARTAPPL-Anweisung in der Generierungsdatei stehen.

Folgende Application Context Namen sind vordefiniert:

= def-cont Standard Application Context, bestehend aus den Application Service Elementen ACSE, TP-ASE, CCR (optional, siehe Parameter CCR = *ccr_option*), und UDT. UDT (mit Octet String Mapping) ist die abstrakte Syntax, als Transfersyntax wird BER (Basic Encoding Rules) verwendet (Standardwert).

= utm-secu Dieser Application Context muss verwendet werden, wenn der Partner eine UTM-Anwendung ist und die CPI-C-Aufrufe *Set_Conversation_Security_Type (CMSCST)*, *Set_Conversation_Security_User_ID (CMSCSU)* und *Set_Conversation_Security_Password (CMSCSP)* verwendet werden sollen (siehe Anmerkung 2 auf Seite 38).

= xatmi Dieser Application Context muss verwendet werden, wenn beide Partnerprogramme die X/Open-Schnittstelle XATMI benutzen.

Die folgende Übersicht gibt neben den Objektbezeichnern der vordefinierten Application Context Namen an, welche abstrakten Syntaxen jeweils enthalten sind. Als Transfersyntax wird immer BER (Basic Encoding Rules) verwendet.

Application Context Name (Schlüsselwort)	**Objektbezeichner**	**Enthaltene abstrakte Syntaxen**
def-cont	(1, 0, 10026, 6, 2)	UDT, ggf. CCR
utm-secu	(1, 3, 12, 2, 1107, 1, 6, 1, 3, 0)	UDT, UTM-Security, CCR
xatmi	(1, 2, 826, 0, 1050, 4, 2, 1)	XATMI, ggf. CCR

Die folgende Tabelle enthält die Objektbezeichner der benutzten abstrakten und Transfersyntaxen.

Syntax Name	**Objektbezeichner**
BER	(2, 1, 1)
CCR	(2, 7, 1, 1, 1)
UDT	(1, 0, 10026, 6, 1, 1)
UTM-Security	(1, 3, 12, 2, 1107, 1, 6, 1, 2, 0)
XATMI	(1, 2, 826, 0, 1050, 4, 1, 0)

Weitere Informationen finden Sie im Abschnitt „APPLICATION-CONTEXT - Application Context definieren“ auf Seite 48.

ccr_option

gibt an, ob die Partneranwendung die abstrakte Syntax CCR (Commitment, Concurrency and Recovery) unterstützt, d. h., ob der Application Context der Partneranwendung die CCR-Syntax enthält. Die CCR-Syntax ist notwendig für das Betreiben von Conversations mit dem *sync_level CM_SYNC_POINT* oder *CM_SYNC_POINT_NO_CONFIRM*. Dieser Parameter ist optional und kann die folgenden Werte annehmen:

= **YES** Der verwendete Application Context enthält die CCR-Syntax (Standardwert).

= **NO** Der verwendete Application Context enthält die CCR-Syntax nicht.

Für den Application Context Namen *utm-secu* wird dieser Parameter nicht ausgewertet. Im Application Context *utm-secu* ist die CCR-Syntax standardmäßig enthalten.

associations_number

ist die maximale Anzahl paralleler Associations zwischen der lokalen Anwendung und der Partneranwendung. Zulässige Werte für *associations_number* sind ganze Zahlen von 1 bis 256. Der Standardwert ist 1.

Associations sind Schicht-7-Verbindungen im ISO-Referenzmodell. Diese werden derzeit eins zu eins auf Transportverbindungen abgebildet. Deshalb werden Transportsysteme empfohlen, die parallele Transportverbindungen unterstützen, z. B. TCP/IP oder OSI-Protokolle. Es ist wirkungslos, für *associations_number* einen größeren Wert anzugeben als die maximale Anzahl von parallelen Transportverbindungen, die das Transportsystem unterstützt.

Für jeden Dialog zwischen zwei Anwendungsprogrammen (und damit für jede CPI-C- Conversation) muss eine Association aufgebaut werden. Deshalb ist *associations_number* gleichzeitig eine Obergrenze für die Anzahl der Dialoge bzw. Conversations, die gleichzeitig zwischen den Anwendungsprogrammen der lokalen und der Partneranwendung aktiv sein können.

connect_number

ist die Anzahl der Associations zur Partneranwendung, die beim Start des lokalen OpenCPIC-Managers automatisch aufgebaut werden sollen. Zulässige Werte sind ganze Zahlen von 0 bis *associations_number*. Der Standardwert ist 0.

Kann der OpenCPIC-Manager die gewünschte Anzahl von Associations beim Start nicht aufbauen, erfolgt alle fünf Minuten ein neuer Versuch.

contwin_min, contwin_max

sind die minimale bzw. maximale Anzahl von Associations, für die der lokale OpenCPIC-Manager der Contention Winner ist. Zulässige Werte sind ganze Zahlen von 0 bis *associations_number*. Dabei muss *contwin_min* kleiner gleich *contwin_max* sein. Der Standardwert für *contwin_min* ist 0, der Standardwert für *contwin_max* ist *associations_number*.

Der Contention Winner ist zuständig für die Verwaltung der Association. Unabhängig davon, welche Seite für eine Association der Contention Winner ist, können beide Partneranwendungen auf dieser Association einen Dialog bzw. eine Conversation beginnen. Kommt es zu einer Kollision, d. h. beide Partner versuchen gleichzeitig, auf der Association einen Dialog zu beginnen, wird die Association vom Contention Winner belegt, während der Dialogaufbau auf der Seite des Contention Loser abgelehnt wird.

Es wird empfohlen, nur dann Contention-Winner-Associations zu konfigurieren, wenn tatsächlich von der lokalen Seite aus ein Dialogaufbau erfolgen soll (d. h. Ihre lokalen CPI-C-Anwendungsprogramme bauen mit *Allocate (CMALLC)* aktiv Conversations auf). Erfolgt auf der lokalen Seite kein Dialogaufbau (d. h. Ihre Anwen-

dungsprogramme nehmen nur Dialogaufbauwünsche von der Partnerseite mittels *Accept_Conversation(CMACCP)* bzw. *Accept_Incoming (CMACCI)* an), dann sollten Sie *contwin_min* und *contwin_max* auf 0 setzen.

Sollen auf der lokalen Seite sowohl aktiv Dialoge aufgebaut als auch Dialogwünsche vom Partner akzeptiert werden, so wird empfohlen, folgendermaßen vorzugehen: Setzen Sie *contwin_min* auf die Anzahl der (gleichzeitig aktiven) Dialoge, die von der lokalen Anwendung aufgebaut werden sollen und *contwin_max* auf einen Wert, der kleiner ist als *associations_number*, so dass die Differenz L = *associations_number* - *contwin_max* größer oder gleich der Anzahl der (gleichzeitig aktiven) Dialoge ist, die von der Partnerseite aus aufgebaut werden. Da L größer oder gleich der Anzahl der Contention-Loser-Associations ist, ist damit sichergestellt, dass alle ankommenden Dialogwünsche von der lokalen Anwendung akzeptiert werden.

Hinweis: Ist der Partner eine UTM-Anwendung, müssen *contwin_min* und *contwin_max* mit der UTM-Generierung auf der Partnerseite abgestimmt werden.

idle_number

ist die Überwachungszeit (in Sekunden) für den Idle-Zustand der Associations zur Partneranwendung. *idle_number* wird in Form einer ganzen Zahl von 0 bis 32767 angegeben. Der Standardwert ist 0.

Wird eine Association in der angegebenen Zeit (d. h. länger als *idle_time* Sekunden) nicht durch einen Dialog belegt, baut der OpenCPIC-Manager die Association ab. Ein Wert von *idle_number = 0* bedeutet, dass der Idle-Zustand der Associations nicht überwacht wird.

Anmerkungen

1. Es müssen sowohl APT = *apt_name* als auch AEQ = *aeq_name* angegeben sein.
2. Mit dem aus APT und AEQ gebildeten Application Entity Title (AET) muss die Partneranwendung im Netz eindeutig adressierbar sein. Der AET wird als Parameter im OSI-TP-Protokollelement *TP-BEGIN-DIALOGUE-request* mitgegeben.
3. Soll der *partappl_name* als Argument für den CPI-C-Aufruf *Set_Partner_LU_Name (CMSPLN)* verwendet oder mit *Extract_Partner_LU_Name (CMEPLN)* abgefragt werden (siehe Abschnitt “Die SYMDEST-Anweisung”, Anmerkung 1, auf Seite 45), so darf er aufgrund der Definition der CPI-C-Schnittstelle insgesamt nicht länger als 17 Zeichen sein. *CMSPLN* liefert andernfalls den Returnwert *CM_PROGRAM_PARAMETER_CHECK*, während *CMEPLN* den Namen nach 17 Zeichen abschneidet.

3.1.3 LOCAPRO - Lokales OpenCPIC-Anwendungsprogramm definieren

Syntax

LOCAPRO	locapro_name**,**
PATH	**=** path_name**,**
LOGIN	**=** login_name

Beschreibung

Mit dieser Anweisung definieren Sie ein lokales OpenCPIC-Anwendungsprogramm (**Loc**al **A**pplication **Pro**gram).

Jedes lokale Anwendungsprogramm (AP), das vom lokalen OpenCPIC-Manager aufgrund eines von der Partneranwendung empfangenen Dialogwunsches automatisch gestartet werden soll, muss mit einer LOCAPRO-Anweisung in der Generierungsdatei definiert sein.

Lokale Anwendungsprogramme, die nicht vom OpenCPIC-Manager gestartet werden sollen, müssen nicht konfiguriert werden. Ein solches AP muss sich durch den CPI-C-Aufruf *Specify_Local_TP_Name (CMSLTP)* beim OpenCPIC-Manager mit seinem Namen anmelden, bevor es einen ankommenden Dialogwunsch akzeptieren kann.

locapro_name

ist der lokale Name des AP, wie er von der Partneranwendung beim Dialogaufbau (als Parameter im OSI-TP-Protokollelement *TP-BEGIN-DIALOGUE-request*) mitgegeben wird.

Der *locapro_name* ist normalerweise nicht der Name einer ausführbaren Datei im Unix- oder Linux-System, sondern stellt einen symbolischen Namen dar, der über den Parameter PATH = *path_name* (siehe unten) mit einem Programmnamen verbunden wird.

In der OSI-TP-Terminologie entspricht dieser symbolische Name dem TPSU Title, in CPI-C dem TP Name.

Der *locapro_name* kann bis zu 64 Zeichen lang sein. Es sind nur Zeichen aus dem Zeichensatz *T61String* (siehe “Abschnitt „Zeichensätze“ auf Seite 197”) zulässig.

Hinweis: Wenn das lokale AP von einer UTM-Partneranwendung angesprochen werden soll, darf der *locapro_name* nur maximal acht Zeichen lang sein und nur aus Großbuchstaben und Ziffern bestehen.

path_name
: ist der (vollqualifizierte) Unix- oder Linux-Pfadname des lokalen Anwendungsprogramms, das bei Empfang eines Dialogwunsches für den symbolischen Namen *locapro_name* (siehe oben) gestartet werden soll. Der *path_name* kann bis zu 64 Zeichen lang sein und muss mit '/' beginnen.

login_name
: ist die Unix- oder Linux-Benutzerkennung, unter der das lokale Anwendungsprogramm gestartet werden soll. Das lokale AP läuft nach dem Start durch den lokalen OpenCPIC-Manager in der Umgebung und mit den Zugriffsrechten dieser Benutzerkennung. Der *login_name* ist maximal acht Zeichen lang.

Anmerkungen

1. Der lokale Name entspricht auf der Partnerseite dem *TP_Name* in der Side Information bzw. dem vom Partner-AP mittels *Set_TP_Name (CMSTPN)* gesetzten Namen. Ist der Partner eine UTM-Anwendung, entspricht der lokale Name dem Wert des Parameters RTAC in der LTAC-Anweisung der UTM-Generierung (KDCDEF). Eine Beispiel-Konfigurierung finden Sie im Kapitel „Kommunikation mit Partneranwendungen“ auf Seite 121ff.).
2. Jedes lokale Anwendungsprogramm, das einen Dialogaufbauwunsch von einem Partner-AP mittels *Accept_Conversation* (*CMACCP)* oder *Accept_Incoming* (*CMACCI)* akzeptieren will, muss beim OpenCPIC-Manager mit mindestens einem lokalen Namen angemeldet sein. Die Anmeldung erfolgt entweder beim automatischen Programmstart durch den OpenCPIC-Manager, wobei der lokale Name mit der hier beschriebenen LOCAPRO-Anweisung generiert sein muss, oder bei laufendem Programm mit dem CPI-C-Aufruf *Specify_Local_TP_Name (CMSLTP)*.
   Programme, die zwar in einer LOCAPRO-Anweisung generiert sind, jedoch nicht durch den OpenCPIC-Manager, sondern durch ein Operator-Kommando gestartet werden, sind nicht automatisch mit dem in der LOCAPRO-Anweisung generierten lokalen Namen angemeldet. Jedes durch ein Operator-Kommando gestartete AP muss sich mittels *CMSLTP* selbst anmelden.
3. Wenn der lokale OpenCPIC-Manager von einer Partneranwendung einen Dialogaufbauwunsch für einen bestimmten lokalen Namen erhält, wird die Gültigkeit dieses Namens in folgenden Schritten geprüft:
   – Gibt es bereits mindestens ein aktives AP, das mit dem betreffenden lokalen Namen beim OpenCPIC-Manager angemeldet ist, so wird der Dialogwunsch so lange in eine Warteschleife gestellt, bis eines dieser APs mit *Accept_Conversation (CMACCP)* oder *Accept_Incoming (CMACCI)* den Dialog akzeptiert. Wenn zum Zeitpunkt des Eintreffens des Dialogwunsches vom Partner bereits ein AP auf die Beantwortung eines offenen *CMACCP/CMACCI*-Aufrufs wartet, wird der Dialogwunsch unmittelbar an das lokale AP weitergeleitet und die Conversation auf der lokalen Seite einge-

richtet. Falls mehrere APs mit demselben lokalen Namen einen Dialogwunsch akzeptieren wollen, muss die Reihenfolge, in der ankommende Dialogwünsche weitergereicht werden, nicht mit der Reihenfolge der *CMACCP/CMACCI*-Aufrufe übereinstimmen.

Falls Sie bisher Anwender von OpenCPIC V2.0 waren, beachten Sie bitte, dass hier ein Unterschied in der Behandlung von ankommenden Dialogwünschen besteht. Da diese immer in die Warteschlange gestellt werden, sobald mindestens ein Anwendungsprogramm mit dem angegebenen lokalen Namen aktiv ist, ist es nicht ohne Weiteres möglich, mehrere Anwendungsprogramme mit demselben Namen gleichzeitig automatisch starten zu lassen. Falls Sie einen mehrfachen automatischen Start ein und desselben Anwendungsprogramms wünschen, können Sie dies auf folgende zwei Arten erreichen:

a) Sie definieren mehrere verschiedene lokale Namen für das Anwendungsprogramm (jedes lokale Anwendungsprogramm kann beliebig viele lokale Namen haben).

b) Sie erklären den lokalen Namen nach erfolgreichem *Accept_Conversation (CMACCP)* bzw. *Accept_Incoming (CMACCI)* mittels *Release_Local_TP_Name (CMRLTP)* für ungültig (siehe dazu auch Anmerkung 4).

– Wenn sich noch kein laufendes AP mit diesem Namen beim OpenCPIC-Manager angemeldet hat, jedoch eine entsprechende LOCAPRO-Anweisung in der aktuellen Konfigurierung existiert, so versucht der OpenCPIC-Manager das betreffende Programm (Parameter PATH = *path_name*) im lokalen System zu starten. Der Dialogwunsch wird in eine Warteschlange gestellt, bis ein lokales AP einen Dialog für den betreffenden lokalen Namen mit *CMACCP/CMACCI* akzeptiert. (Dies muss nicht unbedingt das vom OpenCPIC-Manager gestartete Programm sein).

– Wenn sich bei Empfang eines Dialogaufbauwunsches kein lokales AP mit dem gewünschten Namen angemeldet hat und kein lokales AP diesen Namens mit einer LOCAPRO-Anweisung generiert wurde, lehnt der lokale OpenCPIC-Manager den Dialogwunsch ab, was auf der Partnerseite zu einem Dialogabbruch führt. Dies erzeugt z. B. bei einem CPI-C-Partnerprogramm den Returnwert *CM_DEALLOCATED_ABEND*.

– Eine Ablehnung des Dialogwunsches erfolgt auch, wenn der Start des lokalen AP fehlschlägt oder wenn durch Abbruch eines lokalen AP der gewünschte lokale Name ungültig wird und sich kein anderes, mit diesem Namen generiertes AP, starten lässt.

4. CPI-C ermöglicht Anwendungsprogrammen, die sich durch den Aufruf *Specify_Local_TP_Name (CMSLTP)* beim lokalen System unter einem speziellen lokalen Namen angemeldet haben, sich unter diesem Namen auch wieder abzumelden. Dies erfolgt durch den Aufruf *Release_Local_TP_Name (CMRLTP)*. Ein mit der LOCAPRO-Anweisung generierter lokaler Name kann auf diesem Wege nur bedingt annulliert werden. Das Anwendungsprogramm, welches sich unter diesem Namen mit *CMRLTP* beim OpenCPIC-Manager abmeldet, kann bis zu seiner Beendigung (bzw. bis der lokale Name durch einen entsprechenden *CMSLTP*-Aufruf wieder gültig wird), keine Dialogaufbauwünsche für diesen lokalen Namen mehr annehmen. Der *locapro_name* in der Konfigurationsdatei behält jedoch seine Gültigkeit. Das heißt, wenn der OpenCPIC-Manager einen weiteren Dialogaufbauwunsch für diesen lokalen Namen erhält und kein laufendes AP mit diesem Namen angemeldet ist, wird wieder das mit der LOCAPRO-Anweisung generierte Programm in der unter 3. beschriebenen Weise gestartet.

3.1.4 SYMDEST - Kommunikationspartner definieren

Syntax

SYMDEST	symdest_name,
PARTNER-APPL	**=** partappl_name**,**
PARTNER-APRO	**=** partner_name [**,**
PARTNER-TYPE	**=** partner_type][**,**
SEC-TYPE	**=** security_type][**,**
SEC-UID	**=** security_userid][**,**
SEC-PASS	**=** security_passwd]

Beschreibung

Mit dieser Anweisung definieren Sie einen Kommunikationspartner bei einer Partneranwendung. Dies kann z. B. ein CPI-C-Anwendungsprogramm sein, falls das Trägersystem des Partners OpenCPIC ist, oder ein UTM-Transaktionscode, falls der Partner eine UTM-Anwendung ist.

Jeder Partner wird in einem eigenen Eintrag in der CPI-C Side Information beschrieben. In CPI-C wird ein solcher Eintrag über einen sogenannten Symbolic Destination Name referenziert.

symdest_name

ist der Symbolic Destination Name für den Partner. Dieser Name referenziert den Eintrag in der Side Information und wird als Kurzname zur Adressierung des Partners im CPI-C-Aufruf *Initialize_Conversation (CMINIT)* mitgegeben (Parameter *symdest_name*).

Der *symdest_name* besteht aus bis zu acht Zeichen aus dem CPI-C Zeichensatz 01134 (siehe “Abschnitt „Zeichensätze“ auf Seite 197”), d. h. es dürfen nur Großbuchstaben und Ziffern verwendet werden. Ein *symdest_name*, der kürzer als acht Zeichen ist, wird mit Leerzeichen (Blanks) aufgefüllt.

Zusätzlich ist ".DEFAULT" als *symdest_name* zulässig (siehe Anmerkung 4).

partappl_name
: ist der Name der Partneranwendung, über die das Partnerprogramm angesprochen wird. Der *partappl_name* muss in einer PARTAPPL-Anweisung in der Generierungsdatei definiert sein.

partner_name
: ist der symbolische Name des Partnerprogramms, wie er auf der Partnerseite als lokaler Name definiert ist. Der *partner_name* wird beim Dialogaufbau als Parameter im OSI-TP-Protokollelement *TP-BEGIN-DIALOGUE-request* mitgegeben.

  In der OSI-TP-Terminologie entspricht dieser symbolische Name dem TPSU Title.

  Der *partner_name* kann bis zu 64 Zeichen lang sein. Die Menge der zulässigen Zeichen ist abhängig vom Wert des Parameters PARTNER-TYPE = *partner_type* (siehe unten).

  Bitte beachten Sie: Wenn der Partner eine UTM-Anwendung ist, darf der *partner_name* nur maximal acht Zeichen lang sein und nur aus Großbuchstaben und Ziffern bestehen.

partner_type
: gibt den ASN.1-Datentyp (siehe ISO 8824 [29]) an, mit dem der *partner_name* gebildet ist. Bei der Kodierung des *partner_name* wird dieser Typ mitgegeben. Zulässige Werte sind die folgenden Schlüsselwörter:

  = **PRINTABLE-STRING**
  : ASN.1-Character-String-Type *PrintableString*
    Die zulässigen Zeichen für diesen Datentyp sind im Abschnitt „Zeichensätze“ auf Seite 197 aufgelistet.

  = **T61-STRING**
  : ASN.1-Character-String-Type *T61String* (Standardwert)
    Die zulässigen Zeichen für diesen Datentyp sind im Abschnitt „Zeichensätze“ auf Seite 197 aufgelistet. (Standardwert)

  = **INTEGER** ASN.1-Type *Integer*

security_type
: gibt an, welche Zugangsschutzinformationen (Security-Daten) beim aktiven Dialogaufbau zur Partneranwendung mitgeschickt werden (siehe dazu auch Anmerkung 3 auf Seite 47).

  Sie können folgende Schlüsselwörter angeben:

  = **NONE** Es werden keine Zugangsschutzinformationen mitgeschickt (Standardwert).

= **SAME** Die Benutzerkennung, unter welcher das lokale Anwendungsprogramm gestartet wurde, wird mitgeschickt. Es wird kein Passwort mitgegeben.

= **PROGRAM** Die mit SEC-UID = *security_userid* generierte Benutzerkennung und das mit SEC-PASS = *security_passwd* generierte Passwort werden mitgeschickt.

Der Parameter SEC-TYPE = *security_type* wird nur dann ausgewertet, wenn in der zugehörigen PARTAPPL-Anweisung der Application Context Name **utm-secu** vereinbart wurde (siehe Parameter APPLICATION-CONTEXT = *context_name*, Seite 35). Mit jedem anderen Application Context wird SEC-TYPE standardmäßig auf den Wert NONE gesetzt.

security_userid
ist die Benutzerkennung, die als Teil der Zugangsschutzinformation mitgeschickt wird (siehe oben, Parameter SEC-TYPE). Die *security_userid* ist bis zu acht Zeichen lang. Sie ist Teil der CPI-C Side Information und kann mit dem Aufruf *Set_Conversation_Security_User_ID (CMSCSU)* modifiziert werden.

Dieser Parameter wird nur für SEC-TYPE = PROGRAM ausgewertet.

security_passwd
ist das Passwort, das als Teil der Zugangsschutzinformation mitgeschickt wird (siehe oben, Parameter SEC-TYPE). Das Passwort kann bis zu acht Zeichen lang sein. Es ist Teil der CPI-C Side Information und kann mit dem Aufruf *Set_Conversation_Security_Password (CMSCSP)* modifiziert werden.

Dieser Parameter wird nur für SEC-TYPE = PROGRAM ausgewertet.

Anmerkungen

1. Die mit der SYMDEST-Anweisung generierten Daten bilden einen Eintrag in der CPI-C Side Information, wobei die Felder *AP_title*, *AE_qualifier* und *application_context_name* aus dem Inhalt der zugehörigen PARTAPPL-Anweisung (siehe Parameter PARTNERAPPL) belegt werden.

   Das Feld *mode_name* in der Side Information wird nicht unterstützt (siehe auch “Abschnitt „Die CPI-C-Schnittstelle von OpenCPIC“ auf Seite 79”).

   Anstelle eines mit der SYMDEST-Anweisung generierten Symbolic Destination Name kann im CPI-C-Aufruf *Initialize_Conversation (CMINIT)* auch ein leerer Symbolic Destination Name (bestehend aus acht Leerzeichen) als Parameter *sym_dest_name* übergegeben werden. In diesem Fall ist das lokale CPI-C-Anwendungsprogramm für die Bereitstellung der notwendigen Informationen zuständig. Insbesondere müssen die Partneranwendung und der Name des Partneranwendungsprogramms bekannt

gemacht werden. Die Partneranwendung wird durch einen oder mehrere der CPI-C-Aufrufe *Set_AP_Title (CMSAPT)*, *Set_AE_Qualifier (CMSAEQ)* und *Set_Application_Context_Name (CMSACN)* angegeben.

Der Name des Partneranwendungsprogramms wird durch *Set_TP_Name (CMSPTN)* gesetzt (siehe auch Anmerkung 4).

Auch wenn im Aufruf *Initialize_Conversation (CMINIT)* ein mit einer SYMDEST-Anweisung generierter Symbolic Destination Name übergeben wird, kann der Eintrag in der Side Information der Conversation nachträglich durch *CMSAPT*, *CMSAEQ* bzw. *CMSACN* geändert werden. Beachten Sie jedoch, dass bei einer Modifizierung der Side Information durch mindestens einen der genannten CPI-C-Aufrufe der Parameter PARTAPPL in der SYMDEST-Anweisung und damit alle hierüber verfügbaren Adressinformationen über die Partneranwendung (d. h. *AP_title*, *AE_qualifier* und *application_context_name)* ungültig werden. Das bedeutet, wenn Sie z. B. den *AE_Qualifier* mit *CMSAEQ* ändern, so werden auch die Werte für *AP_title* und *application_context_name* in der Side Information gelöscht und müssen ggf. neu gesetzt werden.

Bei einem *Allocate (CMALLC)*-Aufruf prüft der OpenCPIC-Manager, ob mit dem aktuellen Side Information Eintrag durch *AP_title*, *AE_qualifier* und *application_context_name* adressierte Partneranwendung gültig ist. Wurde zuvor im *CMINIT* ein gültiger, nicht-leerer *symdest_name* übergeben und die Side Information nicht durch *CMSAPT*, *CMSAEQ* oder *CMSACN* geändert, ist die Gültigkeit der Partneranwendung durch die vorher erfolgte Prüfung der Konfigurierungsdaten garantiert. Wurde jedoch zwischen *CMINIT*- und *CMALLC*-Aufruf die Side Information durch mindestens einen der drei genannten Aufrufe modifiziert, wird anhand des aktuellen Side Information Eintrages geprüft, ob eine passende Partneranwendung in der Konfigurationsdatei gefunden werden kann. Es muss mindestens eines der drei genannten Felder belegt sein und die Kombination aus den belegten Feldern muss eindeutig eine mit einer PARTAPPL-Anweisung generierte Partneranwendung identifizieren. Ist das nicht der Fall, wird der *CMALLC* mit dem Returnwert *CM_PARAMETER_ERROR* abgelehnt. Ist die Partneranwendung dagegen eindeutig identifizierbar, so wird zum Dialogaufbau die vollständige Adressinformation aus der gefundenen PARTAPPL-Anweisung verwendet.

Mit den CPI-C-Aufrufen *Extract_AP_Title (CMEAPT)*, *Extract_AE_Qualifier (CMEAEQ)* und *Extract_Application_Context_Name (CMEACN)* können die aktuell gültigen Werte aus der Side Information abgefragt werden.

Eine weitere Möglichkeit zur Adressierung einer Partneranwendung stellt der CPI-C-Aufruf *Set_Partner_LU_Name (CMSPLN)* dar. In OpenCPIC kann mit diesem Aufruf der symbolische Name einer Partneranwendung angegeben werden, zu der mit einem nachfolgenden *Allocate (CMALLC)* eine Verbindung aufgebaut werden soll. Existiert in

der Konfigurierung eine entsprechende PARTAPPL-Anweisung, so wird die vollständige Adressinformation aus dieser Anweisung (also *AP_title*, *AE_qualifier* und *application_context_name*) in den Side Information Eintrag übernommen.

Mit dem CPI-C-Aufruf *Extract_Partner_LU_Name (CMEPLN)* kann der aktuell gültige Name der Partneranwendung abgefragt werden, solange keiner der Werte von *AP_title*, *AE_qualifier* und *application_context_name* geändert wurde.

Beachten Sie jedoch: Diese Art der Adressierung ist zwar bequemer, als die oben beschriebene mittels *CMSAPT*, *CMSAEQ* und *CMSACN*, sie ist jedoch nicht portabel. Die CPI-C-Aufrufe *CMSPLN* und *CMEPLN* werden offiziell nur für CRM unterstützt, die auf das LU6.2-Protokoll aufsetzen, nicht aber für OSI-TP CRM. Die Unterstützung dieser Aufrufe durch OpenCPIC stellt eine in der X/Open CPI-C-Schnittstelle nicht vorgesehene Erweiterung und damit eine Ausnahme dar.

2. Bei Verwendung der Aufrufe *Set_Partner_LU_Name (CMSPLN)* bzw. *Extract_Partner_LU_Name (CMEPLN)* darf der Name der Partneranwendung aufgrund der Definition der CPI-C-Schnittstelle insgesamt nicht länger als 17 Zeichen sein. *CMSPLN* liefert andernfalls den Returnwert *CM_PROGRAM_PARAMETER_CHECK*, während *CMEPLN* den Namen nach 17 Zeichen abschneidet.

3. Das OSI-TP-Protokoll unterstützt keine Conversation Security im Sinne von CPI-C. In der offiziellen CPI-C-Schnittstelle sind deshalb die Aufrufe *Set_Conversation_Security_Type (CMSCST)*, *Set_Conversation_Security_User_ID (CMSCSU)*, *Set_Conversation_Security_Password (CMSCSP)* und *Extract_Security_User_ID (CMESUI)* nur für LU6.2 CRM beschrieben. Abweichend von diesem Standard realisiert OpenCPIC ein internes Protokoll zum Austausch von Zugangsschutzinformationen zwischen OpenCPIC und UTM-Anwendungen und bietet die genannten CPI-C-Aufrufe als Schnittstelle für die Verwaltung der Security-Daten an. Ihre Verwendung ist jedoch nur mit einer UTM-Anwendung als Kommunikationspartner sinnvoll. Die Zugangsschutzinformationen werden beim Dialogaufbau im *TP-BEGIN-DIALOGUE-request* als Benutzerdaten (Initialization Data) mitgeschickt. OpenCPIC wertet jedoch keine bei einem Dialogaufbauwunsch empfangenen Zugangsschutzinformationen aus.

4. Gibt das CPI-C-Programm beim Aufruf *Initialize_Conversation (CMINIT)* acht Leerzeichen im Parameter *sym_dest.name* an und ist ein *SYMDEST.DEFAULT* generiert, so übernimmt OpenCPIC die zu *.DEFAULT* generierten Parameter in die Conversation. Dies ist eine Erweiterung von X/Open CPI-C.

3.1.5 APPLICATION-CONTEXT - Application Context definieren

Syntax

APPLICATION-CONTEXT	application_context_name**,**
OBJECT-IDENTIFIER	**=** context_objectid

Beschreibung

Mit dieser Anweisung generieren Sie einen Application Context. Ein Application Context umfasst die Menge der Regeln, die zwei Kommunikationspartner beim Austausch von Daten und Informationen über einen Dialog befolgen und über die sie sich zuvor einigen müssen. Der Application Context wird beim Aufbau einer Association ausgehandelt und gilt damit für alle Dialoge, die über diese Association betrieben werden.

OpenCPIC verlangt, dass ein Application Context mindestens die Application Service Elemente ACSE, TP-ASE und enthält. Weitere Bestandteile des Application Context sind die abstrakten Syntaxen sowie die Transfersyntax, nach deren Regeln die Kodierung und Dekodierung der Daten für die Übertragung erfolgt.

Ein Application Context wird über einen eindeutigen Objektbezeichner identifiziert, der durch eine autorisierte Registrierungsbehörde vergeben wird (siehe auch Seite 31).

Die APPLICATION-CONTEXT-Anweisung ist optional. Insbesondere, wenn Sie nur Verbindungen zu OpenCPIC- oder UTM-Partneranwendungen betreiben wollen, ist es nicht notwendig, eigene Application Context Namen zu definieren. Sie können statt dessen einen der vier vordefinierten Application Context Namen für die Kommunikation zur Partneranwendung verwenden (siehe Abschnitt „PARTAPPL - Partneranwendung definieren“ auf Seite 34).

Es ist nicht möglich, für selbstdefinierte Application Context Namen anzugeben, welche abstrakten und Transfersyntaxen der Application Context beinhalten soll. OpenCPIC ordnet selbstdefinierten Application Context Namen immer die abstrakten Syntaxen UDT und - abhängig vom Parameter CCR in der PARTAPPL-Anweisung - CCR, sowie die Transfersyntax BER zu.

application_context_name
: ist der symbolische Name, der innerhalb der Generierungsdatei zur Referenzierung des Application Context benutzt wird. Diesen Namen müssen Sie in der PARTAPPL-Anweisung angeben (Parameter APPL-CONTEXT), wenn Sie anstelle eines vordefinierten Application Context einen selbst definierten verwenden wollen.

Der *application_context_name* kann bis zu acht Zeichen lang sein.

Die Namen **def-cont**, **utm-secu** und **xatmi** sind für vordefinierte Application Context Namen reserviert und dürfen nicht verwendet werden.

context_objid
: ist der dem Application Context zugewiesene Objektbezeichner (siehe Seite 31).

Anmerkungen

Der Application Context Name ist Bestandteil der Side Information und kann mit dem CPI-C-Aufruf *Set_Application_Context_Name (CMSACN)* modifiziert bzw. mit dem Aufruf *Extract_Application_Context_Name (CMEACN)* abgefragt werden. Als Parameter bzw. Returnwert wird hier jeweils der Objektbezeichner übergeben. Weitere Informationen zur Side Information finden Sie in Abschnitt „SYMDEST - Kommunikationspartner definieren“ auf Seite 43, Anmerkung 1.

3.2 Generierungsprogramm starten

Mit dem Generierungsprogramm *ocpic_gen* können Sie aus einer Generierungsdatei eine binäre Konfigurationsdatei erzeugen (Format 1) sowie aus Konfigurationsdateien Generierungsdaten in ASCII-Form lesen (Format 2). Die Begriffe Generierungsdatei und Konfigurationsdatei sind am Anfang dieses Kapitels (Seite 29) erläutert.

Syntax

ocpic_gen [gen_file] [**-f** conf_file] (Format 1)

ocpic_gen -r [conf_file] (Format 2)

Beschreibung

ocpic_gen wertet den Inhalt der Umgebungsvariablen *OCPICDIR* aus. Ist *$OCPICDIR* nicht gesetzt, wird als Standardwert der Pfadname des OpenCPIC-Installationsverzeichisses (kurz *openCPIC-pfad*) angenommen.

Format 1: Binäre Konfigurationsdatei erzeugen

gen_file
: ist der Name der Generierungsdatei, aus der die Generierungsdaten gelesen werden sollen. Die Generierungsdatei müssen Sie mit einem Editor erstellen. Der Aufbau der Generierungsdatei ist im Abschnitt „Aufbau der Generierungsdatei“ auf Seite 30 beschrieben.

  Ist *gen_file* nicht angegeben, werden die Generierungsdaten aus der Standarddatei *$OCPICDIR/gen/genfile* gelesen.

conf_file
: ist der Name der Konfigurationsdatei, in die die Generierungsdaten in binärer Form geschrieben werden sollen. Ist *conf_file* nicht angegeben, erfolgt die Ausgabe in die Datei *$OCPICDIR/conffile*.

Der OpenCPIC-Manager liest während seiner Initialisierung die Konfigurationsdaten aus der Datei *$OCPICDIR/conffile*. Wenn Sie die Konfigurationsdatei mit *ocpic_gen -f* unter einem anderen Namen erzeugen, müssen Sie sie vor dem Start des OpenCPIC-Managers umbenennen bzw. nach *$OCPICDIR/conffile* kopieren.

Sie können die Konfigurationsdatei auch bei laufendem OpenCPIC-Manager neu erzeugen. Die neuen Konfigurationsdaten werden jedoch erst beim nächsten Start des OpenCPIC-Managers wirksam.

Hinweis für UTM-Anwender: Im Gegensatz zum UTM-Generierungsprogramm *kdcdef* löscht *ocpic_gen* keine Wiederanlauf-Informationen aus dem Verzeichnis *$OCPICDIR/SYNC* (siehe Abschnitt „Wiederanlauf (Recovery)“ auf Seite 144). Diese bleiben grundsätzlich bis zum nächsten Start des OpenCPIC-Managers erhalten.

Format 2: Generierungsdaten aus einer binären Konfigurationsdatei lesen

conf_file
: ist der Name der binären Konfigurationsdatei, aus der die Generierungsdaten gelesen werden sollen. Ist *conf_file* nicht angegeben, wird aus der Datei *$OCPICDIR/conffile* gelesen.

Die Generierungsdaten werden in lesbarer Form in dem in Abschnitt „Aufbau der Generierungsdatei“ auf Seite 30 beschriebenen Format nach *stdout* geschrieben. Sie können die Ausgabe in eine Datei umleiten und diese als Basis für eine Generierungsdatei verwenden.

Die Ausgabe der Generierungsdaten ist vollständig, d. h. sie enthält auch alle optionalen Parameter, die in der originalen Generierungsdatei, aus der die Konfigurationsdatei erzeugt wurde, eventuell fehlten.

Der Aufruf von *ocpic_gen -r* kann nützlich sein, um einen Generierungslauf zu prüfen.

3.3 Beispielgenerierung für OpenCPIC

Bei der Installation von OpenCPIC wird in *openCPIC-pfad/gen* die Datei *genfile* abgelegt, die eine Beispielgenerierung enthält.

In dieser Generierung werden für einen lokalen OpenCPIC-Manager zwei Partneranwendungen definiert, wovon eine ebenfalls ein OpenCPIC-Manager ist, während es sich bei der anderen Partneranwendung um eine UTM-Anwendung handelt. Für Verbindungen zum UTM-Partner soll das interne Security-Protokoll unterstützt werden.

Die beiden mit der LOCAPRO-Anweisung generierten Anwendungsprogramme können von Programmen der Partneranwendung unter diesem Namen angesprochen werden und werden dann automatisch durch die lokale Anwendung gestartet.

Mit den drei generierten Symbolic Destination Names wird eine einfache Adressierung der entsprechenden Anwendungsprogramme beim Kommunikationspartner ermöglicht.

Inhalt der Datei genfile

```
* ------------------------------------------------------------------------------------------
* OpenCPIC
* Sample generation file genfile
* ------------------------------------------------------------------------------------------------------------------------------


***** Local application: local OpenCPIC manager

LOCAPPL ocp_loc,

    APT = (2, 89, 111, 9),
    AEQ = 21


***** Partner application: remote OpenCPIC manager
***** Use default application context (1, 0, 10026, 6, 2) with CCR syntax included

PARTAPPL ocp_rem,

    APT = (2, 15, 99),
    AEQ = 100,
    APPL-CONTEXT = def-cont,
    ASSOCIATIONS = 10,
    CONTWIN = (2 4),
    CONNECT = 2,
    IDLETIME = 0,
    CCR = YES


***** Partner application: remote UTM application
***** CCR syntax always included

PARTAPPL utm_rem,
```

```
    APT = (2, 88, 235, 10),
    AEQ = 450,
    APPL-CONTEXT = utm-secu,
    ASSOCIATIONS = 10,
    CONTWIN = (2 4),
    CONNECT = 0,
    IDLETIME = 0


***** Local application programs: to be started automatically on receipt of
***** a conversation startup request from partner application

LOCAPRO sv_app_a,

    PATH = /home/ocpappl/server_a,
    LOGIN = gast

LOCAPRO sv_app_b,

    PATH = /home/ocpappl/server_b,
    LOGIN = gast


***** Symbolic destination names for active conversation startup by local
***** application programs


SYMDEST SYMDOCP0,

    PARTNER-APPL = ocp_rem,
    PARTNER-APRO = sv_app_0


SYMDEST SYMDOCP1,

    PARTNER-APPL = ocp_rem,
    PARTNER-APRO = sv_app_1


SYMDEST SYMDUTM0,

    PARTNER-APPL = utm_rem,
    PARTNER-APRO = UTMSVTAC,
    SEC-TYPE = PROGRAM,
    SEC-UID = OCPCLIEN,
    SEC-PASS = OCPPASSW


* --- End of sample generation file genfile ----------------------------------------
```

4 OpenCPIC administrieren

Dieses Kapitel beschreibt:

- wie Sie den OpenCPIC-Manager starten und beenden
- wie Sie sich Zustandsinformationen über OpenCPIC anzeigen lassen
- wie Sie Associations zu speziellen Anwendungen verwalten
- wie Sie Traceinformationen mitschreiben und auswerten

Arbeitsverzeichnis $OCPICDIR

OpenCPIC verwendet ein Arbeitsverzeichnis für die Ablage aller während des Betriebs benötigten bzw. erzeugten Dateien und Unterverzeichnisse. Standardmäßig ist *openCPIC-pfad* das Arbeitsverzeichnis. Wenn ein anderes Arbeitsverzeichnis verwendet werden soll (siehe Hinweis), müssen Sie die Umgebungsvariable *OCPICDIR* auf den entsprechenden (vollqualifizierten) Pfadnamen setzen. Das Verwenden von *OCPICDIR* ist insbesondere dann notwendig, wenn Sie mehrere OpenCPIC-Manager gleichzeitig auf einem System laufen lassen wollen. Jeder OpenCPIC-Manager benötigt sein eigenes Arbeitsverzeichnis.

Im Folgenden ist mit *$OCPICDIR* immer das Arbeitsverzeichnis des OpenCPIC-Managers, den Sie administrieren wollen, gemeint (also *openCPIC-pfad*, falls *OCPICDIR* nicht gesetzt ist).

Es wird dringend empfohlen, immer ein eigenes Arbeitsverzeichnis einzurichten, auch wenn nur ein OpenCPIC-Manager auf dem System läuft. Bitte beachten Sie:

- Die für den Betrieb des OpenCPIC-Managers verwendete Benutzerkennung muss Schreibrechte auf *$OCPICDIR* haben.
- Außerdem müssen Sie in *$OCPICDIR* zwei Unterverzeichnisse mit den Namen "PROT" und "SYNC" einrichten.

4.1 OpenCPIC-Manager starten

Vor dem Start muss, wie in Kapitel 3 beschrieben, die Konfigurationsdatei *conffile* erzeugt und im Arbeitsverzeichnis *$OCPICDIR* abgelegt werden.

Der OpenCPIC-Manager wird mit Hilfe der Shell-Prozedur *ocpic_start* gestartet. Rufen Sie das Programm *ocpic_mgr* bitte nicht direkt auf, es könnte sonst zu Problemen während der Initialisierung kommen.

Syntax

ocpic_start [**-tp**] [**-tm**] [**-c**]

Beschreibung

ocpic_start startet den OpenCPIC-Manager als Hintergrundprozess.

-tp — Mit dieser Option schalten Sie den sogenannten XAP-TP-Trace beim Start des OpenCPIC-Managers ein. Das bedeutet, es werden Traceinformationen des XAP-TP-Bausteins für Diagnosezwecke mitgeschrieben. Weitere Hinweise dazu finden Sie im Abschnitt “Abschnitt „Traceinformationen mitschreiben“ auf Seite 70”.

-tm — Mit dieser Option schalten Sie den Manager-Trace beim Start des OpenCPIC-Managers ein. Das bedeutet, es werden Traceinformationen des OpenCPIC-Managers für Diagnosezwecke mitgeschrieben. Weitere Hinweise dazu finden Sie im Abschnitt „Traceinformationen mitschreiben“ auf Seite 70.

-c — Mit dieser Option veranlassen Sie einen sogenannten Kaltstart des OpenCPIC-Managers.
Standardmäßig führt *ocpic_start* einen Warmstart durch. Dies ist notwendig, wenn Sie nach einem vorhergehenden Systemfehler einen Wiederanlauf (Recovery) durchführen wollen. Mehr zum Thema Wiederanlauf finden Sie im Kapitel „Globale Transaktionen“ auf Seite 141.

Beachten Sie bitte: Bei einem Kaltstart (Schalter *-c*) des OpenCPIC-Managers gehen alle Wiederanlauf-Informationen aus dem Verzeichnis *$OCPICDIR/SYNC* mit Ausnahme der Log-Damage-Records verloren (siehe Abschnitt „Wiederanlauf (Recovery)“ auf Seite 144). Sie sollten daher sicherstellen, dass die Informationen in den Log-Records nicht mehr benötigt werden, bevor Sie einen Kaltstart veranlassen. Falls bei einer vorangegangen Transaktionsverarbeitung ein Fehler aufgetreten sein sollte und Log-Records vorhanden sind, sollten Sie unbedingt einen Warmstart durchführen.

Nach dem erfolgreichen Start erscheinen am Bildschirm die folgenden Meldungen:

```
OpenCPIC manager: start
```

```
OpenCPIC [BD] 001 Start des OpenCPIC-Managers mit PID nnnn

Copyright (C)
2017 Fujitsu Technology Solutions GmbH
Alle Rechte vorbehalten

OpenCPIC [BD] 016 Initialisierung des OpenCPIC-Managers abgeschlossen
```

Der OpenCPIC-Manager ist nun bereit, Anforderungen von lokalen Anwendungsprogrammen oder entfernten Anwendungen entgegenzunehmen. Alle weiteren Meldungen des OpenCPIC-Managers werden in die Datei *$OCPICDIR/PROT/prot.mgr* ausgegeben.

4.2 OpenCPIC-Manager beenden

Den OpenCPIC-Manager beenden Sie mit dem folgenden Kommando:

ocpic_adm -e

Das Kommando *ocpic_adm* ist ein Dienstprogramm für die Administration von OpenCPIC und ist vollständig auf Seite 76 beschrieben. *ocpic_adm -e* bewirkt das Senden des Signals *SIGTERM* (15) an den OpenCPIC-Manager, worauf sich dieser geordnet beendet.

Dasselbe bewirken Sie mit dem Kommando:

kill -15 pid_mgr

Dabei geben Sie für *pid_mgr* die PID des OpenCPIC-Managers an.

4.3 Zustandsinformationen anzeigen

Mit dem Kommando *ocpic_sta* können Sie sich Zustandsinformationen über den OpenCPIC-Manager anzeigen lassen.

Syntax

ocpic_sta [**-l**]

Beschreibung

ocpic_sta gibt die Zustandsinformationen mit Hilfe des Unix-/Linux-Kommandos *pg* auf die Standardausgabe aus. Damit ist Vor- und Rückwärtsblättern innerhalb des ausgegebenen Textes mit den entsprechenden *pg*-Befehlen möglich.

Wenn Sie *ocpic_sta* ohne Schalter aufrufen, wird eine Kurzinformation ausgegeben.

-l	Dieser Schalter bewirkt, dass *ocpic_sta* anstelle der Kurzinformation eine ausführliche Zustandsinformation ausgibt.

4.3.1 Ausgabe von ocpic_sta (Kurzinformation)

```
        OpenCPIC V4.0
    >> STATUS BERICHT <<
```

```
PID des OpenCPIC-Managers : nnnn
Lokale Anwendung : <name>

Partneranwendung   max Asso          aufgebaute Asso    belegte Asso
partappl_name_1    <anzahl>          <anzahl>           <anzahl>
...                ...               ...                ...
partappl_name_n    <anzahl>          <anzahl>           <anzahl>
```

Bedeutung der Ausgabefelder

PID des OpenCPIC-Managers
: Prozess-Id des Programms *ocpic_mgr*

Lokale Anwendung
: Name der lokalen Anwendung, wie in der Generierungsdatei mit der LOCAPPL-Anweisung angegeben

Partneranwendung
: Name der Partneranwendung, wie in der Generierungsdatei mit der PARTAPPL-Anweisung angegeben

max Asso
: maximale Anzahl der Associations, die zu dieser Partneranwendung aufgebaut werden können (Parameter ASSOCIATIONS der PARTAPPL-Anweisung in der Generierungsdatei).

aufgebaute Asso
: Anzahl der aktuell aufgebauten Associations zu dieser Partneranwendung

belegte Asso
: Anzahl der zur Zeit belegten Associations zu dieser Partneranwendung

4.3.2 Ausgabe von ocpic_sta -l (ausführliche Information)

```
OpenCPIC V4.0
>> STATUS BERICHT <<
```

```
PID des OpenCPIC-Managers : nnnn
Lokale Anwendung : <name>
```

Tabelle der aktiven Anwendungsprogramme

AP-PID	Starter	TA-Zustand	Node-Position	Conversations
nnnn	<wert>	<zustand>	<position>	<anzahl>
...	...	...	...	...
nnnn	<wert>	<zustand>	<position>	<anzahl>

Tabelle der Partneranwendungen und Verbindungen

```
Partneranwendung: <name>
Associations:                 Maximum = <anzahl>
Contention Winner:            Minimum = <anzahl>,    Maximum = <anzahl>
```

Assoziations Zustand	Content Rolle	Dialogue Zustand	Functional Units	TA-Modus	Conv. ID	Conv. Zustand	AP-PID
<zustand>	<rolle>	<zustand>	<units>	<modus>	<id>	<zustand>	nnnn
...	...	...	...	...	...	...	...
<zustand>	<rolle>	<zustand>	<units>	<modus>	<id>	<zustand>	nnnn

```
Partneranwendung: %s
Associations:                 Maximum = <anzahl>
Contention Winner:            Minimum = <anzahl>,    Maximum = <anzahl>
...
```

Tabelle der lokalen TPSU Title

```
AP-PID nnnn:                    <anzahl>
                                ...

AP-PID nnnn:                    <anzahl>
                                ...

...
```

Tabelle der wartenden Conversation Startup Requests

```
Partneranwendung                Lokaler TPSU Title
<name>                          <titel>
...                             ...
<name>                          <titel>
```

Bedeutung der Ausgabefelder

PID des OpenCPIC-Managers
: Prozess-Id des Programms *ocpic_mgr*

Lokale Anwendung
: Name der lokalen Anwendung, wie in der Generierungsdatei mit der LOCAPPL-Anweisung angegeben

Tabelle der aktiven Anwendungsprogramme

AP-PID
: Prozess-Id des Anwendungsprogramms

Starter
: gibt an, von wem das Anwendungsprogramm gestartet wurde

= `MGR`
: Das AP wurde aufgrund eines empfangenen Dialogwunsches vom OpenCPIC-Manager gestartet.

= `USER`
: Das AP wurde durch ein Operatorkommando gestartet.

TA-Zustand
: gibt den aktuellen Zustand des Transaktionsknotens an, falls das AP an einer Transaktion beteiligt ist. Wenn das AP an keiner Transaktion beteiligt ist, bleibt dieses Feld leer.

  Falls sich der Transaktionsknoten noch vor oder in der Precommit-Phase befindet, und es wurde noch kein *TP-COMMIT-request/TP_ROLLBACK-request* abgesendet bzw. noch keine *TP-COM-*

MIT-indication/TP_ROLLBACK-indication empfangen, ist dieser Wert immer *IDLE*.
Die möglichen Ausgabewerte und deren Bedeutung entnehmen Sie bitte der Tabelle der Dialogzustände im “Abschnitt „Zustandswerte für Dialoge und Transaktionen“ auf Seite 66”.

Node-Position — gibt an, welche Position das AP innerhalb des Transaktionsbaumes einnimmt, falls das AP momentan an einer Transaktion beteiligt ist

- = `ROOT` — Das AP ist der Root-Knoten der Transaktion
- = `INTERMEDIATE` — Das AP ist ein Intermediate-Knoten der Transaktion
- = `LEAF` — Das AP ist ein Leaf-Knoten der Transaktion

Conversations — gibt die Anzahl der momentan aktiven Conversations des AP an

Tabelle der Partneranwendungen und Verbindungen

Partneranwendung — Name der Partneranwendung, wie in der Generierungsdatei in der PARTAPPL-Anweisung angegeben

Associations — - Maximum
maximale Anzahl der Associations, die zu dieser Partneranwendung aufgebaut werden können

Contention Winner — - Maximum
maximale Anzahl der Contention-Winner-Associations, die zu dieser Partneranwendung aufgebaut werden können

- Minimum
minimale Anzahl der Contention-Winner-Associations, die zu dieser Partneranwendung aufgebaut werden

Association-Zustand — momentaner Zustand der Association

- = `aassrq` — Die Association befindet sich in der Aufbauphase. Der lokale XAP-TP-Baustein baut die Association auf. Es wird auf Bestätigung des Aufbaus von der Partneranwendung gewartet. Dieser Zustand sollte nur vorübergehend sein.

= `bidding` Der lokale XAP-TP-Baustein will auf einer bisher freien Association, für die er Contention Loser ist, einen Dialog eröffnen. Dazu hat er zunächst ein *TP-BID-RI* gesendet und wartet nun auf eine Antwort von der Partneranwendung (d. h. auf ein *TP-BID-RC*). Dieser Zustand sollte nur vorübergehend angezeigt werden.

= `busy` Die Association ist aufgebaut und durch einen Dialog belegt.

= `free` Die Association ist aufgebaut und momentan nicht durch einen Dialog belegt.

= `rsp stop` Die Association befindet sich in der Aufbauphase. Der entfernte XAP-TP-Baustein baut die Association auf. Der lokale XAP-TP-Baustein wartet auf ein *DATA-GO* Signal von OSS. Dieser Zustand sollte nur vorübergehend angezeigt werden.

= `waitgo` Die Association befindet sich in der Aufbauphase. Der lokale XAP-TP-Baustein wartet auf ein *DATA-GO* Signal von OSS. Dieser Zustand sollte nur vorübergehend sein.

Content Rolle gibt an, ob die lokale Anwendung Contention Loser oder Contention Winner für diese Association ist

= `LOSER` Die lokale Anwendung ist Contention Loser.

= `WINNER` Die lokale Anwendung ist Contention Winner.

Dialog Zustand gibt den momentanen Zustand des Dialoges an, der die Association belegt. Falls die Association von keinem Dialog belegt ist, bleibt dieses Feld leer.
Die möglichen Ausgabewerte und deren Bedeutung entnehmen Sie bitte der Tabelle der Dialogzustände im “Abschnitt „Zustandswerte für Dialoge und Transaktionen“ auf Seite 66”.

Functional Units gibt an, welche OSI-TP-Funktionseinheiten (Functional Units) für den Dialog, der diese Association belegt, unterstützt werden. Die unterstützten Funktionseinheiten werden beim Dialogaufbau durch die Partneranwendungen ausgewählt und sind abhängig von *sync_level* und *transaction_control* der CPI-C-Conversation, die über diesen Dialog betrieben werden soll.
Wenn die Association durch keinen Dialog belegt ist, bleibt dieses Feld leer.
Die verschiedenen Funktionseinheiten werden bei der Ausgabe durch Großbuchstaben abgekürzt und mit ‘+’ verbunden. Die Abkürzungen haben folgende Bedeutung:

= D — *Dialogue* Functional Unit
Diese Funktionseinheit wird immer unterstützt.

= P — *Polarized Control* Functional Unit
Diese Funktionseinheit wird immer unterstützt.

= H — *Handshake* Functional Unit
Diese Funktionseinheit wird für *sync_level = CM_CONFIRM* bzw. *sync_level = CM_SYNC_POINT* unterstützt.

= C — *Commit* und *Chained Transactions* Functional Unit
Diese Funktionseinheiten werden für *sync_level = CM_SYNC_POINT* bzw. *sync_level = CM_SYNC_POINT_NO_CONFIRM* und *transaction_control = CM_CHAINED_TRANSACTIONS* unterstützt.

= U — *Commit* und *Unchained Transactions* Functional Unit
Diese Funktionseinheiten werden für *sync_level = CM_SYNC_POINT* bzw. *sync_level = CM_SYNC_POINT_NO_CONFIRM* und *transaction_control = CM_UNCHAINED_TRANSACTIONS* unterstützt.

Die OSI-TP-Funktionseinheit *Shared Control* wird in OpenCPIC nicht unterstützt. Die einzelnen OSI-TP-Funktionseinheiten (Functional Units) sind in ISO 10026 [32] beschrieben.

TA-Modus — gibt an, ob der Dialog, der die Association belegt, momentan an einer Transaktion beteiligt ist. Wenn die Association durch keinen Dialog belegt ist, bleibt dieses Feld leer.

= TRUE — Der Dialog ist Bestandteil einer Transaktion.

= FALSE — Der Dialog ist nicht Bestandteil einer Transaktion.

Conv. ID — gibt die Conversation ID der Conversation an, die über den Dialog betrieben wird, der die Association belegt. Wenn die Association durch keinen Dialog belegt ist oder dem Dialog keine Conversation zugeordnet ist, bleibt dieses Feld leer.

Conv. Zustand — gibt den *conversation_state* der Conversation an, die über den Dialog betrieben wird, der die Association belegt. Wenn die Association durch keinen Dialog belegt ist oder dem Dialog keine Conversation zugeordnet ist, bleibt dieses Feld leer.
Die angegebenen Zustandswerte entsprechen den in der CPI-C-Spezifikation [24] definierten Werten wie folgt:

= INITIALIZE	*CM_INITIALIZE_STATE*	(2)
= SEND	*CM_SEND_STATE*	(3)
= RECEIVE	*CM_RECEIVE_STATE*	(4)
= CONFIRM	*CM_CONFIRM_STATE*	(6)
= CONF_SEND	*CM_CONFIRM_SEND_STATE*	(7)
= CONF_DEAL	*CM_CONFIRM_DEALLOCATE_STATE*	(8)
= DEFER_RECV	*CM_DEFER_RECEIVE_STATE*	(9)
= DEFER_DEAL	*CM_DEFER_DEALLOCATE_STATE*	(10)
= SYNC_POINT	*CM_SYNC_POINT_STATE*	(11)
= SY_PO_SEND	*CM_SYNC_POINT_SEND_STATE*	(12)
= SY_PO_DEAL	*CM_SYNC_POINT_DEALLOCATE_STATE*	(13)
= INIT_INCOM	*CM_INITIALIZE_INCOMING_STATE*	(14)
= PREPARED	*CM_PREPARED_STATE*	(18)

AP-PID — gibt die Prozess-Id des Anwendungsprogramms an, die über den Dialog, der diese Association belegt, eine Conversation betreibt. Wenn die Association von keinem Dialog belegt ist, bleibt dieses Feld leer.

Tabelle der lokalen TPSU-Title

AP-PID — gibt Prozess-Id des lokalen Anwendungsprogramms an. Daneben erscheint die Liste aller lokalen Namen, unter denen sich dieses AP angemeldet hat (entweder durch *Specify_Local_TP_Name (CMSLTP)* oder beim Start durch den OpenCPIC-Manager aufgrund einer LOCAPRO-Anweisung in der Generierungsdatei).

Tabelle der wartenden Conversation Startup Requests

Partneranwendung — Name der Partneranwendung, von der ein Dialogaufbauwunsch empfangen wurde

Lokaler TPSU Title — lokaler Name (entspricht dem TPSU Title in OSI-TP, bzw. dem *TP_Name* in CPI-C), für den der Dialogwunsch empfangen wurde

4.3.3 Zustandswerte für Dialoge und Transaktionen

Neben dem Zustand der Conversations gibt *ocpic_sta* auch Zustandswerte für Dialoge und Transaktionsknoten aus.

Der OpenCPIC-Manager bildet CPI-C-Conversations auf OSI-TP-Dialoge ab. Für jede Conversation wird ein OSI-TP-Dialog aufgebaut, auf dem OSI-TP-Protokollelemente gesendet und empfangen werden. Die Protokollelemente sowie die Zustandstabelle für OSI-TP-Dialoge sind in 10026-2 [32], Part 2 definiert.

In OpenCPIC wird jeder Dialog von einer XAP-TP-Dialoginstanz verwaltet. Daneben gibt es noch eine weitere Instanz, die die Transaktionen, an denen die lokalen Anwendungsprogramme beteiligt sind, verwaltet und das Zwei-Phasen-Commit-Protokoll zur Transaktionsbeendigung überwacht. Diese Instanz wird als Kontrollinstanz bezeichnet.

Der von *ocpic_sta* ausgegebene Dialogzustand entspricht dem aktuellen Wert des XAP-TP-Environment-Attributs *AP_STATE* der zugehörigen Dialoginstanz. Er gibt den Zustand des Knotens im Dialogbaum wieder.
Der von *ocpic_sta* ausgegebene Transaktionszustand entspricht dem aktuellen Wert des XAP-TP-Environment-Attributs *AP_TP_STATE* der Kontrollinstanz bezüglich der zugehörigen Transaktion. Er gibt den Zustand des Knotens im Transaktionsbaum wieder.

XAP-TP [25] definiert 35 verschiedene Zustandswerte für Dialog- und Kontrollinstanzen, wobei die Zustände 1-25 den in der ISO-Norm 10026-2 definierten Zuständen für OSI-TP Dialoge entsprechen.

Die folgende Tabelle gibt an, welche Zustandswerte von *ocpic_sta* ausgegeben werden und welchem OSI-TP-Zustand sie entsprechen. Für Zustände, die nicht in ISO 10026, sondern nur in XAP-TP definiert sind, bleibt die entsprechende Spalte leer.

Dialog-Zustand	**Beschreibung**	**ISO 10026**
`IDLE`	Keine Verbindung	1
`WALLOCcnf`	Die lokale Seite versucht, eine Association zu belegen.	
`ALLOCATED`	Die lokale Seite hat eine Association belegt und versucht nun, eine Verbindung zum Partner aufzubauen.	
`DATA_XFER`	Die lokale Seite ist sendeberechtigt.	2
`RECEIVE`	Die lokale Seite ist nicht sendeberechtigt.	3
`ERROR_RECEIVE`	Die lokale Seite ist nicht sendeberechtigt, hat einen *TP-U-ERROR-request* abgeschickt und wartet auf den Erhalt des Senderechts.	4
`ERROR`	Die lokale Seite ist sendeberechtigt und hat eine *TP-U-ERROR-indication* empfangen. Die Sendeberechtigung wird abgegeben.	5

Dialog-Zustand	Beschreibung	ISO 10026
WHANDcnf	Die lokale Seite hat einen *TP-HANDSHAKE-request* abgeschickt und wartet auf den Empfang der *TP-HANDSHAKE-confirmation*.	6
WHANDrsp	Die lokale Seite hat eine *TP-HANDSHAKE-indication* empfangen und noch keine *TP-HANDSHAKE-response* abgeschickt.	7
WENDcnf	Die lokale Seite hat einen *TP-END-DIALOGUE-request* mit *confirmation=TRUE* abgeschickt und wartet auf den Empfang der *TP-END-DIALOGUE-confirmation*.	11
WENDrsp	Die lokale Seite hat eine *TP-END-DIALOGUE-indication* mit *confirmation=TRUE* erhalten und noch keine *TP-END-DIALOGUE-response* abgeschickt.	12
WHAND_GCcnf	Die lokale Seite hat einen *TP-HANDSHAKE-AND-GRANT-CONTROL-request* abgeschickt und wartet auf den Empfang der *TP-HANDSHAKE-AND-GRANT-CONTROL-confirmation*.	13
WHAND_GCrsp	Die lokale Seite hat eine *TP-HANDSHAKE-AND-GRANT-CONTROL-indication* empfangen und noch keine *TP-HANDSHAKE-AND-GRANT-CONTROL-response* abgeschickt.	14
WREADYind	Die lokale Seite hat einen *TP-PREPARE-request* mit *data_permitted=FALSE* abgeschickt und wartet auf den Empfang der *TP-READY-indication*.	15
WREADY_DATAP	Die lokale Seite hat einen *TP-PREPARE-request* mit *data_permitted=TRUE* abgeschickt und wartet auf den Empfang der *TP-READY-indication*.	16
READY	Die lokale Seite hat eine *TP-READY-indication* erhalten.	17
WPREP_ALLreq	Die lokale Seite hat eine *TP-PREPARE-indication* mit *data_permitted=FALSE* erhalten.	18
WPREP_DATAP	Die lokale Seite hat eine *TP-PREPARE-indication* mit *data_permitted=TRUE* erhalten.	19
PREPARING	Die lokale Seite hat einen *TP-PREPARE-ALL-request* abgeschickt. Damit befindet sie sich in der ersten Phase des Zwei-Phasen-Commit-Protokolls (Precommit-Phase).	
LOGGIN_READY	Die lokale Seite hat im Zustand *PREPARING* eine *TP-READY-ALL-indication* erhalten. Damit ist die erste Phase des Zwei-Phasen-Commit-Protokolls (Precommit-Phase) abgeschlossen.	

Dialog-Zustand	Beschreibung	ISO 10026
WCOMMITind	Die lokale Seite hat im Zustand *LOGGIN_READY* einen *TP-COMMIT-request* abgeschickt Damit beginnt die zweite Phase des Zwei-Phasen-Commit-Protokolls (Commit-Phase).	20
COM_WDONEreq	Die lokale Seite hat im Zustand *WCOMMITind* eine *TP-COMMIT-indication* erhalten. Die lokalen Resource Manager werden aufgefordert, Ihre Daten in den Endzustand zu bringen, die im Laufe der Transaktion vorgenommenen Datenmodifikationen werden damit fest. Dieser Zustand wird auch eingenommen, wenn die lokale Seite im Zustand *WCOM_COMPind* eine *TP-U-ABORT-indication* oder *TP-P-ABORT-indication* erhält.	21
WCOM_COMPind	Die lokale Seite hat im Zustand *COM_WDONEreq* einen *TP-DONE-request* abgeschickt. Alle lokalen Resource Manager haben Ihre Datenänderungen abgeschlossen.	22
ROL_WDONEreq	Die lokale Seite hat vor oder in der Precommit-Phase ein Protokollelement abgeschickt oder empfangen, welches einen Rollback der Transaktion zur Folge hat.	23
WROL_COMPind	Die lokale Seite hat im Zustand *ROL_WDONEreq* einen *TP-DONE-request* abgeschickt. Alle lokalen Resource Manager haben Ihre Datenänderungen rückgängig gemacht.	24
ZOMBIE	Die lokale Seite hat vor Beginn der Precommit-Phase ein Protokollelement erhalten, welches den Dialog beendet hat (z. B. *TP-U-ABORT-indication*), wobei *Rollback=FALSE* gesetzt war. Der Dialog existiert damit zwar nicht mehr, aber die Instanz bleibt belegt, solange bis die aktuelle Transaktion beendet ist.	25
HEURISTIC_LO	Diesen Zustand nimmt der Transaktionsknoten nach Beendigung der Transaktion ein, wenn es während der Commit-Phase infolge heuristischer Entscheidungen (siehe Seite 27) möglicherweise zu Inkonsistenzen zwischen den beteiligten Transaktionsknoten gekommen ist.	

4.4 Associations abbauen

Als Systemverwalter können Sie momentan bestehende Associations zu einer Partneranwendung durch die lokale Anwendung abbauen lassen. Dazu rufen Sie das Kommando *ocpic_adm* mit dem Schalter *-d* (disconnect) bzw. *-a* (abort) auf.

Syntax

ocpic_adm {**-d** | **-a**} partappl_name

Beschreibung

-d	Mit diesem Schalter veranlassen Sie, dass alle freien Associations, die im Moment zur Partneranwendung *partappl_name* bestehen, abgebaut werden. Eine Association heißt frei, wenn sie durch keinen Dialog belegt ist.
-a	Mit diesem Schalter veranlassen Sie, dass alle bestehenden Associations zur Partneranwendung *partappl_name* sofort abgebrochen werden, unabhängig davon, ob sie durch Dialoge belegt sind oder nicht.
partappl_name	ist der symbolische Name der Partneranwendung, wie er in der Generierungsdatei mit der PARTAPPL-Anweisung generiert wurde.

Anmerkungen

1. Über den momentanen Zustand aller bestehenden Associations können Sie sich mit dem Kommando *ocpic_sta -l* informieren (siehe ”Abschnitt „Zustandsinformationen anzeigen“ auf Seite 58”).
2. Das Abbauen nicht benötigter Associations kann auch über den Generierungsparameter IDLETIME in der PARTAPPL-Anweisung gesteuert werden. Wenn Sie hier einen Wert $t > 0$ angeben, wird eine Association automatisch abgebaut, wenn sie länger als t Sekunden durch keinen Dialog belegt ist.

4.5 Traceinformationen mitschreiben

Traceinformationen sind ein Hilfsmittel für die Fehlerdiagnose in OpenCPIC. Sie können durch Einschalten eines Trace erzeugt werden. Der Trace schreibt die Informationen während des Programmablaufs in spezielle Protokolldateien.

In OpenCPIC gibt es die folgenden Traces:

1. XAP-TP-Trace
   Ist dieser Trace eingeschaltet, wird das gesamte OSI-TP-Protokoll unterhalb der XAP-TP-Schnittstelle mitgeschrieben. Der XAP-TP-Trace beinhaltet auch den OSS-Trace.
2. Manager-Trace
   Ist dieser Trace eingeschaltet, werden interne Ablaufinformationen des OpenCPIC-Managers oberhalb der XAP-TP-Schnittstelle protokolliert.
3. CPI-C-Trace
   Dieser Trace ist Bestandteil der Bibliothek *libocpic.a* und legt ein Protokoll aller CPI-C- und TX-Aufrufe eines CPI-C-Anwendungsprogramms an.
4. XATMI-Trace
   Dieser Trace ist Bestandteil der XATMI-Bibliothek *libxtclt.a* und legt ein Protokoll aller XATMI-Aufrufe eines XATMI-Client-Programms an.

Sie können einzelne Traces oder auch mehrere Traces gleichzeitig einschalten. Jeder Trace legt seine eigenen Protokolldateien an. Die Protokolldateien des XAP-TP- und des Manager-Trace werden im Verzeichnis *$OCPICDIR/PROT* abgelegt. Die Protokolldateien des CPI-C- und des XATMI-Trace werden im aktuellen Arbeitsverzeichnis des Anwendungsprogramms erzeugt.

Namenskonventionen der Protokolldateien

Die einzelnen Traces legen ihre Protokolldateien unter folgenden Namen an:

XAP-TP-Trace:	**mgrlog.xap.**suff1.suff2
Manager-Trace:	**mgrlog.**suff
CPI-C-Trace:	**OCPIC**pid**.**suff
XATMI-Trace	**XTC**pid.suff

suff und *pid* sind Platzhalter mit der folgenden Bedeutung:

suff,suff1,suff2 ganzzahliger Dateinamen-Suffix

Der Name einer Protokolldatei wird um einen ganzzahligen Suffix ergänzt, der, beginnend mit 0, bei jedem Dateiwechsel um eins hochgezählt wird. Ein Dateiwechsel kann durch ein Administrationskommando veranlasst werden oder automatisch beim Überschreiten einer bestimmten Maximalgröße erfolgen (siehe nachfolgenden Abschnitt bzw. “Abschnitt „CPI-C-Trace aktivieren“ auf Seite 89”). Es entsteht so eine Folge von Protokolldateien, wobei der Suffix *suff* deren Reihenfolge angibt.

pid — Prozess-Id des Anwendungsprogramms

Bei den Protokolldateien des CPI-C- und des XATMI-Trace ist die Prozess-Id des Anwendungsprogramms Bestandteil des Dateinamens, um eine eindeutige Zuordnung zwischen Protokolldatei und Anwendungsprogramm zu ermöglichen.

4.5.1 XAP-TP-Trace und Manager-Trace aktivieren

Sie können den XAP-TP-Trace und den Manager-Trace bereits beim Start des OpenCPIC-Managers aktivieren. Dies erfolgt mit den Optionen *-tp* bzw. *-tm* des Kommandos *ocpic_start* (siehe “Abschnitt „OpenCPIC-Manager starten“ auf Seite 56”).

Die Steuerung des XAP-TP-Trace und des Manager-Trace erfolgt bei laufendem OpenCPIC-Manager mit dem Kommando *ocpic_adm -t*:

Syntax

ocpic_adm -t {**ton** | **mon** | **bon** | **tof** | **mof** | **bof** | **tfl** | **mfl** | **bfl**}

Beschreibung

ton	XAP-TP trace on schaltet den XAP-TP-Trace ein
mon	manager trace on schaltet den Manager-Trace ein
bon	both traces on schaltet den XAP-TP-Trace und den Manager-Trace ein
tof	XAP-TP trace off schaltet den XAP-TP-Trace aus
mof	manager trace off schaltet den Manager-Trace aus
bof	both traces off schaltet den XAP-TP-Trace und den Manager-Trace aus

tfl	XAP-TP trace flush erzwingt beim XAP-TP-Trace einen Wechsel der Protokolldatei *mgrlog.xap.suff* nach *mgrlog.xap.suff+1*
mfl	manager trace flush erzwingt beim Manager-Trace einen Wechsel der Protokolldatei *mgrlog.suff* nach *mgrlog.suff+1*
bfl	both traces flush erzwingt beim XAP-TP-Trace einen Wechsel der Protokolldatei *mgrlog.xap.suff* nach *mgrlog.xap.suff+1 und* beim Manager-Trace einen Wechsel der Protokolldatei *mgrlog.suff* nach *mgrlog.suff+1*

Anmerkungen

1. Das Aus- und anschließende Wiedereinschalten eines Trace bewirkt implizit einen Wechsel der zugehörigen Protokolldatei.
2. Der Manager-Trace veranlasst einen automatischen Wechsel der Protokolldatei, sobald die aktuelle Protokolldatei die Maximalgröße von 512 KByte überschritten hat. Der XAP-TP-Baustein veranlasst einen automatischen Wechsel der Protokolldatei, wenn eine gewisse Größe erreicht ist. Dabei wird *suff2* um 1 hochgezählt. Wird der Dateiwechsel per Administrationsbefehl (*ocpic_adm -t tfl* bzw. *ocpic_adm -t bfl*) herbeigeführt, so wird *suff1* um 1 hochgezählt.
3. Standardmäßig werden maximal zehn Protokolldateien für jeden Trace angelegt. Beim Manager-Trace beginnt, nachdem die Protokolldatei mit dem Suffix 9 (*mgrlog.9*) geschlossen wurde, die Zählung wieder mit 0 und in der Folge werden bereits vorhandene Protokolldateien überschrieben.
   Beim XAP-TP Trace wird der Suffix fortlaufend weitergezählt, jedoch wird beim Öffnen der Datei *mgrlog.xap.suff* die Datei *mgrlog.xap.suff-10* gelöscht.
   Die Anzahl der maximal gehaltenen Protokolldateien können Sie ändern, indem Sie die Umgebungsvariable *OCP_TRACELIMIT* vor dem Start des OpenCPIC-Managers auf den gewünschten Wert setzen. Es werden jedoch immer mindestens zwei Protokolldateien angelegt. Setzen Sie *OCP_TRACELIMIT* auf den Wert 0, bedeutet dies, dass es keine Obergrenze für die Anzahl der Protokolldateien gibt.

4.5.2 CPI-C-Trace und XATMI-Trace aktivieren

Die Traces der Bibliothek *libocpic.a* und *libxtclt.a* werden über Umgebungsvariablen gesteuert.

Die Umgebungsvariablen müssen in der Arbeitsumgebung des Anwendungsprogramms vor dem Start des Programms gesetzt und exportiert werden. Für Anwendungsprogramme, die durch den OpenCPIC-Manager gestartet werden, sollten die Umgebungsvariablen dementsprechend vor dem Start des OpenCPIC-Managers gesetzt werden.

Mit welchen Umgebungsvariablen Sie den CPI-C-Trace steuern, ist im “Abschnitt „CPI-C-Trace aktivieren“ auf Seite 89” beschrieben.

Mit welchen Umgebungsvariablen Sie den XATMI-Trace steuern, ist im “Abschnitt „XATMI-Trace aktivieren“ auf Seite 119” beschrieben.

4.5.3 Traceinformationen aufbereiten

Die von den Traces angelegten Protokolldateien enthalten Binärinformationen, die zunächst aufbereitet, d. h. in ein lesbares Format umgewandelt werden müssen. Da die Protokolldateien von verschiedenen Traces unterschiedliche Binärformate aufweisen, gibt es unterschiedliche Aufbereitungsbefehle.

Mit Ausnahme des Schnittstellen-Trace (Trace-Level 1), des CPI-C-Trace und des XATMI-Trace dienen die von den Traces mitgeschriebenen Informationen hauptsächlich als Diagnoseunterlagen für den Service. Sie sind für Sie als Anwender nur sehr eingeschränkt nutzbar und Sie sollten sich bei einem Verdacht auf Fehlverhalten von OpenCPIC mit den **nicht aufbereiteten** Protokolldateien an Ihren Service wenden. Zur Vordiagnose kann es jedoch sinnvoll sein, wenn bereits durch den Anwender eine Aufbereitung der Traceinformationen erfolgt.

4.5.3.1 Protokolldateien des XAP-TP-Trace aufbereiten

Die Protokolldateien des XAP-TP-Trace (*mgrlog.xap.suff*) werten Sie mit dem OSS-Programm *openCPIC-pfad/oss/step* aus.

Syntax

step [**-h**] [**-d**] [**-l=**nnn[**k**]] [**-s={n|l|m|h}**] [**-ps=[t][s][p][a][F]**] [**-cref=**n] [**-f=**hh[**:**mm[**:**ss]]] [**-t=**hh[**:**mm[**:**ss]]] tracefile [tracefile ...]

Beschreibung

-h — Es erfolgt die Ausgabe aller Schalter von *step* mit kurzen Erläuterungen. Alle weiteren angegebenen Schalter werden ignoriert.

-d — Ausgabe aller Benutzerdaten im Hexadezimalformat. step versucht standardmäßig, alle Presentation-, ACSE- und FTAM-Syntaxen, die nach den ASN.1 Basic Encoding Rules (BER) kodiert sind, zu dekodieren und aufzubereiten. Mit diesem Schalter wird die Dekodierung der Benutzerdaten unterdrückt.

-l=nnn[k] — *nnn* gibt die maximale Länge der Benutzerdaten (in Byte oder KByte) an, die aufbereitet werden. Fehlt dieser Schalter, werden alle Benutzerdaten in ihrer vollen Länge ausgegeben. Der Wertebereich von *nnn* ist 0 - 999. Ist nnn von **k** gefolgt, ist die Längenangabe in KByte, andernfalls in Byte.

-s — Funktionsumfang der Ausgabe von schutzwürdigen Informationen.

=n	(no)	alle Informationen werden ausgegeben
=l	(low)	Passworte werden nicht ausgegeben
=m	(medium)	Benutzerkennungen, Account-Nummern und Passworte werden nicht ausgegeben (Standardwert)
=h	(high)	wie ‘m’, außerdem werden keine Dateinamen ausgegeben

-ps — Mit diesem Schalter wird die Protokollschicht ausgewählt, deren Information ausgegeben werden soll. Jede mögliche Kombination der angegebenen Flags ist zulässig.

t — alle Transportsystem-Ereignisse; nicht sichtbar werden *T-DATAIN*-Ereignisse und alle ausgehenden Protokollelemente

s	alle Session-Ereignisse
p	alle Presentation-Ereignisse
a	alle ACSE-Ereignisse
F	alle FTAM-Ereignisse

-cref=n	Es werden nur Tracedatensätze ausgegeben, die die Verbindungsreferenz *n* enthalten.
-f	Es werden nur Tracedatensätze ausgegeben, die ab dem angegebenen Zeitpunkt aufgezeichnet wurden. Die Zeit wird im Format hh:mm:ss (Stunden:Minuten:Sekunden) angegeben.
-t	Es werden nur Tracedatensätze ausgegeben, die bis zum angegebenen Zeitpunkt aufgezeichnet wurden. Die Zeit wird im Format hh:mm:ss (Stunden:Minuten:Sekunden) angegeben.
tracefile	Name der Protokolldatei(en), die ausgewertet werden sollen.

4.5.3.2 Protokolldateien des Manager-Trace aufbereiten

Die Protokolldateien des Manager-Trace (*mgrlog.suff*) werten Sie mit folgendem Kommando aus:

Syntax

ocpic_adm -l tracefile

Beschreibung

tracefile	ist der Unix-/Linux-Pfadname der binären Protokolldatei, die aufbereitet werden soll, z. B. *$OCPICDIR/PROT/mgrlog.0*

Die aufbereiteten Daten werden auf die Standardausgabe (*stdout*) ausgegeben und können in eine Datei umgeleitet werden.

4.5.3.3 Protokolldateien des CPI-C-Trace und XATMI-Trace aufbereiten

Wie Sie die Protokolldateien des CPI-C-Trace aufbereiten, ist im “Abschnitt „CPI-C-Trace aktivieren“ auf Seite 89” beschrieben.

Die Protokolldateien des XATMI-Trace müssen nicht extra aufbereitet werden, sie sind sofort lesbar.

4.6 Administration mit ocpic_adm (Übersicht)

Dieser Abschnitt enthält eine zusammenfassende Beschreibung des Administrationskommandos *ocpic_adm*.

Syntax

ocpic_adm **-t** {**ton**|**mon**|**bon**|**tof**|**mof**|**bof**|**tfl**|**mfl**|**bfl**} | **-d** partappl_name | **-a** partappl_name | **-e** | **-l** tracefile

Beschreibung

-t	Steuerung des XAP-TP- und den Manager-Trace (siehe “Abschnitt „Traceinformationen mitschreiben“ auf Seite 70”)
-d	Abbau aller freien Association zur Partneranwendung *partappl_name* (siehe “Abschnitt „Associations abbauen“ auf Seite 69”)
-a	Abbau aller bestehenden Association zur Partneranwendung *partappl_name* (siehe Abschnitt „Associations abbauen“ auf Seite 69)
-e	Beenden des OpenCPIC-Managers (siehe “Abschnitt „OpenCPIC-Manager beenden“ auf Seite 57”)
-l	Aufbereitung der Protokolldatei *tracefile* (siehe “Abschnitt „Traceinformationen aufbereiten“ auf Seite 73”)

5 CPI-C-Anwendungsprogramme erstellen

Das Implementieren von CPI-C-Anwendungsprogrammen für OpenCPIC setzt voraus, dass Sie mit der Schnittstelle CPI-C vertraut sind.
Falls Sie gesicherte Transaktionen durchführen wollen benötigen Sie auch Kenntnisse der Schnittstelle TX (Transaction Demarcation Interface) sowie ggf. der Schnittstelle für Datenbankzugriffe in Ihrem Datenbanksystem, z. B. SQL. (Die Schnittstelle für den Datenbankzugriff ist nicht Bestandteil von OpenCPIC und wird in diesem Handbuch nicht behandelt.) Kenntnisse der XA-Schnittstelle sind von Vorteil, jedoch müssen Sie XA-Funktionen nicht direkt aus Ihrem Anwendungsprogramm aufrufen, da die Kommunikation zwischen Anwendungsprogramm und lokalem Resource Manager über XA von den TX-Funktionen übernommen wird.

Die in diesem Kapitel gemachten Angaben zu CPI-C und TX beziehen sich jeweils auf die X/Open Definition der genannten Schnittstellen. Die entsprechenden X/Open Dokumente finden Sie im Literaturverzeichnis am Ende des Buches. CPI-C 2.0 ist in [24], TX in [26] beschrieben.

Die CPI-C- und TX-Aufrufe sind in der Bibliothek *libocpic.a* realisiert. Gegenüber den X/Open Definitionen beinhaltet *libocpic.a* sowohl einige Einschränkungen als auch spezielle Erweiterungen. Die Unterschiede zur X/Open Definition sowie Besonderheiten, die Sie beim Programmieren mit *libocpic.a* beachten sollten, sind im “Abschnitt „Anwendungsprogramme mit libocpic.a implementieren“ auf Seite 78” beschrieben.

Um ein ablauffähiges Programm zu erstellen, binden Sie die Bibliothek *libocpic.a* zu den Objektmodulen Ihres Anwendungsprogramms. Wollen Sie Transaktionen unter Beteiligung lokaler Resource Manager durchführen, müssen Sie die XA-Bibliothek Ihres Datenbank-Produktes ebenfalls dazubinden. Die XA-Aufrufe für die Abwicklung gesicherter Transaktionen werden intern von den TX-Funktionen durchgeführt. OpenCPIC bietet Ihnen eine komfortable Möglichkeit zur Erstellung von Anwendungsprogrammen (siehe “Abschnitt „Anwendungsprogramme binden“ auf Seite 87”).

5.1 Anwendungsprogramme mit libocpic.a implementieren

Dieser Abschnitt enthält allgemeine Hinweise zur CPI-C- und TX-Schnittstelle von OpenCPIC sowie Besonderheiten, die Sie bei der Implementierung von Anwendungsprogrammen mit *libocpic.a* beachten sollten.

5.1.1 Allgemeine Hinweise

Programmiersprachen

OpenCPIC-Anwendungsprogramme können in C oder in COBOL erstellt werden. Die X/Open-Spezifikationen für CPI-C und TX enthalten jeweils spezielle Hinweise für C- und COBOL-Programme.

Benötigte Include-Dateien

C-Programme müssen für die Verwendung von CPI-C-Aufrufen die Datei *cpic.h* includieren. Für die Verwendung von TX-Aufrufen ist die Datei *tx.h* erforderlich.

COBOL-Programme müssen die Datei *CMCOBOL* (CPI-C-Definitionen) sowie die Dateien *TXINFDEF* und *TXSTATUS* (TX-Definitionen) includieren.

Für den XA-Anschluss wird die Datei *xa.h* benötigt. Der Source für den XA-Anschluss wird in der Datei *openCPIC-pfad/xa/xa_info.c* zur Verfügung gestellt.

Alle genannten Include-Dateien werden bei der Installation von OpenCPIC im Verzeichnis *openCPIC-pfad/include* abgelegt.

Verwendung von Signalen

In OpenCPIC wird das Signal 13 (*SIGPIPE*) zur Steuerung der Kommunikation zwischen OpenCPIC-Manager und Anwendungsprogrammen verwendet. Daher darf dieses Signal in OpenCPIC-Anwendungsprogrammen nicht verwendet oder abgefangen werden. Darüberhinaus bestehen keine Einschränkungen bezüglich der Verwendung von Signalen.

Erzeugung von Kindprozessen

Da der OpenCPIC-Manager zur Identifikation und Verwaltung der lokalen Anwendungsprogramme deren Prozess-Id nutzt, dürfen CPI-C- und TX-Aufrufe in OpenCPIC-Anwendungsprogrammen nicht an Kindprozesse weitergeleitet werden. Alle CPI-C- und TX-Aufrufe müssen aus demselben Prozess heraus erfolgen. Insbesondere müssen automatisch gestartete Anwendungsprogramme ihre Aufrufe aus demselben Prozess absetzen, mit dem sie durch den OpenCPIC-Manager gestartet wurden.

5.1.2 Die CPI-C-Schnittstelle von OpenCPIC

OpenCPIC bietet im wesentlichen eine Implementierung der von X/Open definierten Schnittstelle CPI-C 2.0, wie sie in [24] beschrieben ist. Wir verweisen daher, insbesondere in Bezug auf Syntax und Semantik der CPI-C-Aufrufe, auf dieses Dokument. Der vorliegende Abschnitt beschreibt lediglich die von der offiziellen X/Open Dokumentation abweichenden Eigenschaften und spezielle Besonderheiten der CPI-C-Schnittstelle von OpenCPIC.

CPI-C Side Information

Wie in [24] beschrieben, können in der CPI-C Side Information Adressdaten zu jeder Partneranwendung in einem eigenem Eintrag beschrieben werden. Ein solcher Eintrag in der Side Information wird über einen sogenannten Symbolic Destination Name referenziert, der beim *Initialize_Conversation (CMINIT)*-Aufruf übergeben werden kann.

In OpenCPIC können Sie Side-Information-Einträge mit Hilfe der Generierungsanweisung SYMDEST erzeugen. Der symbolische Name der SYMDEST-Anweisung entspricht dabei dem Symbolic Destination Name in CPI-C.

Den in [24] beschriebenen Attributen der CPI-C Side Information entsprechen in OpenCPIC folgende Generierungsparameter:

Attribut	**Generierungsparameter in der SYMDEST-Anweisung**
AP_title	PARTNER-APPL (Parameter APT in der zugehörigen PARTAPPL-Anweisung)
AE_qualifier	PARTNER-APPL (Parameter AEQ in der zugehörigen PARTAPPL-Anweisung)
application_context_name	PARTNER-APPL (Parameter APPL-CONTEXT in der zugehörigen PARTAPPL-Anweisung)
partner_LU_name	PARTNER-APPL
TP_name	PARTNER-APRO
mode_name	nicht unterstützt (Standardwert: acht Leerzeichen)
conversation_security_type	SEC-TYPE
security_user_ID	SEC-UID
security_password	SEC-PASS

Senden und Empfangen von Benutzerdaten

Mit einem Aufruf von *Send_Data (CMSEND)* bzw. *Receive (CMRCV)* können maximal 32767 Byte Benutzerdaten gesendet bzw. empfangen werden. Dieser Maximalwert kann auch mit *Extract_Maximum_Buffer_Size (CMEMBS)* abgefragt werden.

Benutzerdaten und Statusinformationen werden stets in getrennten *Receive (CMRCV)*-Aufrufen zurückgeliefert. Das heißt, wenn *CMRCV* einen Wert für *data_received* ungleich *CM_NO_DATA_RECEIVED* zurückliefert, ist *status_received* immer gleich *CM_NO_STATUS_RECEIVED*. Demzufolge kann eine Conversation nie in den Zustand *CM_SEND_PENDING_STATE* gelangen.

Übersicht über die CPI-C-Aufrufe

Die folgende Tabelle gibt einen Überblick über alle in CPI-C 2.0 definierten Aufrufe. Der Spalte “OpenCPIC” können Sie entnehmen, ob der Aufruf in OpenCPIC unterstützt wird.

Ein “+” in dieser Spalte bedeutet, der Aufruf wird unterstützt und entspricht genau der X/Open Definition.

Ein “*” bedeutet, es gibt Besonderheiten oder Einschränkungen gegenüber X/Open CPI-C 2.0. Diese sind in den nachfolgenden Anmerkungen im einzelnen beschrieben. Die Nummer in der Spalte “Anmerkung” verweist dabei auf die entsprechende Anmerkung unter der Tabelle.

Ein “-” in der Spalte “OpenCPIC” bedeutet, der Aufruf wird in OpenCPIC nicht unterstützt und liefert immer den Returnwert *CM_CALL_NOT_SUPPORTED*, falls in den Anmerkungen nichts anderes angegeben ist.

Tabelle der CPI-C 2.0 Aufrufe

CPI-C 2.0 Aufruf		OpenCPIC	Anmerkung
Accept_Conversation	(CMACCP)	*	(1)
Accept_Incoming	(CMACCI)	*	(1)
Allocate	(CMALLC)	+	
Cancel_Conversation	(CMCANC)	+	
Confirm	(CMCFM)	+	
Confirmed	(CMCFMD)	+	
Convert_Incoming	(CMCNVI)	*	(2)
Convert_Outgoing	(CMCNVO)	*	(2)
Deallocate	(CMDEAL)	+	
Deferred_Deallocate	(CMDFDE)	+	

CPI-C 2.0 Aufruf		OpenCPIC	Anmerkung
Extract_AE_Qualifier	(CMEAEQ)	+	
Extract_AP_Title	(CMEAPT)	+	
Extract_Application_Context_Name	(CMEACN)	+	
Extract_Conversation_State	(CMECS)	+	
Extract_Conversation_Type	(CMECT)	+	
Extract_Initialization_Data	(CMEID)	-	
Extract_Maximum_Buffer_Size	(CMEMBS)	+	
Extract_Mode_Name	(CMEMN)	*	(3)
Extract_Partner_LU_Name	(CMEPLN)	*	(4)
Extract_Secondary_Information	(CMESI)	-	
Extract_Security_User_ID	(CMESUI)	*	(5)
Extract_Send_Receive_Mode	(CMESRM)	-	
Extract_Sync_Level	(CMESL)	+	
Extract_TP_Name	(CMETPN)	+	
Extract_Transaction_Control	(CMETC)	+	
Flush	(CMFLUS)	+	
Include_Partner_In_Transaction	(CMINCL)	+	
Initialize_Conversation	(CMINIT)	+	
Initialize_For_Incoming	(CMINIC)	+	
Prepare	(CMPREP)	+	
Prepare_To_Receive	(CMPTR)	+	
Receive	(CMRCV)	+	
Receive_Expedited_Data	(CMRCVX)	-	
Release_Local_TP_Name	(CMRLTP)	+	
Request_To_Send	(CMRTS)	+	
Send_Data	(CMSEND)	+	
Send_Error	(CMSERR)	+	
Send_Expedited_Data	(CMSNDX)	-	
Set_AE_Qualifier	(CMSAEQ)	*	(6)(7)
Set_Allocate_Confirm	(CMSAC)	+	
Set_AP_Title	(CMAPT)	*	(6)(8)

CPI-C 2.0 Aufruf		OpenCPIC	Anmerkung
Set_Application_Context_Name	(CMSACN)	*	(6)
Set_Begin_Transaction	(CMSBT)	+	
Set_Confirmation_Urgency	(CMSCU)	+	
Set_Conversation_Security_Password	(CMSCP)	*	(5)
Set_Conversation_Security_Type	(CMSCST)	*	(5)
Set_Conversation_Security_User_ID	(CMSCSU)	*	(5)
Set_Conversation_Type	(CMSCT)	*	(9)
Set_Deallocate_Type	(CMSDT)	+	
Set_Error_Direction	(CMSED)	+	
Set_Fill	(CMSF)	+	
Set_Initialization_Data	(CMSID)	-	
Set_Join_Transaction	(CMSJT)	+	
Set_Log_Data	(CMSLD)	-	(10)
Set_Mode_Name	(CMSMN)	*	(3)
Set_Partner_LU_Name	(CMSPLN)	*	(4)
Set_Prepare_Data_Permitted	(CMSPDP)	+	
Set_Prepare_To_Receive_Type	(CMSPTR)	+	
Set_Processing_Mode	(CMSPM)	+	
Set_Queue_Callback_Function	(CMSQCF)	-	
Set_Queue_Processing_Mode	(CMSQCM)	-	
Set_Receive_Type	(CMSRT)	+	
Set_Return_Control	(CMSRT)	+	
Set_Send_Receive_Mode	(CMSSRM)	-	
Set_Send_Type	(CMSST)	+	
Set_Sync_Level	(CMSSL)	+	
Set_TP_Name	(CMSTPN)	+	
Set_Transaction_Control	(CMSTC)	+	
Specify_Local_TP_Name	(CMSLTP)	+	
Test_Request_To_Send_Received	(CMTRTS)	+	
Wait_For_Completion	(CMWCMP)	-	
Wait_For_Conversation	(CMWAIT)	+	

Anmerkungen

1. **Accept_Conversation(CMACCP), Accept_Incoming (CMACCI）**

   Prinzipiell erlaubt CPI-C, dass ein Anwendungsprogramm mehrere Conversations mittels *Accept_Conversation (CMACCP)* bzw. *Accept_Incoming (CMACCI)* akzeptiert und somit mehrere Conversations zu Superior-Partnern gleichzeitig unterhält. Dieser sogenannte “Multiple Accept” wird auch von OpenCPIC unterstützt, jedoch mit folgender Einschränkung: Da ein OpenCPIC-Anwendungsprogramm zu einem Zeitpunkt immer nur an einer Transaktion beteiligt sein kann (d. h. alle aktiven Protected Conversations eines AP gehören zur selben globalen Transaktion) und jeder Knoten in einem Transaktionsbaum immer genau einen Superior besitzt (mit Ausnahme des Root-Knotens), darf ein OpenCPIC-Anwendungsprogramm maximal eine Protected Conversation zu einem Superior unterhalten.
   Zum anderen kann ein Transaktionsbaum nur nach unten, d. h. durch Hinzufügen weiterer Subordinates erweitert werden. Das heißt, ein an einer Transaktion beteiligtes AP darf, auch wenn es noch keinen Superior hat, keine Conversation von einem Superior akzeptieren.

   In OpenCPIC wird daher ein *CMACCP* bzw. *CMACCI* mit dem Returnwert *CM_PRODUCT_SPECIFIC_ERROR* beantwortet, wenn wenigstens eine der folgenden Bedingungen wahr ist:
   – Das AP hat mindestens eine aktive Conversation mit dem *sync_level* C*M_SYNC_POINT* oder *CM_SYNC_POINT_NO_CONFIRM*.
   – Das AP hat bereits *tx_begin()* aufgerufen und befindet sich damit in einer Transaktion.

   Im Falle von *CMACCI* ändert sich der Zustand der Conversation dadurch nicht, sondern behält den Wert *CM_INITIALIZE_STATE*.

2. **Convert_Incoming (CMCNVI), Convert_Outgoing (CMCNVO)**

   Diese Aufrufe führen eine Code-Konvertierung von EBCDIC nach ASCII (*CMCNVI*) bzw. von ASCII nach EBCDIC (*CMCNVO*) durch. Die Konvertierung erfolgt anhand der Tabellen aus dem IBM-Dokument “NetView/PC Application Programming, Version1.2” [33]. Die Zeichensätze in diesen Tabellen stimmen mit den in CPI-C 2.0 definierten Zeichensätzen überein. Die Konvertierungstabellen sind im Abschnitt „Code-Konvertierungstabellen“ auf Seite 200 abgedruckt.

3. **Extract_Mode_Name (CMEMN), Set_Mode_Name (CMSMN）**

   OSI-TP und somit auch XAP-TP unterstützen keine Mode-Namen. Das vergleichbare XAP-TP-Attribut “Quality-Of-Service” (QOS) wird in der OpenCPIC zugrundeliegenden XAP-TP-Implementierung nicht unterstützt, da die OSS-Schnittstelle den QOS nicht beinhaltet. Folglich ist die Definition von Mode-Namen in OpenCPIC-Anwendungspro-

grammen nicht sinnvoll. *CMSMN* akzeptiert zwar alle syntaktisch richtigen Mode-Namen, jedoch ist der Aufruf wirkungslos. *CMEMN* liefert immer einen Standard-Mode-Namen, bestehend aus acht Leerzeichen, zurück.

4. **Extract_Partner_LU_Name (CMEPLN), Set_Partner_LU_Name (CMSPLN)**

   Diese Aufrufe sind normalerweise nur bei einem LU6.2-CRM von Bedeutung. Für die Adressierung der Partneranwendung in OSI-TP stellt CPI-C 2.0 die Aufrufe *Set_AP_Title (CMSAPT)*, *Set_AE_Qualifier (CMSAEQ)* und *Set_Application_Context_Name (CMSACN)* bzw. *Extract_AP_Title (CMEAPT)*, *Extract_AE_Qualifier (CMEAEQ)* und *Extract_Application_Context_Name (CMEACN)* zur Verfügung. Da die Aufrufe *Set_Partner_LU_Name (CMSPLN)* bzw. *Extract_Partner_LU_Name (CMEPLN)* jedoch in der Vorgängerversion OpenCPIC V2.0 für die symbolische Adressierung benutzt wurden, werden sie in der aktuellen OpenCPIC-Version aus Kompatibilitätsgründen weiterhin unterstützt. In OpenCPIC entspricht der *partner_LU_name* in der CPI-C Side Information dem symbolischen Namen der Partneranwendung, wie er mit der PARTAPPL-Anweisung generiert wird (siehe Seite 34). Er ist, abweichend von der CPI-C-Beschreibung, maximal 17 Zeichen lang.

   Die in OpenCPIC unterstützte Methode der symbolischen Adressierung mittels *CMEPLN* und *CMSPLN* ist u. U. nicht mit anderen auf OSI-TP basierenden Anwendungen kompatibel. Es wird daher empfohlen, in neu entwickelten Anwendungsprogrammen die Partneranwendung stets mit *CMSAPT*, *CMSAEQ* und *CMSACN* zu adressieren, um die Portabilität zu gewährleisten.

5. **Extract_Security_User_ID (CMESUI), Set_Conversation_Security_Password (CMSCSP), Set_Conversation_Security_Type (CMSCST), Set_Conversation_Security_User_ID (CMCSU)**

   Das OSI-TP-Protokoll unterstützt keine Conversation Security im Sinne von CPI-C. In der offiziellen X/Open CPI-C-Schnittstelle sind deshalb die Aufrufe *Set_Conversation_Security_Type (CMSCST)*, *Set_Conversation_Security_User_ID (CMSCSU)*, *Set_Conversation_Security_Password (CMSCSP)* und *Extract_Security_User_ID (CMESUI)* nur für LU6.2 CRM beschrieben. Abweichend von diesem Standard realisiert OpenCPIC ein internes Protokoll zum Austausch von Zugangsschutzinformationen zwischen OpenCPIC und UTM-Anwendungen und bietet die genannten CPI-C Aufrufe als Schnittstelle für die Verwaltung der Security-Daten an. Ihre Verwendung ist jedoch nur mit einer UTM-Anwendung als Kommunikationspartner sinnvoll. Bei der Generierung der Partneranwendung mit der PARTAPPL-Anweisung muss dazu der Parameter APPL-CONTEXT = utm-secu gesetzt werden.

   Die Zugangsschutzinformationen werden beim Dialogaufbau im *TP-BEGIN-DIALOGUE-request* als Benutzerdaten (Initialization Data) mitgeschickt. OpenCPIC wertet jedoch keine bei einem Dialogaufbauwunsch empfangenen Zugangsschutzinformationen aus.

6. **Extract_AE_Qualifier (CMEAEQ),**
   Extract_AP_Title (CMEAPT),
   Extract_Application_Context_Name (CMEACN),
   Set_AE_Qualifier (CMSAEQ),
   Set_AP_Title (CMSAPT),
   Set_Application_Context_Name (CMSACN)

   In OSI-TP wird eine Partneranwendung durch den Application Process Title (APT) und den Application Entity Qualifier (AEQ) eindeutig adressiert. Zusätzlich muss für jede Partneranwendung ein Application Context vereinbart werden. X/Open CPI-C definiert für diese Adressierungsparameter je ein Feld in der CPI-C Side Information (siehe Seite 79). Damit können *AP_title*, *AE_qualifier* und *application_context_name* durch Definition eines Symbolic Destination Name (SYMDEST-Anweisung) festgelegt werden, der beim CMINIT als Parameter *symdest_name* übergeben wird.

   Mit den Funktionen *Set_AE_Qualifier (CMSAEQ)*, *Set_AP_Title (CMSAPT)* und *Set_Application_Context_Name (CMSACN)* kann der Eintrag aus der Side Information geändert bzw. eine Partneranwendung ohne Benutzung eines Symbolic Destination Name (Übergabe eines leeren *symdest_name* an *CMINIT*) adressiert werden. Dabei ist jedoch zu beachten, dass in OpenCPIC jede Partneranwendung durch eine PARTAPPL-Anweisung in der Generierungsdatei definiert sein muss. Das heißt, mit der durch *CMSAEQ*, *CMSAPT* und *CMSACN* übergebenen Adressinformation muss eine Partneranwendung aus der Generierungsdatei eindeutig identifizierbar sein.
   Im “Abschnitt „SYMDEST - Kommunikationspartner definieren“”, Anmerkung 1 auf Seite 45 ist beschrieben, wie der OpenCPIC-Manager die Prüfung der Adressinformation vornimmt.

   Mit den CPI-C-Aufrufen *Extract_AP_title (CMEAPT)*, *Extract_AE_Qualifier (CMEAEQ)* und *Extract_Application_Context_Name (CMEACN)* können die aktuell gültigen Werte aus der Side Information abgefragt werden.

7. **Set_AE_Qualifier (CMSAEQ)**

   OpenCPIC schreibt für den Parameter *AE_qualifier_format* den Wert *CM_INT_DIGITS* fest vor. Der Parameter *AE_qualifier* muss einen ganzzahligen Wert zwischen 1 und 67 108 863 besitzen. Wird ein ungültiger Wert oder das Format *CM_DN* übergeben, kehrt der Aufruf mit dem Returnwert *CM_PRODUCT_SPECIFIC_ERROR* zurück.

8. **Set_AP_Title (CMSAPT)**

   OpenCPIC schreibt für den Parameter *AP_title_format* den Wert *CM_OID* fest vor. Der Parameter *AP_title* muss einen Wert vom Typ Objektbezeichner besitzen. Wird ein ungültiger Wert oder das Format *CM_DN* übergeben, kehrt der Aufruf mit dem Returnwert *CM_PRODUCT_SPECIFIC_ERROR* zurück.

9. Set_Conversation_Type (CMSCT)

OpenCPIC erlaubt als Parameterwert für *conversation_type* nur den Standardwert *CM_MAPPED_CONVERSATION*.

Aufrufe mit *conversation_type = CM_BASIC_CONVERSATION* werden mit dem Returnwert *CM_PRODUCT_SPECIFIC_ERROR* beantwortet.

10. Set_Log_Data (CMSLD)

OpenCPIC unterstützt keine Log-Daten. Dieser Aufruf liefert daher immer den Returnwert *CM_PRODUCT_SPECIFIC_ERROR*.

5.1.3 Die TX-Schnittstelle von OpenCPIC

OpenCPIC realisiert alle in der X/Open Schnittstelle TX definierten Funktionen. Es wird daher an dieser Stelle auf die X/Open Dokumentation [26] verwiesen.

Tabelle der TX-Aufrufe (C)

```
int tx_begin(void)
int tx_close(void)
int tx_commit
int tx_info(TXINFO *info)
int tx_open(void)
int tx_rollback(void)
int tx_set_commit_return(COMMIT_RETURN when_return)
int tx_set_transaction_control(TRANSACTION_CONTROL control)
int tx_set_transaction_timeout(TRANSACTION_TIMEOUT timeout)
```

Tabelle der TX-Aufrufe (COBOL)

```
CALL "TXBEGIN" USING TX-RETURN-STATUS
CALL "TXCLOSE" USING TX-RETURN-STATUS
CALL "TXCOMMIT" USING TX-RETURN-STATUS
CALL "TXINFORM" USING TX-INFO-AREA TX-RETURN-STATUS
CALL "TXOPEN" USING TX-RETURN-STATUS
CALL "TXROLLBACK" USING TX-RETURN-STATUS
CALL "TXSETCOMMITRET" USING TX-INFO-AREA TX-RETURN-STATUS
CALL "TXSETTIMEOUT" USING TX-INFO-AREA TX-RETURN-STATUS
CALL "TXSETTRANCTL" USING TX-INFO-AREA TX-RETURN-STATUS
```

5.2 Anwendungsprogramme binden

Um ein lauffähiges Anwendungsprogramm zu erstellen, müssen Sie die Objektmodule Ihres Anwendungsprogramms mit der Bibliothek *libocpic.a* binden.

OpenCPIC-Programme können Sie in den Programmiersprachen C/C++ oder COBOL erstellen.

Für die Programmiersprache C/C++ bzw. COBOL wird ein Include bzw. COBOL COPY zur Programmierunterstützung ausgeliefert, siehe im openUTM-Handbuch „Anwendungen erstellen mit X/Open-Schnittstellen“.

Gehen Sie wie folgt vor:

1. Übersetzen Sie Ihre OpenCPIC Programme mit dem C/C++- bzw. COBOL-Compiler und geben Sie dabei die notwendigen Compiler-Optionen an.
2. Binden Sie Ihr ausführbares OpenCPIC Anwendungsprogramm mit folgenden Bestandteilen:
   - Objekte Ihrer OpenCPIC-Programme
   - OpenCPIC-Bibliothek unter *openCPIC-pfad/libocpic.a*
   - OpenCPIC XA-Bibliothek unter *openCPIC-pfad/xa/ocpic_xa_connect*

     Damit steuern Sie, ob das Anwendungsprogramm auf einen lokalen Resource Manager zugreifen kann oder nicht. Details dazu siehe Abschnitt „Lokale Resource Manager - XA-Anschluss“ auf Seite 141.
   - weitere benötigte Bibliotheken

Falls auf Ihrem System mehrere Resource Manager installiert sind, ist unbedingt darauf zu achten, dass der OpenCPIC-Manager mit demselben Resource Manager arbeitet wie die lokalen Anwendungsprogramme. Das heißt, die Programmdatei *ocpic_mgr* und die lokalen Anwendungsprogramme müssen denselben XA-Anschluss verwenden und denselben Resource Manager referenzieren.

5.3 Anwendungsprogramme starten

Vor dem Start eines OpenCPIC-Anwendungsprogramms muss der OpenCPIC-Manager gestartet worden sein.

Anwendungsprogramme können Sie entweder durch ein Operatorkommando selbst starten oder durch den OpenCPIC-Manager bei einem ankommenden Dialogwunsch automatisch starten lassen.

Der automatische Start ist nur sinnvoll für Anwendungsprogramme, die mit *Accept_Conversation (CMACCP)* oder *Accept_Incoming (CMACCI)* Dialogaufbauwünsche vom Partnerprogramm akzeptieren. Dies sind i. a. Anwendungsprogramme, die in einem Client-Server-System eine Server-Funktion übernehmen und nur bei aktuellen Client-Anforderungen gestartet werden sollen.

Alle Anwendungsprogramme, die vom OpenCPIC-Manager gestartet werden sollen, müssen in der Generierungsdatei mit einer LOCAPRO-Anweisung definiert sein (siehe Seite 39).

5.3.1 Umgebungsvariablen

Vor dem Start eines OpenCPIC-Anwendungsprogramms sind ggf. die erforderlichen Umgebungsvariablen zu setzen. Automatisch gestartete Programme übernehmen die Umgebungsvariablen aus der Arbeitsumgebung des OpenCPIC-Managers. In diesem Fall müssen also bereits vor dem Start des OpenCPIC-Managers alle erforderlichen Umgebungsvariablen gesetzt werden.

– Die für die Kommunikation des Anwendungsprogramms mit dem OpenCPIC-Manager benötigten Named Pipes werden im Verzeichnis *$OCPICDIR* erzeugt. Der Inhalt der Variablen *OCPICDIR* in der Umgebung des Anwendungsprogramms muss deshalb mit dem Inhalt von *OCPICDIR* für den OpenCPIC-Manager übereinstimmen. Ist *OCPICDIR* nicht gesetzt, verwenden sowohl der OpenCPIC-Manager als auch die Bibliothek *libocpic.a* den Standardwert *openCPIC-pfad*.

– Mit den Umgebungsvariablen *CPICTRACE*, *CPICPATH*, *CPICSIZE*, *OCP_TRACELIMIT* wird der CPI-C-Trace gesteuert (siehe “Abschnitt „CPI-C-Trace aktivieren“ auf Seite 90“).

– Weiterhin sind alle von der XA-Bibliothek Ihres Resource Managers geforderten Umgebungsvariablen zu setzen.

Arbeitsumgebung automatisch gestarteter Anwendungsprogramme

Der OpenCPIC-Manager startet ein Anwendungsprogramm mit dem Systemaufruf *fork()*. Das Anwendungsprogramm erhält dabei folgende Arbeitsumgebung:

– Benutzerkennung und Gruppe ergeben sich aus dem LOGIN-Parameter der LOCAPRO-Anweisung in der Generierungsdatei.

– Das Arbeitsverzeichnis des Anwendungsprogramms ist das HOME-Verzeichnis der Benutzerkennung (LOGIN-Parameter).

– Alle File-Deskriptoren (einschließlich *stdin*, *stdout* und *stderr*) werden bei der Prozesserzeugung durch *fork()* geschlossen.

– Die Umgebungsvariablen aus der Arbeitsumgebung des OpenCPIC-Managers (z. B. *OCPICDIR*, *OCP_TRACELIMIT*, *CPICTRACE*) werden übernommen.
Die Umgebungsvariablen *USER*, *LOGNAME*, *HOME* und *PATH* werden entsprechend dem LOGIN-Parameter der LOCAPRO-Anweisung gesetzt.

5.3.2 CPI-C-Trace aktivieren

Zu Diagnosezwecken können Sie den Trace der Bibliothek *libocpic.a* (CPI-C-Trace) einschalten. Damit bewirken Sie, dass *libocpic.a* während des Programmablaufs Traceinformationen mitprotokolliert. Der CPI-C-Trace arbeitet in zwei Stufen (Trace-Level): Schnittstellen-Trace (Trace-Level 1) und interner Trace (Trace-Level 2). Zur Fehlerdiagnose in Anwendungsprogrammen ist im allgemeinen der Schnittstellen-Trace ausreichend.

Die Protokolldateien des CPI-C-Trace werden im aktuellen Arbeitsverzeichnis bzw. in dem mit der Umgebungsvariable *CPICTRACE* (siehe nachfolgenden Abschnitt “CPI-C-Trace steuern”) festgelegten Verzeichnis abgelegt. Der Name einer Protokolldatei setzt sich wie folgt zusammen:

OCPICpid.suff

suff und *pid* sind Platzhalter mit der folgenden Bedeutung:

suff	ganzzahliger Dateinamen-Suffix Der Name einer Protokolldatei wird um einen ganzzahligen Suffix ergänzt, der, beginnend mit 0, bei jedem Dateiwechsel um eins hochgezählt wird. Ein Dateiwechsel erfolgt automatisch beim Überschreiten einer bestimmten Maximalgröße, deren Wert über die Umgebungsvariable *CPICTRACE* (siehe nachfolgenden Abschnitt “CPI-C-Trace steuern”) gesteuert wird. Es entsteht so eine Folge von Protokolldateien, wobei der Suffix *suff* deren Reihenfolge angibt.
pid	Prozess-Id des Anwendungsprogramms

CPI-C-Trace steuern

Die Steuerung des CPI-C-Trace erfolgt über Umgebungsvariablen. Folgende Umgebungsvariablen werden durch die Bibliothek *libocpic.a* ausgewertet:

CPICTRACE Diese Umgebungsvariable muss gesetzt werden, um den CPI-C-Trace zu aktivieren. Es sind zwei Stufen, sogenannte Trace-Level, möglich:

=-[s|S] Aktiviert den sogenannten Schnittstellen-Trace (Trace Level 1). Dieser Trace schreibt alle CPI-C und TX-Aufrufe mit ihren Parametern und Returnwerten in die Protokolldatei. Er eignet sich besonders zur Fehlerdiagnose in Anwendungsprogrammen.

=-[i| I] Aktiviert den internen Fehler-Trace (Trace-Level 2).
Dieser Trace beinhaltet den Schnittstellen-Trace und schreibt zusätzlich zahlreiche interne Informationen der Bibliothek in die Protokolldatei. Sie sollten ihn nur einschalten, wenn ein Verdacht auf Fehlverhalten von OpenCPIC vorliegt.

CPICPATH=pathname
gibt das Verzeichnis an, in welchem die Protokolldateien des CPI-C-Trace abgelegt werden sollen. Ist *CPICPATH* nicht gesetzt, werden die Protokolldateien im aktuellen Arbeitsverzeichnis des Anwendungsprogramms abgelegt.
Beachten Sie bitte: Ist *pathname* kein gültiges Verzeichnis auf Ihrem System oder können die Protokolldateien im angegebenen Verzeichnis nicht erzeugt werden, bleibt der Trace ausgeschaltet.

CPICSIZE=size gibt die maximale Dateigröße der Protokolldateien (in Byte) an. Sobald die Protokolldatei *OCPIC<pid>.suff* diese Größe überschritten hat, wird automatisch zur Datei *OCPIC<pid>.suff+1* gewechselt.
Wenn *CPICSIZE* nicht gesetzt oder *size* kein ganzzahliger Wert ist, wird als Standardeinstellung der Wert 524288 Byte (512 KByte) angenommen.

OCP_TRACELIMIT=limit
gibt die maximale Anzahl der Protokolldateien an, die vom CPI-C-Trace angelegt werden. *limit* ist eine ganze Zahl zwischen 0 und 1024.
Standardmäßig werden maximal zehn Protokolldateien angelegt. Nachdem die Protokolldatei mit dem Suffix 9 (*OCPIC<pid>.9*) geschlossen wurde, beginnt die Zählung wieder mit 0 und in der Folge

werden bereits vorhandene Protokolldateien überschrieben.
Die Anzahl der maximal gehaltenen Protokolldateien können Sie ändern, indem Sie die Umgebungsvariable *OCP_TRACELIMIT* vor dem Start des Anwendungsprogramms auf den gewünschten Wert setzen. Es werden jedoch immer mindestens zwei Protokolldateien angelegt. Setzen Sie *OCP_TRACELIMIT* auf den Wert 0, bedeutet dies, dass es keine Obergrenze für die Anzahl der Protokolldateien gibt.

Protokolldateien aufbereiten

Die Protokolldateien des CPI-C-Trace (*OCPIC<pid>.suff*) werten Sie mit folgendem Kommando aus:

Syntax

ocpic_tredit tracefile

Beschreibung

tracefile — ist der Pfadname der binären Protokolldatei, die aufbereitet werden soll, z. B. *$HOME/OCPIC1244.0*

Die aufbereiteten Daten werden auf die Standardausgabe (*stdout*) ausgegeben und können in eine Datei umgeleitet werden.

6 XATMI-Client-Programme erstellen

XATMI (X/Open Application Transaction Manager Interface) ist eine von X/Open standardisierte Programmschnittstelle für einen Communication Resource Manager, der Client-Server-Kommunikation mit (optionalem) Einsatz von Transaktionsfunktionen ermöglicht.

Die in OpenCPIC realisierte XATMI-Bibliothek setzt auf der CPI-C-Schnittstelle auf. Das heißt, XATMI-Programme, die XATMI-Aufrufe verwenden, müssen außer mit der XATMI-Bibliothek auch mit der CPI-C-Bibliothek von OpenCPIC (*libocpic.a*) gebunden sein.

Grundlage der XATMI-Programmschnittstelle für OpenCPIC ist die X/Open XATMI-Spezifikation [28]. Dieses Kapitel beschreibt die XATMI-Schnittstelle für Clients mit dem Trägersystem OpenCPIC. Die Kenntnis dieser Spezifikation wird im Folgenden vorausgesetzt. In diesem Kapitel werden folgende Begriffe verwendet:

Client
: Eine Anwendung, die Service-Funktionen aufruft.

Server
: Eine UTM-Anwendung, die Service-Funktionen in C und/oder in COBOL enthält. Die Service-Funktionen können aus mehreren Teilprogrammen bestehen.

Service
: Eine Service-Funktion, die entsprechend der XATMI-Spezifikation in C oder COBOL programmiert ist. XATMI unterscheidet zwei Arten von Services: End-Services und Intermediate-Services.

  Ein End-Service ist nur mit seinem Client verbunden und ruft keine anderen Services auf.

  Ein Intermediate-Service ruft einen oder mehrere weitere Services auf.

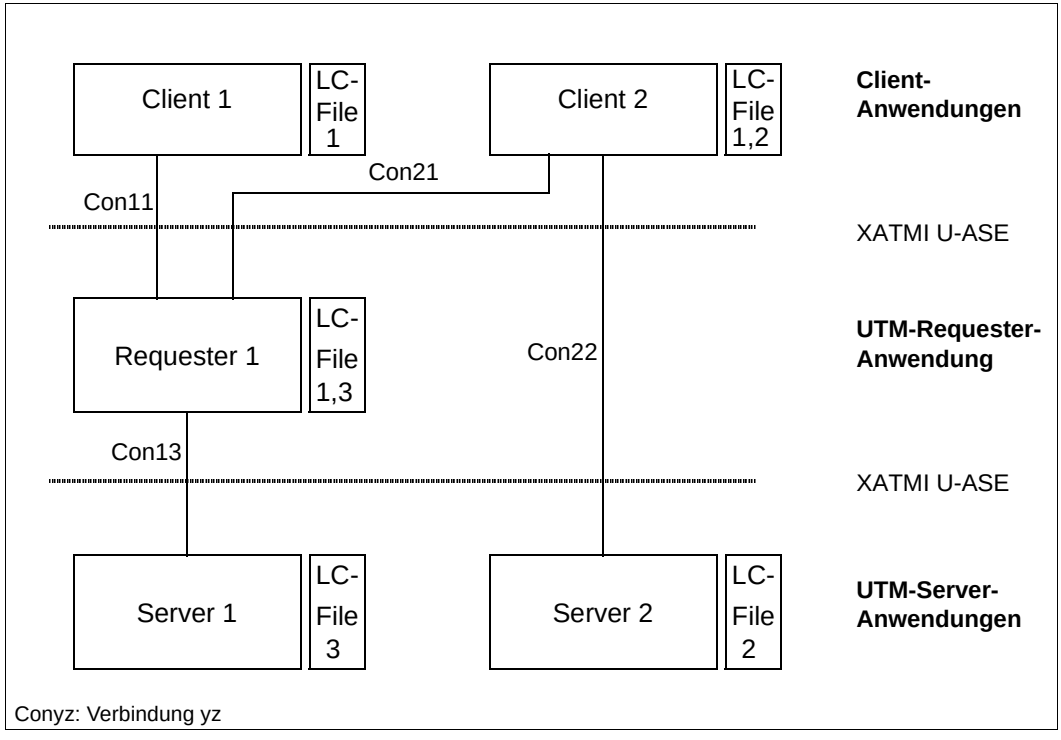
Request
: Ein Request ist ein Aufruf eines Services. Dieser kann von einem Client wie auch von einem Intermediate-Service aus erfolgen.

Requester
: Die XATMI-Spezifikation verwendet den Begriff "Requester" als Bezeichnung für jegliche Anwendung, die einen Service aufruft. Diese kann sowohl Client wie auch Server sein.

Typisierte Puffer
: Puffer für den Austausch von typisierten und strukturierten Daten zwischen Kommunikationspartnern. Durch diese typisierten Puffer ist die Struktur der ausgetauschten Daten dem Trägersystem und der Anwendung implizit bekannt und wird auch im heterogenen Verbund automatisch angepasst (kodiert, dekodiert).

6.1 Client-Server-Anwendungsverbund

Das folgende Bild zeigt einen Client-Server-Anwendungsverbund, bei dem Clients, Server und Requester miteinander kommunizieren. Sie tauschen ihre typisierten Datenstrukturen (Typisierte Puffer) nach dem Protokoll der "XATMI U-ASE Definition" aus.



In einem beliebigen, heterogenen Anwendungsverbund muss sowohl den Servern wie auch den Clients eine Local Configuration beigestellt sein, die jeweils in der Local Configuration File (LCF) definiert ist. Die Local Configuration beschreibt jeweils die Services und ihre zugehörigen Datenstrukturen, d. h.:

- bei einem Server alle aufrufbaren Services
- bei einem Client die Services aller Server, mit denen der Client in Verbindung steht.
- bei einem Requester (Intermediate-Service) sowohl alle bereitgestellten Services als auch alle benutzten Sevices.

Die Local Configurations aller beteiligten Anwendungen müssen aufeinander abgestimmt sein.

Um Client-Server-Verbindungen Con11, Con13,. .. abzuwickeln, stehen mehrere Kommunikationsmodelle zur Verfügung.

6.1.1 Default-Server

Zur Vereinfachung der Client-Server-Konfiguration bietet Ihnen openUTM die Möglichkeit, mit der Angabe DEST=.DEFAULT in der SVCU-Anweisung der Local Configuration File einen Default-Server zu vereinbaren.

Falls bei den Aufrufen *tpcall*, *tpacall* oder *tpconnect* ein Service *svcname2* verwendet wird, der keinen SVCU-Eintrag in der Local Configuration File besitzt, wird automatisch folgender Eintrag verwendet:

```
SVCU svcname2, RSN=svcname2, TAC=scvname2, DEST=.DEFAULT, MODE=RR
```

Die OpenCPIC-Generierung benötigt dafür folgenden Eintrag in der Generierungsdatei:

```
SYMDEST .DEFAULT
PARTNER-APPL=...
PARTNER-APRO=...
```

6.2 Kommunikationsmodelle

Für die Client-Server-Kommunikation stehen dem Programmierer drei Kommunikationsmodelle zur Auswahl:

– Synchrones Request-Response Modell: Einschritt- Dialog.
Der Client ist nach dem Senden der Service-Anforderung bis zum Eintreffen der Antwort blockiert.

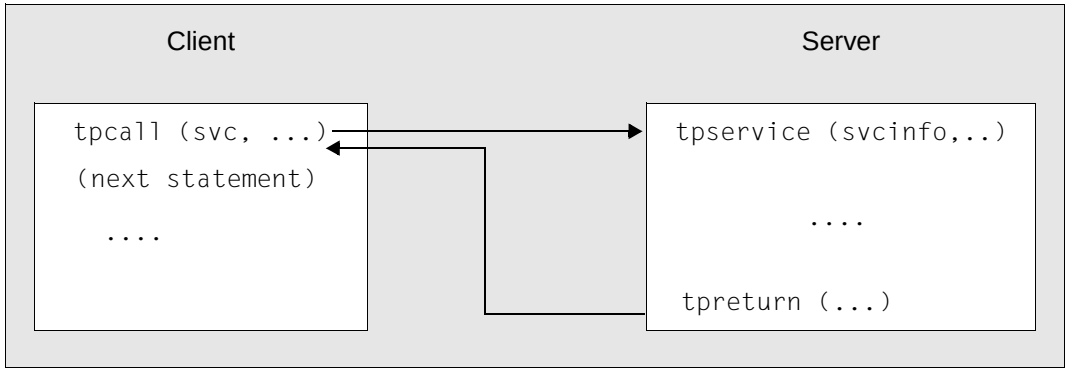
– Asynchrones Request- Response Modell: Einschritt-Dialog.
Der Client ist nach dem Senden der Service-Anforderung nicht blockiert.

– Conversational Modell: Mehrschritt-Dialog.
Client und Server können beliebig Daten austauschen.

Die für diese Kommunikationsmodelle notwendigen XATMI-Funktionen werden im Folgenden nur skizziert, dabei wird die C-Notation verwendet. Die genaue Beschreibung der XATMI-Funktionen finden Sie in der XATMI-Spezifikation [28].

6.2.1 Synchrones Request-Response-Modell

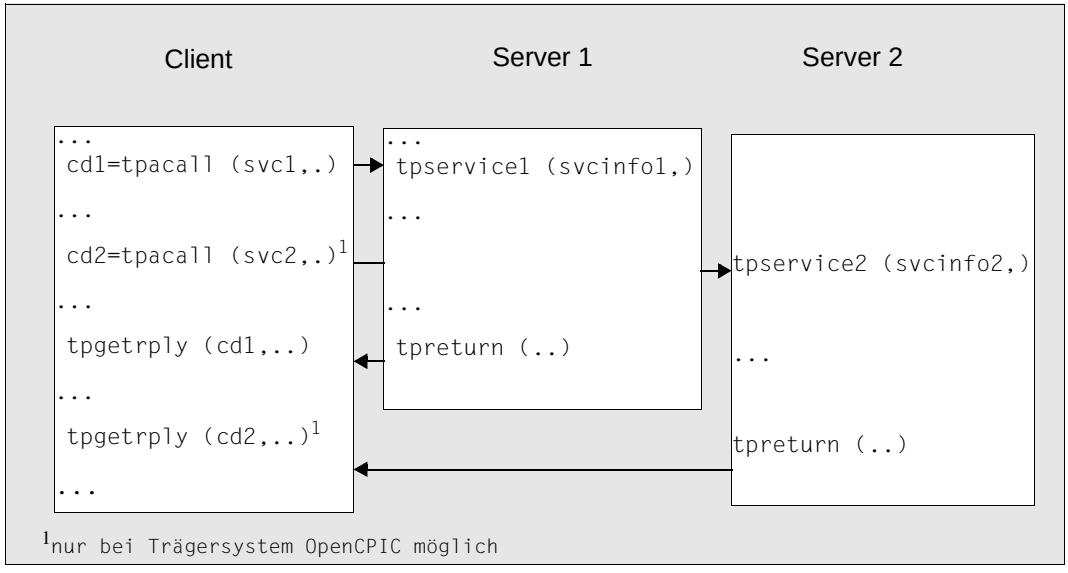
Ein solcher Service wird mit einer *tpcall*-Funktion aufgerufen. Ein *tpcall*-Aufruf adressiert den Service, schickt genau eine Nachricht an diesen ab und wartet solange, bis ihn die Antwort erreicht, d. h. *tpcall* wirkt blockierend.



Dabei bezeichnet *svc* den intern verwendeten Namen des Services, *svcinfo* die Service-Info-Struktur mit dem Service-Namen und *tpservice* den Programmnamen der Service-Routine. Für die Programmiersprache C steht dafür das Template *tpservice()* zur Verfügung. Bei COBOL muss der XATMI-Aufruf TPSVCSTART verwendet werden. Die Service-Info-Struktur ist in der XATMI-Schnittstelle definiert und wird intern durch XATMI versorgt.

6.2.2 Asynchrones Request-Response-Modell

Bei diesem Modell wird die Kommunikation in zwei Schritten abgewickelt. Mit dem ersten Aufruf *tpacall*) wird der Service addressiert und die Nachricht verschickt, mit dem zweiten Aufruf (*tpgetrply*) zu einem späteren Zeitpunkt wird die Antwort abgeholt. Bei diesem Modell können parallel mehrere Services beauftragt werden. Das folgende Bild veranschaulicht den Sachverhalt.



Dabei bezeichnen *svc1*, *svc2* die intern verwendeten Namen der Services , *cd1*, *cd2* die prozesslokalen Communication Descriptoren, *svcinfo1*, *svcinfo2* die Service-Info-Strukturen mit den Service-Namen und *tpservice1, tpservice2* die Programmnamen der Service-Routinen.

tpacall ist nicht blockierend, d. h. der Client kann in der Zwischenzeit weitere lokale Verarbeitungen durchführen.

tpgetrply hingegegen ist blockierend, d. h. der Client wartet solange, bis die Antwort eingetroffen ist.

Bei diesem Modell muss auf der UTM-Server-Seite für den Service ein Dialog-TAC generiert sein (wie beim synchronen Request-Response-Modell).

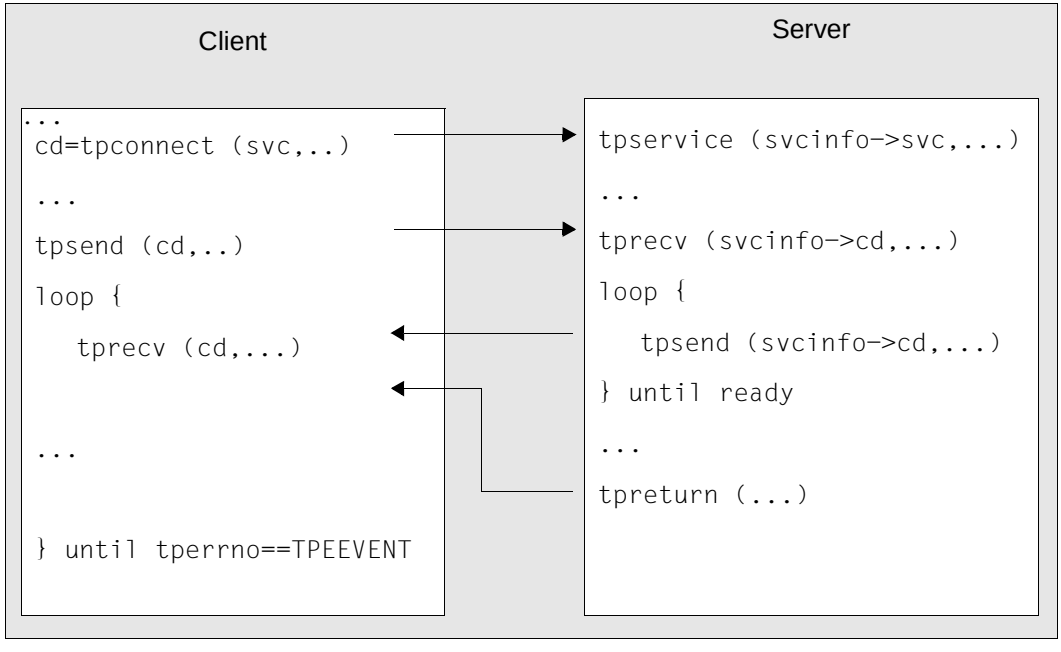
OpenCPIC erlaubt das Ausführen paralleler Aufträge.

6.2.3 Conversational Modell

Für verbindungsorientiertes Arbeiten ("Conversation") bietet XATMI das Conversational Modell an.

Dieses Modell kann z. B. verwendet werden, um große Datenmengen in mehreren Teilschritten zu übertragen. Beim synchronen Aufruf *tpcall* kann es in solchen Fällen wegen der Begrenzung für lokale Datenpuffer zu Problemen kommen.

Beim Conversational Modell wird die Conversation zu einem Service explizit mit dem Aufruf *tpconnect* aufgebaut. Solange sie besteht, können Client und Server mit *tpsend* und *tprecv* Daten austauschen. Beendet wird die Conversation, wenn der Server mit *tpreturn* das Ende signalisiert; der Client erhält dann beim *tprecv* in der Variable *tperrno* einen entsprechenden Code. Daher muss das Client-Programm mindestens einen *tprecv*-Aufruf enthalten.



Dabei bezeichnet *svc* den lokalen Namen des Services, *cd* den prozesslokalen Communication Descriptor, *tpservice* den Programmnamen der Service-Routine und *svcinfo* die Service-Info-Struktur mit dem Service-Namen und dem Communication Descriptor.

Bei diesem Modell muss auf der UTM-Server-Seite für den Service ein Dialog-TAC generiert sein.

In Fehlerfällen kann der Client den Abbruch einer Conversation mit dem Aufruf *tpdiscon* erzwingen.

6.3 Typisierte Puffer

XATMI-Anwendungen tauschen Nachrichten mit Hilfe von "typisierten Datenpuffern" aus. Dadurch werden die über das Netz gehenden Daten korrekt an die Anwendung übergeben, d. h. entprechend der Datenstruktur mit ihren Datentypen, die sich hinter dem Puffernamen verbirgt.

Dies hat den Vorteil, dass die Anwendung keine Maschinenabhängigkeiten berücksichtigen muss wie z. B. Big Endian/Little Endian Darstellung, ASCII/EBCDIC-Konvertierung oder Ausrichtung auf Wortgrenzen. Damit können Datentypen wie *int*, *long*, *float* usw. als solche übertragen werden. Eine eventuell notwendige Kodierung/Dekodierung durch das Programm entfällt, da dies von XATMI übernommen wird (gemäß den Regeln der XATMI U-ASE Definition).

Ein Datenpuffer-Objekt besteht aus vier Komponenten:

– Typ: Definiert die Klasse des Puffers. Es gibt drei Klassen.

– Subtyp: Definiert das Objekt der Klasse, d. h. die eigentliche Datenstruktur.

– Längenangabe

– Dateninhalt

Ein solcher Datenpuffer wird während der Laufzeit erzeugt und kann dann über seinen Variablen-Namen (=Subtyp-Namen) angesprochen werden. In C-Programmen werden solche Puffer dynamisch mit *tpallcoc* erzeugt, man spricht dann von "typisierten Puffern", in Cobol-Programmen sind diese Puffer statisch festgelegt, man spricht von "typisierten Records".

Typen

Mit dem Typ eines Datenpuffers wird festgelegt, welche elementaren programmiersprachenabhängigen Datentypen erlaubt sind. Dadurch wird ein gemeinsames Datenverständnis in einem heterogenen Client-Server-Verbund ermöglicht.
Bei XATMI sind drei Typen definiert:

X_OCTET	Untypisierter Datenstrom von Bytes ("Userbuffer"). Dieser Typ besitzt keine Subtypen. Es wird keine Konvertierung vorgenommen
X_COMMON	Alle von C und COBOL gemeinsam verwendbaren Datentypen. Die Konvertierung wird von XATMI vorgenommen.
X_C_TYPE	Alle elementaren C-Datentypen mit Ausnahme von Zeigern. Die Konvertierung wird von XATMI vorgenommen.

Subtypen

Subtypen haben einen bis zu 16 Zeichen langen Namen, unter dem sie im Anwendungsprogramm angesprochen werden. Jedem Subtyp ist eine Datenstruktur (C-Structure oder COBOL-Record) zugeordnet, die die Syntax des Subtyps bestimmt, siehe Seite 107.

Die Zuordnung zwischen Datenstrukturen, Subtypen und gewünschten Services wird in der Local Configuration festgelegt, siehe Seite 111.

Die folgende Tabelle gibt einen Überblick über die elementaren ASN.1-Datentypen (Basistypen, siehe ISO 8824 [29]) und deren Codes:

Code[1]	**Bedeutung**	**ASN.1-Typ**	**X_C_TYPE**	**X_COMMON**
s	short integer	INTEGER	short	S9(4) COMP-5
S<n>	short integer array	SEQUENCE OF INTEGER	short[n]	S9(4) COMP-5 ...
i	integer	INTEGER	integer	--[2]
I<n>	integer array	SEQUENCE OF INTEGER	integer[n]	--
l	long integer	INTEGER	long	S9(9) COMP-5
L<n>	long integer array	SEQUENCE OF INTEGER	long[n]	S9(9) COMP-5 ...
f	float	REAL	float	--
F<n>	float array	SEQUENCE OF REAL	float[n]	--
d	double	REAL	double	--
D<n>	double array	SEQUENCE OF REAL	double[n]	--
c	character	OCTET STRING	char	PIC X
t	character	T.61-String	char	PIC X
C<n>	character array: Alle Werte von 0 bis 255 (dezimal)	OCTET STRING	char[n]	PIC X(n)
C!<n>	character array, durch Null ('\0') terminiert	OCTET STRING	char[n]	--
C<m>:<n>	character matrix[3]	SEQUENCE OF OCTET STRING	char [m][n]	--
C!<m>:<n>	character matrix, durch Null ('\0') terminiert	SEQUENCE OF OCTET STRING	char [m][n]	--
T<n>	Die abdruckbaren Zeichen A-Z, a-z und 0-9 plus[4] eine Reihe von Sonderzeichen und Steuerzeichen.	T.61-String	t61str[n]	PIC X(n)
T!<n>	character array, durch Null ('\0') terminiert	T.61-String	t61str[n]	--

Code[1]	Bedeutung	ASN.1-Typ	X_C_TYPE	X_COMMON
T<m>:<n>	character matrix	SEQUENCE OF T.61-String	t61str[m][n]	--
T!<m>:<n>	character matrix, durch Null ('\0') terminiert	SEQUENCE OF T.61-String	t61str[m][n]	--

[1] Dient in der Local Configuration zur Beschreibung der Datenstrukturen

[2] -- : in X_COMMON nicht vorhanden

[3] eine character matrix ist ein zweidimensionales character array

[4] gemäß CCITT Recommodation T.61 bzw. ISO 6937

Zeichensatz-Konvertierung bei X_C_TYPE und X_COMMON

Die Datenpuffer werden im ASCII-Code über das Netz geschickt. Ein Partner kann jedoch eine andere Zeichensatzkodierung als ASCII verwenden, wie z. B. eine BS2000-Anwendung, die EBCDIC verwendet. In diesem Fall konvertiert die XATMI-Bibliothek bei allen eingehenden und abgehenden Daten den ASN.1-Character-String-Typ *T61String* (siehe ISO 8824 [29]). Daher darf für das jeweilige Trägersystem (openUTM/UPIC/OpenCPIC) keine automatische Konvertierung generiert werden.

6.4 Programm-Schnittstelle

Die folgenden Abschnitte geben einen Überblick über die XATMI-Client-Programmschnittstelle. Eine detaillierte Beschreibung der Programmschnittstelle und der Error- und Returncodes findet sich in der X/Open-Spezifikation zu XATMI [28]. Die Kenntnis dieser Spezifikation ist für die Erstellung von XATMI-Programmen unbedingt erforderlich.

Die Programm-Schnittstelle ist sowohl für C als auch für COBOL verfügbar.

6.4.1 XATMI-Funktionen für Clients

Die folgende Tabelle listet alle XATMI-Client-Aufrufe sowie die zwei zusätzlichen openUTM-Client-Aufrufe auf und beschreibt, bei welchem Kommunikationsmodell sie verwendet werden.

C-Aufruf	COBOL-Aufruf	Kommunikationsmodell
tpcall	TPCALL	Synchroner Request/Response
tpacall[1]	TPACALL	Asynchroner Request/Response
tpgetreply	TPGETRPLY	Asynchroner Request/Response
tpcancel	TPCANCEL	Asynchroner Request/Response
tpconnect	TPCONNECT	Conversational
tpsend	TPSEND	Conversational
tprecv	TPRECV	Conversational
tpdiscon	TPDISCON	Conversational
tpalloc	--	Request/Response und Conversational
tprealloc	--	Request/Response und Conversational
tpfree	--	Request/Response und Conversational
tptypes	--	Request/Response und Conversational
tpinit	TPINIT	Client bei UPIC-Trägersystem initialisieren, nicht im XATMI-Standard enthalten
tpterm	TPTERM	Client vom UPIC-Trägersystem abmelden, nicht im XATMI-Standard enthalten

[1] Single-Request ist nur möglich mit gesetztem *TPNOREPLY*-Flag beim *tpacall*

Die beiden Funktionen *tpinit* und *tpterm* sind nicht im XATMI-Standard enthalten und dienen zum Anschluss von XATMI in das Trägersystem. Sie sind nachfolgend beschrieben.

Beim Trägersystem OpenCPIC ist das explizite An- und Abmelden mit *tpinit / tpterm* zwar möglich, aber nicht notwendig. Auf keinen Fall dürfen beim Aufruf *tpinit* Benutzerkennung und/oder Passwort übergeben werden.

6.4.2 Anschluss an das Trägersystem

Der openUTM-Client verwendet UPIC oder OpenCPIC als Trägersystem. Daher muss sich ein XATMI-Client-Programm explizit beim Trägersystem mit *tpinit* anmelden und mit *tpterm* abmelden:

1. *tpinit()*
2. XATMI-Aufrufe, z. B. *tpalloc(), tpcall(), tpconnect(), ...tpdiscon()*
3. *tpterm()*

Die beiden Aufrufe *tpinit* und *tpterm* sind im Folgenden beschrieben; alle weiteren XATMI-Aufrufe finden Sie in der XATMI-Spezifikation [28].

tpinit - Client initialisieren

Syntax

```
C:      #include <xatmi.h>
       int tpinit (TPCLTINFO *tpinfo)          (in)


COBOL:  01 TPCLTDEF-REC.
        COPY TPCLTDEF.
        CALL "TPINIT" USING TPCLTDEF-REC.
```

Beschreibung

Die Funktion *tpinit* initialisiert einen Client und identifiziert diesen beim Trägersystem. Sie muss als **erste** XATMI-Funktion in einem Client-Programm aufgerufen werden. Als Parameter ist in C ein Zeiger auf die vordefinierte Struktur *TPCLTINFO* zu übergeben, in COBOL muss das COBOL-Record *TPCLTDEF* versorgt werden.

C-Struktur *TPCLTINFO*:

```
#define  MAXTIDENT  9

typedef struct {
    long  flags;                    /* for future use */
    char  usrname[MAXTIDENT];
    char  cltname[MAXTIDENT];
    char  passwd [MAXTIDENT];
} TPCLTINFO;


COBOL-Record TPCLTDEF:

05 FLAG        PIC S(9) COMP-5.
05 USRNAME     PIC X(9).
05 CLTNAME     PIC X(9).
05 PASSWD      PIC X(9).
```

In *usrname* wird eine Benutzerkennung und in *passwd* ein Passwort eingetragen. Beide Parameter werden zur Einrichtung einer Conversation verwendet und dienen dazu, auf der UTM-Seite die Zugangsberechtigung nachzuweisen. Mit *cltname* (= local client name) wird der Client beim Trägersystem identifiziert.

cltname ist bei OpenCPIC der Name in der LOCAPPL-Anweisung.

Wenn *usrname* und *passwd* mit dem Nullstring initialisiert sind (COBOL: SPACES), dann werden die Security-Funktionen nicht aktiviert, d. h. es findet bei openUTM keine Zugangsprüfung statt. Enthält mindestens einer dieser beiden Parameter einen gültigen Wert, dann wird dieser von openUTM geprüft. Ein Nullstring im Parameter *cltname* bedeutet eine Voreinstellung von 8 Leerzeichen. Wird ein solcher leerer *cltname* angegeben, so wird der Eintrag eines Defaultservers in der OpenCIPIC-Generierungsdatei gesucht.

Wenn *tpinit* in C mit einem NULL-Zeiger aufgerufen wird, dann ist keine Zugangsprüfung aktiviert und der "local client name" ist mit 8 Leerzeichen vorbelegt. Bei COBOL muss dazu die Struktur mit SPACES versorgt werden.

Die Einträge in *usrname*, *passwd* und ggf. in *cltname* müssen den UTM-Namenskonventionen entsprechen, d. h. sie dürfen maximal acht Zeichen lang sein und müssen in C mit dem Stringende-Zeichen ("\0") abgeschlossen sein.

Returnwerte

tpinit liefert im Fehlerfall -1 zurück und setzt die Fehlervariable *tperrno* auf einen der folgenden Werte:

TPEINVAL
: Ein oder mehrere Parameter wurden mit einem ungültigen Wert versorgt.

TPENOENT
: Die Initialisierung konnte nicht durchgeführt werden, z. B. steht nicht genügend Speicherplatz für interne Puffer bereit.

TPEPROTO
: *tpinit* wurde an nicht erlaubter Stelle aufgerufen, z. B. der Client ist bereits initialisiert.

TPESYSTEM
: Es ist ein interner Fehler aufgetreten.

tpterm - Client abmelden

Syntax

```
C:      int tpterm ()

COBOL:  CALL "TPTERM".
```

Beschreibung

Die Funktion *tpterm* meldet einen Client, der mit dem letzten *tpinit* initialisiert worden war, beim Trägersystem ab. Nach *tpterm* ist kein XATMI-Aufruf mehr erlaubt, ausgenommen ein erneuter *tpinit*.

Returnwerte

tpterm liefert im Fehlerfall -1 zurück und setzt die Fehlervariable *tperrno* auf einen der folgenden Werte:

TPENOENT
: Der Client konnte sich nicht ordnungsgemäß abmelden. Ursache können z. B. Probleme beim Trägersystem sein.

TPEPROTO
: *tpterm* wurde an einer nicht erlaubten Stelle aufgerufen, d. h. der Client ist noch nicht initialisiert.

TPESYSTEM
: Es ist ein interner Fehler aufgetreten.

6.4.3 Include-Dateien und COPY-Elemente

C-Module mit XATMI-Aufrufen benötigen folgende Include-Dateien:

1. Die Datei *xatmi.h*.
2. Die Datei(en) mit den Datenstrukturen für alle typisierten Puffer, die im Modul verwendet werden, siehe auch Seite 99.

COBOL-Module mit XATMI-Aufrufen benötigen folgende COPY-Elemente:

1. Folgende COPY-Elemente:

   TPSTATUS, TPTYPE, TPSVCDEF und TPCLTDEF.
2. Die Datei(en) mit den Datenstrukturen für alle "typed records", die im Modul verwendet werden.

6.4.4 Ereignisse und Fehlerbehandlung

Wenn ein Ereignis eingetroffen oder ein Fehler aufgetreten ist, geben XATMI-Funktionen den Returnwert -1 zurück. Zur genaueren Bestimmung von Ereignis oder Fehler muss das Programm die Variable *tperrno* auswerten.

Bei der Conversational-Funktion *tprecv* zeigt *tperrno=TPEEVENT* an, dass ein Ereignis eingetroffen ist. Dieses Ereignis kann durch Auswerten des *tprevc*-Parameters *revent* bestimmt werden. Zum Beispiel wird ein erfolgreiches Beenden eines Conversational Service wie folgt angezeigt:

```
Returncode von tprecv =-1
tperrno=TPEEVENT
revent=TPEV_SVCSUCC
```

Bei der Funktion *tpsend* hat der Parameter *revent* keine Bedeutung.

Außerdem kann das Service-Programm beim Ende der Service-Funktion mit *tpreturn* über den Parameter *rcode* einen frei definierten Fehlercode zurückgeben, der clientseitig über die externe Variable *tpurcode* ausgewertet werden kann, siehe X/Open-Spezifikation zu XATMI [28].

6.4.5 Typisierte Puffer erstellen

Typisierte Puffer werden definiert durch Datenstrukturen in Include-Dateien (bei C) bzw. COPY-Elementen (bei COBOL), die in den beteiligten Programmen verwendet werden müssen.

Der Datenaustausch zwischen den Programmen erfolgt auf Basis dieser Datenstrukturen, die daher sowohl dem Client als auch dem Server bekannt sein müssen. Dabei sind alle Datentypen erlaubt, die in der Tabelle auf Seite 100 beschrieben sind.

Die Include- bzw. COBOL-COPY-Dateien, in denen die typisierten Puffer beschrieben sind, dienen als Eingabe für das Generierungsprogramm *xatmigen* (siehe Seite 114). Für diese Dateien gelten folgende Regeln:

– C- und COBOL-Datenstrukturen müssen in eigenen Dateien stehen. Eine Datei, die sowohl C-Include-Dateien als auch COBOL-COPY-Elemente enthält, ist als Eingabe nicht erlaubt.

– Die Dateien dürfen nur aus den Definitionen der Datenstrukturen, Leerzeilen und Kommentaranweisungen bestehen. In C sind Makroanweisungen erlaubt, d. h. Anweisungen, die mit '#' beginnen.

– Die Datenstrukturen-Definitionen müssen vollständig angegeben werden. Insbesondere müssen COBOL-Datensätze mit der Stufennummer "01" beginnen.

– Die Datenstrukturen dürfen nicht verschachtelt sein.

– Als Feldlängen sind nur absolute Werte und keine Makro-Konstanten erlaubt.

– Es sind nur die Datentypen erlaubt, die in der Tabelle auf Seite 100 beschrieben sind. Insbesondere sind bei C keine Zeiger-Typen zugelassen.

Mit Hilfe des Generierungsprogramms *xatmigen* muss der Anwender ggf. die Character-Arrays auf die ASN.1-Character-String-Typen (siehe ISO 8824 [29]) abbilden, da weder C noch COBOL diese Datentypen kennen (siehe “Abschnitt „Generierungsprogramm xatmigen“ auf Seite 114”).

Für C stehen XATMI-Aufrufe für die Speicherbelegung zur Verfügung (*tpalloc* ...).

Im Folgenden je ein einfaches Beispiel für C und COBOL.

Beispiel 1: C-Include für typisierten Puffer

```
typedef struct {
       char  name[20];  /* Personenname */
       int   age;       /* Alter        */
       char  sex;
       long  shoesize;
} t_person;

struct t_city {
       char  name[32];  /* Ortsname */
       char  country;
       long  inhabitants;
       short churches[20];
       long  founded;
}
```

Beispiel 2: COBOL-COPY für Typed Record

```
***** Personal-Record

01 PERSON-REC.
    05 NAME     PICTURE X(20).
    05 AGE      PIC     S9(9) COMP-5.
    05 SEX      PIC     X.
    05 SHOESIZE PIC     S9(9) COMP-5.


***** City-Record

01 CITY-REC.
05 NAME        PIC   X(32).
    05 COUNTRY     PIC   X.
    05 INHABITANTS PIC   S9(9) COMP-5.
    05 CHURCHES    PIC   S9(4) COMP-5 OCCURS 20 TIMES.
    05 FOUNDED     PIC   S9(9) COMP-5.
```

Weitere Beispiele finden Sie in der X/Open-Spezifikation zu XATMI [28].

6.4.6 Charakteristika von XATMI in openUTM-Client

Dieser Abschnitt beschreibt Besonderheiten, die durch die Implementierung der XATMI-Schnittstelle in openUTM auftreten.

Unterstützte XATMI-Aufrufe

Es werden alle für Clients relevanten XATMI-Aufrufe unterstützt. Hinzu kommen die beiden Aufrufe *tpinit* und *tpterm*.

Randbedingungen

- Für Clients mit dem Trägersystem OpenCPIC sind pro Service bis zu 256 Conversations möglich.
- Es dürfen innerhalb einer Client-Anwendung maximal 100 Pufferinstanzen gleichzeitig verwendet werden. Bei einer Anwendung in C heißt das z. B. maximal 100 *tpalloc*-Aufrufe ohne *tpfree*-Aufruf.
- Die maximale Nachrichtenlänge ist 32000 Byte.

  Die maximale Größe eines typisierten Puffers ist immer kleiner als die maximal mögliche Nachrichtenlänge, da die Nachrichten neben den Nettodaten noch einen "Overhead" enthalten. Je komplexer ein Puffer ist, desto größer ist der Overhead.

  Als Faustregel gilt : maximale Puffergröße = 2/3 der maximalen Nachrichtenlänge.

  Bei größeren Datenmengen sollte daher immer das Conversational Modell (*tpsend* / *tprecv*) verwendet werden.
- Für die Namenslängen gelten folgende Grenzwerte:

Servicename	16 Byte
Puffername	16 Byte

  Nach dem Standard dürfen Servicenamen 32 Byte lang sein, von denen allerdings nur die ersten 16 Byte relevant sind (Konstante *XATMI_SERVICE_NAME_LENGTH*). Es empfiehlt sich daher, für Servicenamen nicht mehr als 16 Byte zu verwenden.

Zusammenarbeit mit der TX-Schnittstelle

Beim Aufruf eines Service wird mit dem *TPTRAN* / *TPNOTRAN* Flag gesteuert, ob der Service in die globale Transaktion eingeschlossen wird.

Wird ein Service mit gesetztem *TPTRAN* Flag gestartet, so ist er implizit in der globalen Transaktion eingeschlossen. Im Programm sind keine expliziten TX-Aufrufe notwendig.

Bei *tpreturn* wird mit *TPSUCCESS* bzw. *TPFAIL* gesteuert, ob die Transaktion bestätigt (Commit) oder zurückgesetzt (Rollback) werden soll.

Für weitere Informationen verweisen wir auf die X/Open Sezifikation der XATMI-Schnittstelle [28] (Kapitel “Transaction Functions Affecting the XATMI Interface”).

6.5 Konfigurieren

Für jede XATMI-Anwendung muss der Anwender eine Local Configuration erzeugen. Diese beschreibt die bereitgestellten und genutzten Services mit ihren Zieladressen sowie die verwendeten typisierten Puffer mit ihrer Syntax. Die Information ist in einer Datei, dem Local Configuration File (LCF), hinterlegt, die von der Anwendung beim Starten einmal gelesen wird. Ein LCF ist sowohl für die Client- wie auch für die Service-Seite notwendig.

6.5.1 Local Configuration File erzeugen

Als Anwender müssen Sie eine Eingabedatei erstellen, die sogenannte LC-Definitionsdatei. Diese Eingabedatei muss aus einzelnen Zeilen aufgebaut werden, für die folgende Syntaxregeln gelten:

– Eine Zeile beginnt mit einer SVCU- oder BUFFER-Anweisung und spezifiziert genau einen Service oder einen Subtyp (=typisierten Puffer).

– Zwei Operanden werden durch ein Komma getrennt.

– Eine Anweisungszeile wird durch ein Semikolon (';') abgeschlossen.

– Nehmen die Operanden mehr als eine Zeile ein, dann muss jeweils am Zeilenende das Fortsetzungszeichen '\' (Gegenschrägstrich) stehen.

– Eine Kommentarzeile beginnt mit dem '#'-Zeichen in der ersten Spalte

– Leerzeilen können (zur besseren Lesbarkeit) eingefügt werden.

Aus der LC-Definitionsdatei erstellen Sie mit Hilfe des Generierungsprogramms *xatmigen* das eigentliche Local Configuration File (Seite 114).

Im Folgenden werden die SVCU- und die BUFFER-Anweisung beschrieben.

SVCU-Anweisung: Aufrufbaren Service definieren

Eine SVCU-Anweisung beschreibt für den Client die Eigenschaften, die notwendig sind, um einen Service in der Partneranwendung aufrufen zu können.

Operator	Operanden	Erläuterung
SVCU	internal-service-name	Pflichtoperand, maximal 16 Byte
	[,RSN=remote-service-name]	Standard: internal-service-name
	[,TAC=transaction-code]	Standard: internal-service-name
	,{DEST=destination-name | .DEFAULT}	Pflichtoperand, Partneranwendung
	[,MODE=[RR |CV]]	Standard:RR (Request/Response)
	[,BUFFERS=(subtype-1,...,subtype-n)]	Standard: kein Subtyp

internal-service-name
: maximal 16 Byte langer Name, unter dem ein (ferner) Service im Programm angesprochen wird. Dieser Name muss innerhalb der Anwendung eindeutig sein, d. h. er darf in der LCF nur einmal vorkommen.

  Mehrfachdefinitionen werden nicht überprüft. Der erste *internal-service-name* ist gültig, weitere gleichen Namens werden ignoriert.

  Pflichtoperand!

RSN=remote-service-name
: maximal 16 Byte langer Name eines Services in der fernen Anwendung. Dieser Name wird an die ferne Anwendung übertragen. Er darf in der LCF mehrfach vorkommen.

  Wird dieser Operand weggelassen, dann wird standardmäßig RSN=*internal-service-name* gesetzt.

TAC=transaction-code
: maximal 8 Byte langer Transaktionscode, unter dem der Service in der fernen Anwendung generiert sein muss.

  Wird dieser Operand weggelassen, dann setzt *xatmigen* standardmäßig TAC=*internal-service-name* und kürzt diesen ggf. auf die ersten 8 Byte.

  Mit dem Transaktionscode KDCRECVR kann man einen Recovery-Service definieren, der die letzte Ausgabenachricht von openUTM an den Client schickt.

DEST=destination-name
: Maximal 8 Byte lange Identifikation der Partneranwendung.

  Dieser Name muss in einer SYMDEST-Anweisung der OpenCPIC-Generierung generiert sein (siehe Seite 43).

  .DEFAULT
  Es wird ein Default-Server verwendet.

MODE
: Bestimmt, welches Kommunikationsmodell für den Service verwendet wird:

= RR	Request-Response-Modell (Standardwert)
= CV	Conversational Modell.

BUFFERS=(subtype-1,...,subtype-n**)**
: Liste von Subtyp-Namen, die an den Service geschickt werden dürfen. Jeder Name darf maximal 16 Byte lang sein, wobei alle Zeichen des ASN.1-Character-String-

Typs *PrintableString* (siehe “Seite 199” sowie ISO 8824 [29]) erlaubt sind.

Für jeden hier aufgeführten Subtyp muss eine eigene BUFFER-Anweisung angegeben werden, mit der die Eigenschaften des Subtyps definiert werden (siehe unten).

Der Operand BUFFERS ist stellungssensitiv und muss (falls angegeben) immer der letzte Operand der Anweisung sein.

Wird BUFFERS weggelassen, dann sollte an den Service nur ein Puffer vom Typ *X_OCTET* gesendet werden.

BUFFER-Anweisung: typisierten Puffer definieren

Eine BUFFER-Anweisung definiert einen typisierten Puffer. Gleichnamige Puffer müssen client- und serverseitig gleich definiert sein.
Mehrfachdefinitionen werden nicht überprüft. Der erste Puffer-Eintrag ist gültig, alle anderen werden ignoriert.

Puffer des Typs *X_OCTET* haben keine besonderen Eigenschaften und benötigen deshalb keine Definition. Typisierte Puffer werden mit folgenden Parametern definiert:

Operator	Operanden	Erläuterung
BUFFER	subtype-name	maximal 16 Byte
	[,REC=referenced-record-name]	Standard: subtype-name
	[,TYPE=X_COMMON | X_C_TYPE]	Standard: xatmigen setzt TYPE automatisch

subtype-name
: Maximal 16 Byte langer Name des Puffers, muss auch bei der SVCU-Anweisung im Operanden BUFFERS angegeben werden. Der Name muss in der Anwendung eindeutig sein.

REC=referenced-record-name
: Name der Datenstruktur für den Puffer, z. B. ist dies bei C-Strukturen der Name des "typedef" bzw. der "sstruct-Name".

  Wird der Operand weggelassen, dann setzt *xatmigen* REC=subtype-name.

TYPE=
: Typ des Puffers, näheres zu den Typen siehe Seite 99.

  Wird der Operand weggelassen, dann setzt xatmigen den Typ auf *X_C_TYPE* oder *X_COMMON*, je nachdem, welche elementaren Datentypen verwendet wurden.

TYPE wird ignoriert, wenn für TYPE nicht X_COMMON oder X_C_TYPE gesetzt ist oder wenn die Datenstruktur nicht vom angegebenen Puffertyp ist.

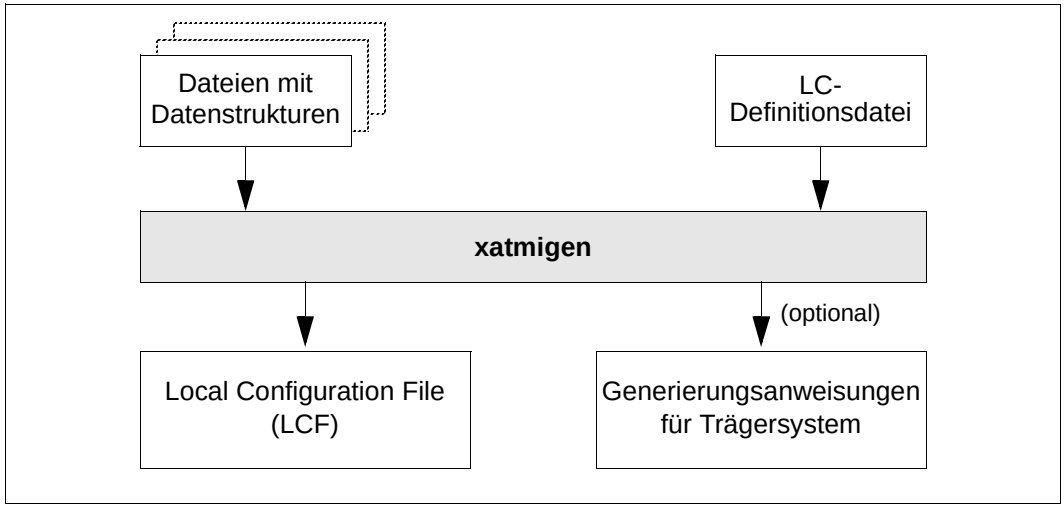
xatmigen erzeugt beim Generierungslauf zusätzlich zwei Operanden mit folgender Bedeutung:

LEN=länge — Länge des Datenpuffers.

SYNTAX=code — Syntaxbeschreibung der Datenstruktur in der Code-Darstellung, wie sie in der Tabelle auf Seite 100 aufgeführt ist.

6.5.2 Generierungsprogramm xatmigen

xatmigen bereitet aus einer Datei mit der Definition der Local Configuration (LC-Definitionsdatei) und einer oder mehreren Dateien mit C- oder COBOL-Datenstrukturen (LC Description Files) ein Local Configuration File (LCF) auf, siehe folgendes Bild:



Die Local Configuration File ist genauso aufgebaut wie die LC-Definitionsdatei und unterscheidet sich von dieser nur durch die Beschreibung von Puffertyp, Pufferlänge und Syntaxstring des Puffers. Das heißt, die BUFFER-Anweisungen werden gegenüber der LC-Definitionsdatei erweitert um die Operanden LEN, SYNTAX und ggf. TYPE.

Falls in der LC-Definitionsdatei der Puffertyp nicht angegeben ist, generiert *xatmigen* den jeweils "kleinsten" Wertebereich für den Puffertyp, d. h. zuerst den Typ *X_COMMON*.

Alle Dateinamen müssen explizit angegeben werden. Optional kann eine Datei erstellt werden, die Generierungsanweisungen für das Trägersystem enthält.

Erfolgs- bzw. Fehlermeldungen werden nach *stdout* und *stderr* geschrieben.

Obwohl das Editieren des LCF prinzipiell möglich ist, wird dringend davon abgeraten.

Das Dienstprogramm *xatmigen* ist unter dem Pfadnamen *instpfad/upic/xatmi/ex/xatmigen* zu finden.

Syntax

xatmigen [system] **-d** lcdf_name [**-l** lcf_name] [**-i**] [**-c** stringcode] [descript_file_1] ... [descript_file_n]

Beschreibung

system

Falls angegeben, wird eine Datei mit Generierungsanweisungen für das Trägersystem *system* erzeugt. Mögliche Angaben sind:

upic es werden Einträge für upicfile erzeugt, (Dateiname: *xtupic.def*)

cpic es wird ein Fragment für die OpenCPIC-Generierung erzeugt. Dieses Fragment muss noch ergänzt werden (Dateiname:*xtcpic.def*)

system muss, sofern angegeben, immer der erste Parameter von *xatmigen* sein. Fehlt die Angabe, dann werden keine Generierungsanweisungen erzeugt.

-d lcdf_name

Name der LC-Definitionsdatei. Dieser Schalter muss immer angegeben werden.

-l lcf_name

Name des zu erzeugenden Local Configuration Files. Der Name muss den Konventionen des jeweiligen Betriebssystems entsprechen.

Ein eventuell vorhandenes, gleichnamiges LCF wird kommentarlos überschrieben.

Wird der Schalter weggelassen, dann erzeugt *xatmigen* im aktuellen Verzeichnis die Datei *xatmilcf*.

-i Interaktiver Modus, d. h. bei jedem typisierten Puffer, der ein Character-Array enthält, wird dessen Stringcode erfragt. Die möglichen Angaben für den Stringcode sind bei Schalter *-c* beschrieben.

Der Schalter *-i* hat Vorrang vor einem eventuell ebenfalls vorhandenen Schalter *-c*. Wenn *xatmigen* im Batch-Betrieb abläuft, dann darf der Schalter *-i* nicht angegeben werden.

-c stringcode

Der angegebene Stringtyp *stringcode* gilt für den gesamten *xatmigen*-Lauf, d. h. für alle Character-Arrays. Im interaktiven Modus (Schalter *-i*) wird der Schalter *-c* ignoriert.

Für *stringcode* sind folgende Angaben möglich (siehe Tabelle auf Seite 100):

C|c	*OCTET STRING*
C!|c!	null-terminated *OCTET STRING*
T|t	*T61String*
T!|t!	null-terminated *T61String* (Standardwert)

Bei fehlender Angabe wird der Wert *T!* voreingestellt.

descript_file_1 . . . descript_file_n

Liste der Dateien, die die Include- bzw. COPY-Elemente mit den Datenstrukturen der typisierten Puffer enthalten.

Fehlt die Angabe, ist nur der Puffertyp X_OCTET zugelassen.

Hinweis

Dem Schalter **-d** und, sofern angegeben, den Schaltern **-l** und **-c** muss jeweils der zugehörige Parameter folgen. Die Angabe des Schalters ohne nachfolgenden Parameter ist nicht zulässig.

6.5.3 Trägersystem OpenCPIC und UTM-Partner konfigurieren

Um eine XATMI-Anwendung in OpenCPIC funktionsfähig zu machen, müssen Sie

– die OpenCPIC-Generierung mit der Local Configuration und der Partner-Generierung abgleichen
– die Initialisierungsparameter, die in *tpinit* angegeben werden, mit der Generierung der UTM-Anwendung abstimmen

6.5.3.1 OpenCPIC generieren

Ein XATMI-Client mit Trägersystem OpenCPIC ist wie eine reine OpenCPIC-Anwendung zu generieren (siehe Kapitel „OpenCPIC generieren“ auf Seite 29). Dabei sind folgende Besonderheiten zu beachten:

– Der für *locappl_name* angegebene Name in der LOCAPPL-Anweisung (siehe Seite 32) wird beim Aufruf *tpinit* übergeben.

– Der für *symdest_name* angegebene Name in der SYMDEST-Anweisung (siehe Seite 43) muss mit dem Wert des Operanden DEST in der SVCU-Anweisung (siehe Seite 111) übereinstimmen. Außerdem ist PARTNER-APRO=XATMI zu setzen.

6.5.3.2 UTM-Partner konfigurieren

UTM-seitig ist für den KDCDEF-Lauf eine OSI-TP-Kopplung zu generieren.

Beispiel:

```
ACCESS-POINT SERVACCP,
             P-SEL=NONE, S-SEL=NONE, T-SEL=C'SERVNAME'

OSI-LPAP RCLTNAME,
         ASSOCIATION-NAMES = ASSONAME,
         ASSOCIATIONS = 2,
         CONTWIN = 1,
         CONNECT = 1,
         PERMIT = ADMIN

 OSI-CON RCLTNAME,
         P-SEL=NONE, S-SEL=NONE,
         T-SEL=C'LCLTNAME',
         N-SEL=C'CLTNODE',
         LOCAL-ACCESS-POINT = SERVACCP,
         OSI-LPAP = RCLTNAME,
         ACTIVE = YES
```

Laufen Client und Server auf verschiedene Systemen, muss auch eine entsprechende TNSX-Generierung vorgenommen werden.

6.5.3.3 Initialisierungsparameter und UTM-Generierung

Ein XATMI-Client wird mit dem Aufruf *tpinit* initialisiert. In der Struktur *TPCLTINIT* werden Parameter für Benutzerkennung und Passwort sowie für den Local Name übergeben.

Für den Parameter Local Name muss bei Trägersystem OpenCPIC immer der Name angegeben werden, der im TNSX generiert ist. Der TNS-Name muss dann auf der UTM-Seite in der OSI-CON-Anweisung mit dem Parameter TSEL definiert sein.

6.6 XATMI-Client-Programme binden und starten

Beim Binden eines XATMI-Client-Programms müssen zu den Anwendermodulen folgende Bibliotheken dazugebunden werden.

– XATMI-Client-Bibliothek *libxtclt.so* (Pfad *instpfad/upic/xatmi/sys/*)
– CPI-C-Bibliothek *openCPIC-pfad/libocpic.a*
– Unix- oder Linux-Systembibliothek “mathlib” (Schalter *-lm*)

Ein XATMI-Client-Programm wird wie andere OpenCPIC-Anwendungsprogramme als ausführbares Programm gestartet (siehe “Abschnitt „Anwendungsprogramme starten“ auf Seite 88”). Zuvor muss der OpenCPIC-Manager gestartet worden sein.

6.6.1 Umgebungsvariablen

Vor dem Start eines XATMI-Client-Programms müssen folgende Umgebungsvariablen gesetzt werden:

XTCLTTR Diese Umgebungsvariable muss gesetzt werden, um den XATMI-Trace einzuschalten (siehe nachfolgenden Abschnitt “XATMI-Trace aktivieren“).

XTPATH Pfadname für die Protokolldateien des XATMI-Trace siehe nachfolgenden Abschnitt “XATMI-Trace aktivieren“).

XTLCF Dateiname für das Local Configuration File (LCF)
Falls diese Variable nicht gesetzt wird, dann wird im aktuellen Arbeitsverzeichnis eine Datei mit dem Namen *xatmilcf* gesucht.

6.6.2 XATMI-Trace aktivieren

Zu Diagnosezwecken können Sie den Trace der Bibliothek *libxtclt.a* einschalten. Damit bewirken Sie, dass *libxtclt.a* während des Programmablaufs Traceinformationen mitprotokolliert. Der XATMI-Trace arbeitet in zwei Stufen (Trace-Level): Schnittstellen-Trace (Trace-Level 1) und Fehler-Trace (Trace-Level 2). Zur Fehlerdiagnose in XATMI-Anwendungsprogrammen ist im allgemeinen der Schnittstellen-Trace ausreichend.

Die Steuerung des XATMI-Trace erfolgt über Umgebungsvariablen. Folgende Umgebungsvariablen werden durch die Bibliothek ausgewertet:

XTCLTTR Diese Umgebungsvariable muss gesetzt werden, um den XATMI-Trace einzuschalten.

- **=E** (Error)
  Aktiviert den Fehler-Trace.
- **=I** (Interface)
  Aktiviert den Schnittstellen-Trace für die XATMI-Aufrufe.
- **=F** (Full)
  Aktiviert den Fehler-Trace und den Schnittstellen-Trace

XTPATH=pathname gibt das Verzeichnis an, in welchem die Protokolldateien des XATMI-Trace abgelegt werden sollen. Ist XTPATH nicht gesetzt, werden die Protokolldateien im aktuellen Arbeitsverzeichnis des XATMI-Client-Programms abgelegt.

Jeder XATMI-Client-Prozess schreibt die Traceinformationen in eine eigene Datei, die in zwei Generationen (alt und neu) existieren kann.

Die maximale Größe einer Protokolldatei beträgt ca. 128 KByte. Sobald diese Größe erreicht wird, wird auf eine zweite Datei umgeschaltet. Hat auch diese die Maximalgröße erreicht, wird wieder in die erste Datei geschrieben. Der Name einer Protokolldatei eines XATMI-Client-Programms setzt sich wie folgt zusammen:

XTCpid**.**suff

pid und suff sind Platzhalter mit der folgenden Bedeutung:

pid Prozess-Id des XATMI-Client-Prozesses

suff Dateinamen-Suffix, der die Generation kennzeichnet: 1 oder 2
Die jeweils jüngere Protokolldatei erkennt man an den Zeitstempeln

Die Protokolldateien sind ASCII-Dateien und ohne weitere Aufbereitung direkt lesbar.

7 Kommunikation mit Partneranwendungen

Das Produkt OpenCPIC bietet Ihnen mit den Programmschnittstellen X/Open CPI-C 2.0 und X/Open TX die Möglichkeit, mit Partnern, die über OSI-TP erreicht werden können, zu kommunizieren und globale Transaktionen abzuwickeln.

Dieses Kapitel enthält Hinweise und Beispiele für Konfigurationsmöglichkeiten in verschiedenen Client-Server-Umgebungen.

Eine Partneranwendung von OpenCPIC kann sein:

- OpenCPIC
- eine UTM-Anwendung
- eine OSI-TP-Anwendung

Eine Beispielkonfiguration für die Kommunikation mit einer OpenCPIC-Anwendung finden Sie im “Abschnitt „OpenCPIC-Kommunikation im homogenen Umfeld“ auf Seite 122“.

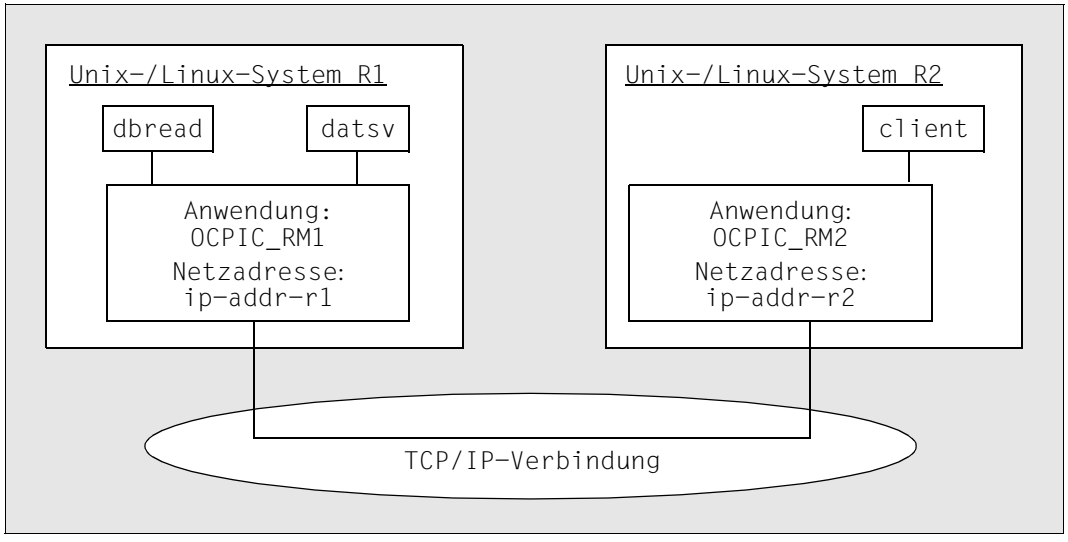
Die Kommunikation mit UTM-Anwendungen ist im “Abschnitt „OpenCPIC-Kommunikation mit openUTM“ auf Seite 125“ beschrieben.

Hinweise für die Verbindung mit Anwendungen anderer Hersteller finden Sie im “Abschnitt „OpenCPIC-Kommunikation mit Fremdsystemen“ auf Seite 139“).

7.1 OpenCPIC-Kommunikation im homogenen Umfeld

Dieser Abschnitt gibt ein Beispiel für eine einfache Kommunikation zwischen zwei OpenCPIC-Anwendungsprogrammen auf unterschiedlichen Unix- oder Linux-Systemen.

Das folgende Bild veranschaulicht den Sachverhalt:



Beispiel für eine Kommunikation zwischen zwei OpenCPIC-Anwendungen

Auf jedem der beiden Unix-/Linux-Systeme R1 und R2 ist das Produkt OpenCPIC installiert und je ein OpenCPIC-Manager konfiguriert. Auf dem System R1 existieren zwei CPI-C-Anwendungsprogramme *dbread* und *datsv*. Auf dem System R2 gibt es ein Anwendungsprogramm *client*.

dbread

Das Programm */home/dbserver/dbread* soll bei Bedarf vom OpenCPIC-Manager mit der Benutzerkennung *dbuser* gestartet werden. Es liefert dem Partner Daten aus einer lokalen Datenbank. (Der besseren Übersicht wegen wurde der lokale Resource Manager im Bild weggelassen. Er ist für die Konfigurierung nicht von Bedeutung.) Vom Partner soll das Programm mit dem lokalen Namen (TPSU Title) *GETDB* angesprochen werden.

Das Programm */home/dbserver/dbread* enthält folgende CPI-C-Aufrufe in der angegebenen Reihenfolge:

```
Accept_Conversation
Receive
Receive
Send_Data
Deallocate
```

datsv

Das Programm */home/cpic/datsv* soll immer beim Systemstart gestartet werden. Es soll lediglich Daten entgegennehmen und im lokalen Dateisystem ablegen, ohne mit einem lokalen Resource Manager zusammenzuarbeiten. Vom Partner soll es mit dem lokalen Namen (TPSU Title) *RCVDATA* aufgerufen werden.

Das Programm */home/cpic/datsv* enthält folgende CPI-C-Aufrufe in der angegebenen Reihenfolge:

```
Specify_Local_TP_Name          /* tp_name = RCVDATA */

in Schleife:
Accept_Conversation
Receive
```

client

Das CPI-C-Anwendungsprogramm */home/locpappl/client* auf dem System R2 wird von unterschiedlichen Benutzern aufgerufen. Es soll je nach Wunsch eine Conversation zum Programm *GETDB* (*dbread*) oder *RCVDATA* (*datsv*) auf dem System R1 aufbauen.

Das Programm */home/locpappl/client* enthält folgende CPI-C-Aufrufe:

```
if ( ... )                      /* Wunschpartner GETDB */
{
    Initialize_Conversation     /* sym_dest_name = GETDB */
    Allocate
    Send_Data
    Prepare_To_Receive
    Receive
}
else                           /* Wunschpartner RCVDATA */
{
    Initialize_Conversation    /* sym_dest_name = RCVDATA */
    Allocate
    Send_Data
    Deallocate
}
```

Für dieses Beispiel sind im Folgenden die TNSX- und die OpenCPIC-Generierung sowie die RC-Datei für OpenCPIC dargestellt.

TNSX-Generierung (tnsxcom**)**

System R1

```
; lokaler OpenCPIC-Manager
ocpic_loc \
    PSEL V ''
    SSEL V ''
    TSEL LANINET A'102'
    TSEL RFC1006 A'ocp1'

; Partner-OpenCPIC-Manager
ocpic_r2 \
    PSEL V ''
    SSEL V ''
    TA RFC1006 ip-addr-r2 \
        PORT 102 A'ocp2'
```

System R2

```
; lokaler OpenCPIC-Manager
ocpic_loc \
    PSEL V ''
    SSEL V ''
    TSEL LANINET A'102'
    TSEL RFC1006 A'ocp2'

; Partner-OpenCPIC-Manager
ocpic_r1 \
    PSEL V ''
    SSEL V ''
    TA RFC1006 ip-addr-r1 \
        PORT 102 A'ocp1'
```

OpenCPIC-Generierung (ocpic_gen**)**

System R1

```
LOCAPPL ocpic_loc,
    APT = (1,17, 25, 1001),
    AEQ = 300

PARTAPPL ocpic_r2,
    APT = (1, 20, 9, 0, 0),
    AEQ = 12,
    CCR = YES

LOCAPRO GETDB,
    PATH = /home/dbserver/dbread,
    LOGIN = dbuser
```

System R2

```
LOCAPPL ocpic_loc,
    APT = (1, 20, 9, 0, 0),
    AEQ = 12

PARTAPPL ocpic_r1,
    APT = ( 1, 17, 25, 1001),
    AEQ = 300,
    CCR = YES

SYMDEST R1GETDBS,
    PARTNER-APPL = ocpic_r1,
    PARTNER-APRO = GETDB

SYMDEST R1RCVDAT,
    PARTNER-APPL = ocpic_r1,
    PARTNER-APRO = RCVDATA
```

System-RC-Datei (/etc/rc2.d/S90OCPIC)

System R1

```
ocpic_start &
/home/ocpappl/datsv &
```

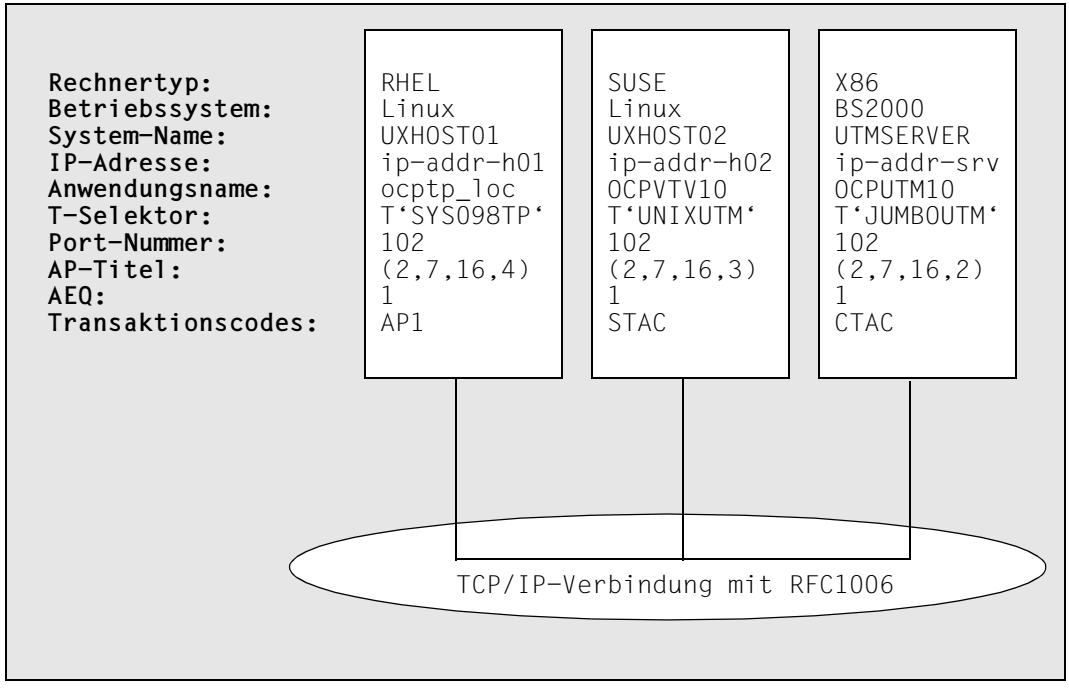
System R2

```
ocpic_start &
```

7.2 OpenCPIC-Kommunikation mit openUTM

Dieser Abschnitt gibt ein Beispiel für eine Kommunikation zwischen OpenCPIC-Anwendungsprogrammen auf einem Unix-/Linux-System und UTM-Anwendungsprogrammen auf einem BS2000-System oder Unix-/Linux-System.

Das folgende Bild veranschaulicht diese Konfiguration:



Beispiel für eine Kommunikation zwischen einer OpenCPIC- und einer UTM-Anwendung

Nachfolgend sind für alle Systeme die OpenCPIC-Generierung, die UTM-Generierung, die Einträge für das Transportsystem sowie eine Anzahl von Programmbeispielen dargestellt.

UTM-Generierung (KDCDEF) im BS2000-System

```
MAX APPLINAME = OCPUTMAP,
UTMD APT = (2, 7, 16, 2),
ACCESS-POINT OCPUTM10,
    P-SEL = *NONE,
    S-SEL = *NONE,
    T-SEL = C'JUMBOUTM',
    AEQ = 1
OSI-CON SO98TP80,
    P-SEL = *NONE,
    S-SEL = *NONE,
    T-SEL = C'SYSO98TP',
    N-SEL = C'UXHOST01',
    LOCAL-ACCESS-POINT = OCPUTM10,
    OSI-LPAP = OLP80
OSI-LPAP OLP80,
    APT = (2,7,16,4),
    AEQ = 1,
    ASS-NAMES = SES1,
    ASSOCIATIONS = 10,
    CONTWIN = 5
LTAC CPICTP,
    LPAP = OLP80,
    RTAC = AP1
TAC CTAC ...
```

BCIN-Kommando im BS2000-System:

```
/BCIN UXHOST01,IPADR=(ip-addr-h01),...
```

UTM-Generierung (KDCDEF) im Linux-System SUSE

```
MAX APPLINAME = utm608ap,
UTMD APT = (2, 7, 16, 3),
ACCESS-POINT OCPVTV10,
    P-SEL = *NONE,
    S-SEL = *NONE,
    T-SEL = C'utm608ot',
    AEQ = 1
OSI-CON SO98TP81,
    P-SEL = *NONE,
    S-SEL = *NONE,
    T-SEL = C'sys098qs',
    N-SEL = C'UXHOST01',
    LOCAL-ACCESS-POINT = OCPVTV10,
    OSI-LPAP = OLP81
OSI-LPAP OLP81,
    APT = (2,7,16,4),
    AEQ = 1,
    ASS-NAMES = SES1,
    ASSICIATIONS = 10,
    CONTWIN = 5
LTAC CPICTP,
    LPAP = OLP81,
    RTAC = AP1
TAC STAC ...
```

TNSX-Generierung für UTM in Linux-System SUSE mit tnsxcom

```
utm608ap\
        TSEL RFC1006 A'13659'
utm608ot\
        TSEL RCF1006 T'UNIXUTM'
sys098qs.UXHOST01\
        TA   RFC1006 ip-addr-h01 PORT 102 T'SYSO98TP'
```

OpenCPIC-Generierung im Linux-System RHEL mit ocpic_gen

```
LOCAPPL ocptp_loc,

    APT = (2,7,16,4),
    AEQ = 1
PARTAPPL ocpvtv10,
    APT = (2,7,16,3),
    AEQ = 1,
    ASSOCIATIONS = 10,
    CONTWIN = (5 5),
    CONNECT = 10
PARTAPPL ocputm10,
    APT = (2,7,16,2),
    AEQ = 1,
    ASSOCIATIONS = 10,
    CONTWIN = (5 5),
    CONNECT = 10
SYMDEST RUTMAC,
    PARTNER-APPL = ocpvtv10,
    PARTNER-APRO = STAC
SYMDEST BUTMAC,
    PARTNER-APPL = ocputm10,
    PARTNER-APRO = CTAC
LOCAPRO AP1
    PATH  = /home/server/ap1
    LOGIN = utmclien
```

TNSX-Generierung für OpenCPIC im Linux-System RHEL mit tnsxcom

```
ocptp_loc\
        PSEL V''
        SSEL V''
        TSEL RCF1006 T'SYS098TP'
ocpvtv10\
        PSEL V''
        SSEL V''
        TA   RFC1006 ip-addr-h02 PORT 102 T'UNIXUTM'
ocputm10\
         PSEL V''
         SSEL V''
         TA   RFC1006 ip-addr-srv PORT 102 T'JUMBOUTM'
```

Programmbeispiele

Beispiel 1:

Einschrittvorgang bei openUTM

```
Initialize_Conversation
Allocate
Send_Data
Receive                                  ->   INIT
                                              MGET
                                              MPUT
. . . CM_COMPLETE_DATA_RECEIVED          <-   PEND FI
Receive CM_DEALLOCATED_NORMAL
```

Beispiel 2:

Mehrschrittvorgang bei openUTM, Senden von zwei Nachrichten hintereinander

```
Initialize_Conversation
Allocate
Send_Data
Send_Data
Receive                                  ->   INIT
                                              MGET
                                              MGET
                                              MPUT
                                              MPUT
. . . CM_COMPLETE_DATA_RECEIVED          <-   PEND RE/KP
Receive CM_COMPLETE_DATA_RECEIVED
Receive CM_SEND_RECEIVED
Send_Data
Receive                                  ->   INIT
                                              MGET
                                              MPUT
. . . CM_COMPLETE_DATA_RECEIVED          <-   PEND FI
Receive CM_DEALLOCATED_NORMAL
```

Beispiel 3:

Asynchronvorgang bei openUTM Quittung bei der Dialogbeendigung

```
Initialize_Conversation
Set_Sync_Level CM_CONFIRM
Allocate
Set_Send_Type CM_BUFFER_DATA
Send_Data
Send_Data
Deallocate                                  --->   INIT
                                                   FGET
                                                   FGET
. . . CM_OK                                 <---   PEND FI
```

Beispiel 4:

Aufruf eines CPI-C-Anwendungsprogramms als Asynchronprogramm von openUTM, CPI-C-Anwendungsprogramm bereits gestartet

```
Specify_Local_TP_Name
Initialize_For_Incoming
Accept_Incoming
                                                   APRO AM   > A
                                                   FPUT NT   > A
                                                   FPUT NT   > A
. . . CM_OK                                 <---   PEND FI
Receive CM_COMPLETE_DATA_RECEIVED
Receive CM_COMPLETE_DATA_RECEIVED
Receive CM_CONFIRM_DEALLOC_RECEIVED
```

Beispiel 5:

Aufruf eines CPI-C-Anwendungsprogramms als Dialogprogramm von openUTM, CPI-C-Anwendungsprogramm noch nicht gestartet

```
                                        APRO DM   > A
                                        MPUT NT   > A
Start durch OpenCPIC-Manager      <---  PEND KP
Accept_Conversation
Receive CM_COMPLETE_DATA_RECEIVED
Send_Data
Deallocate                        --->  INIT
                                        MGET NT   > A   KCRST=(C,U)
```

Beispiel 6:

CPI-C-Anwendungsprogramm fordert Quittung, UTM-Anwendung gibt negative Quittung

```
Initialize_Conversation
Set_Sync_Level CM_CONFIRM
Allocate
Set_Send_Type CM_BUFFER_DATA
Send_Data
Prepare_To_Receive                -->   INIT
                                        MGET
                                        MGET          KCRMGT=H
                                        MPUT EM
. . . CM_PROGRAM_ERROR_PURGING    <--   PEND FI
Deallocate
```

Beispiel 7:

UTM-Anwendung fordert Quittung, CPI-C-Anwendung gibt positive Quittung, CPI-C-Anwendung bereits gestartet

```
Specify_Local_TP_Name
Initialize_For_Incoming
Accept_Incoming
                                          INIT
                                          MGET
                                          APRO DM   > A
                                          MPUT NT   > A
                                          MPUT HM   > A
. . . CM_OK                          <--  PEND KP
Receive CM_COMPLETE_DATA_RECEIVED
Receive CM_CONFIRM_SEND_RECEIVED
Confirmed
Send_Data
Deallocate                           -->  INIT
                                          MGET NT   > A  KCRMGT=C
                                          MGET NT   > A  KCRST=(C,U)
                                          MPUT NE
                                          PEND FI
```

Beispiel 8:

Senden einer Fehlernachricht vom CPI-C-Anwendungsprogramm zur UTM-Anwendung

```
Initialize_Conversation
Allocate
Send_Data
Send_Error
Send_Data
Prepare_To_Receive                   -->   INIT
                                           MGET
                                           MGET          KCRMGT=E
                                           MGET
                                           MPUT NE
Receive CM_COMPLETE_DATA_RECEIVED    <---  PEND FI
Receive CM_DEALLOCATED_NORMAL
```

Beispiel 9:

Senden einer Fehlernachricht von der UTM-Anwendung zum CPI-C-Anwendungsprogramm

```
Initialize_Conversation
Allocate
Send_Data
Prepare_To_Receive                          -->    INIT
                                                   MGET
                                                   MPUT
                                                   MPUT EM
Receive CM_COMPLETE_DATA_RECEIVED           <---   PEND FI
Receive CM_PROGRAM_ERROR_PURGING
Receive CM_DEALLOCATED_NORMAL
```

Beispiel 10:

Abnormale Dialogbeendigung durch das CPI-C-Anwendungsprogramm, CPI-C-Anwendungsprogramm bereits gestartet

```
Specify_Local_TP_Name
Initialize_For_Incoming
Accept_Incoming
                                                   INIT
                                                   MGETAPRO
                                                   DM        >A
                                                   MPUT NT   >A
. . . CM_OK                                 <---   PEND KP
Receive CM_COMPLETE_DATA_RECEIVED
Set_Deallocate_Type CM_DEALLOCATE_ABEND
Deallocate                                  --->   INIT
                                                   MGET NT   >   KCRST=(Z,U)
                                                   MPUT NE   A
                                                   PEND FI
```

Beispiel 11:

Abnormale Dialogbeendigung durch die UTM-Anwendung

```
Initialize_Conversation
Allocate
Send_Data
Prepare_To_Receive                      -->    INIT
                                               MGET
                                               MPUT
Receive CM_DEALLOCATED_ABEND            <---   PEND FR
```

Beispiel 12:

Abnormale Dialogbeendigung durch die UTM-Anwendung ohne Vorgangsabbruch

```
                                               INIT
                                               MGET
                                               APRO DM   > A
                                               MPUT NE   > A
Accept_Conversation                     <---   PEND KP
Receive CM_COMPLETE_DATA_RECEIVED
Receive CM_SEND_RECEIVED
Send_Data                               -->    INIT
                                               MGET      > A
                                               CTRL AB   > A
                                               MPUT NE
Receive CM_DEALLOCATED_ABEND            <---   PEND FI
```

Beispiel 13: Commit einer Transaktion, OpenCPIC als Client

```
tx_open
tx_begin
Initialize_Conversation
Set_Sync_Level CM_SYNC_POINT
Allocate
Send_Data
Receive                                  --->    INIT PU
                                                                  KCENDTA=O
                                                                  KCSEND=Y
                                                 MGET             KCRST= (O,O)
                                                 MPUT NT
. . . CM_COMPLETE_DATA_RECEIVED          <---    PEND KP
Send_Data
Deallocate
tx_commit                                --->    INIT PU
                                                                  KCENDTA=F
                                                                  KCSEND=N
                                                 MGET             KCRST=(O,P)
. . . TX_OK                              <---    PEND FI
tx_close
```

Anmerkung: Beendet sich das zweite UTM-Anwendungsprogramm mit PEND FR anstelle von PEND FI, so erzwingt es damit ein Zurücksetzen der Transaktion (Rollback). In diesem Fall erhält das OpenCPIC-Anwendungsprogramm bei *tx_commit()* den Returnwert *TX_ROLLBACK*.

Beispiel 14:

Zwei Transaktionen nacheinander (Chained Transactions), OpenCPIC als Server

```
                                              INIT
                                              MGET
                                              APRO DM  > A  KCOF=C
                                              MPUT NT  > A
                                              CTRL PR  > A
tx_open                                <---   PEND KP
Accept_Conversation
Receive
Receive CM_TAKE_COMMIT__DATA_OK
Send_Data
tx_commit                              --->   INIT
                                              MGET NT  > A  KCRST=(O,P)
                                              MPUT NE
                                              PEND RE

                                              INIT
                                              MGET
                                              MPUT NT  > A
                                              CTRL PE  > A
Receive                                <--    PEND KP
Receive CM_TAKE_COMMIT_DEALLOCATE_DATA_OK
Send_Data
tx_set_transaction_control TX_UNCHAINED
tx_commit                              -->    INIT
tx_close                                      MGET NT  > A  KCRST=(C,P)
                                              MPUT NE  > A
                                              PEND FI
```

Beispiel 15:

Commit einer Transaktion mit Senderechtswechsel, OpenCPIC als Client

```
tx_open
tx_begin
Initialize_Conversation
Set_Sync_Level CM_SYNC_POINT
Allocate
Send_Data
Prepare_To_Receive
tx_commit                                ---->   INIT PU
                                                              KCENDTA=C
                                                              KCSEND=Y
                                                 MGET         KCRST=(O,P)
                                                 MPUT NT
. . . TX_OK                              <----   PEND RE
Receive CM_COMPLETE_DATA_RECEIVED
Receive CM_SEND_RECEIVED
... weiterer Programmlauf, z.B.
Fortzetzung wie Beispiel 13
```

Beispiel 16:

Zugangsschutz mit User-Id und Passwort:

```
Initialize_Conversation “SECU0001”
Set_Conversation_Security_Type
CM_SECURITY_PROGRAM
Set_Conversation_Security_User_ID
“CPIC1”
Set_Conversation_Security_Password
“12345678”                               --->    INIT
Allocate                                         MGET
Send_Data                                        NPUT NE
Receive                                  <---    PEND FI


. . . CM_COMPLETE_DATA_RECEIVED
Receive CM_DEALLOCATED_NORMAL
```

Für dieses Beispiel sind folgende spezielle Generierungen notwendig:

UTM-Generierung (KDCDEF

```
. . .
ACCESS-POINT . . .
OSI-CON . . . ,
    OSI-LPAP = OPCPIC1, . . .
OSI-LPAP OPCPIC1,
    APPLICATION-CONTEXT = UDTSEC
TAC SECU1, PROGRAM = . . .
USER CPIC1, PASS = 12345678
. . .
```

OpenCPIC-Generierung (*ocpic_gen*)

```
PARTAPPL UTMAW,
    APT = . . . ,
    APPL-CONTEXT = utm-secu

SYMDEST SECU0001,
    PARTNER-APPL = UTMAW,
    PARTNER-APRO = SECU1
```

7.3 OpenCPIC-Kommunikation mit Fremdsystemen

Prinzipiell ist eine Kommunikation zwischen OpenCPIC und Anwendungen anderer Hersteller möglich, wenn sie das OSI-TP-Protokoll benutzen. In diesem Fall muss aber genau geprüft werden, welchen Umfang von OSI-TP das Fremdsystem unterstützt.

In OpenCPIC erfolgt die Abbildung der OSI-TP-Protokollelemente auf CPI-C und umgekehrt wie in der X/Open Spezifikation der Schnittstelle CPI-C 2.0 (Anhang D) beschrieben.

Beachten Sie, dass Anwendungsprogramme auf Fremdsystemen auf einer ganz anderen Kommunikationsschnittstelle als CPI-C, beispielsweise CICS-API (Customer Information Control System - Application Programming Interface) realisiert sein können.

Folgende Punkte sind bei einer heterogenen Kopplung mit OpenCPIC zu beachten:

1. Das Fremdsystem muss mindestens die OSI-TP-Funktionseinheiten (Functional Units) *Dialogue* und *Polarized Control* implementiert haben. Eine Kommunikation mit Anwendungen, die nur *Dialogue* und *Shared Control* anbieten, ist nicht möglich.
   OpenCPIC unterstützt zudem die Funktionseinheiten *Handshake*, *Commit*, *Chained transactions* und *Unchained transactions*. Sollen diese genutzt werden, muss der Partner sie ebenfalls unterstützten.
2. OSI-TP geht davon aus, dass die Anwendungen beim Aufbau einer Association beliebige Datenstrukturen durch Auswahl eines U-ASE (User Application Service Element) vereinbaren können. OpenCPIC-Anwendungsprogramme können nur den Standardumfang der U-ASE UDT (Unstructured Data Transfer, ISO-10026-6, [32]) vereinbaren. Benutzerdaten werden als *OctetString* mit *UD-TRANSFER-RI* gesendet. Benutzte Datenstrukturen müssen privat zwischen den Anwendungsprogrammen vereinbart werden.
3. Wenn das Partnerprogramm alle Daten in EBCDIC-Darstellung senden und empfangen soll, das lokale Anwendungsprogramm jedoch nur den ASCII-Zeichensatz verwendet, so können zur Datenumwandlung die CPI-C-Aufrufe *Convert_Incoming (CMCNVI)* und *Convert_Outgoing (CMCNVO)* benutzt werden. Die Konvertierungstabellen finden Sie im Abschnitt „Code-Konvertierungstabellen“ auf Seite 200.
4. *UDT* definiert keine Maximallänge von Benutzerdaten. Mit OpenCPIC können jedoch nur Daten der Länge 0 bis 32767 Byte (32 KByte) gesendet und empfangen werden. Werden von einem Partnerprogramm Daten empfangen, die länger als 32 KByte sind, wird die Association zu diesem Partner abgebrochen.

5. OSI-TP erlaubt das Mitsenden von Benutzerdaten beim Dialogaufbau mittels *TP-BEGIN-DIALOGUE-request* bzw. *TP-BEGIN-DIALOGUE-confirm* (Initialisierungsdaten) und beim Dialogabbruch mittels *TP-ABORT-request* (Logdaten). OpenCPIC unterstützt dies nicht, d. h. ein OpenCPIC-Anwendungsprogramm kann keine Initialisierungs- bzw.Logdaten senden und ankommende Daten gehen verloren.

6. Die TPSU Title müssen auf beiden Seiten so gewählt werden, dass sie vom Partner aufgerufen werden können.

   In OpenCPIC kann ein entfernter TPSU Title (Partner-AP-Name) entweder per Generierung mit der SYMDEST-Anweisung (Parameter PARTNER-APRO und PARTNER-TYPE) oder mit dem CPI-C-Aufruf *Set_Partner_TP_Name (CMSTPN)* definiert werden. Die SYMDEST-Anweisung erlaubt TPSU Title vom Typ *PrintableString*, *T61String* und *Integer*. Wird der TPSU Title mit *CMSTPN* angegeben, erhält er automatisch den Typ *T61String*.

   Ein lokaler TPSU Title (lokaler Name eines AP) wird entweder per Generierung mit der LOCAPRO-Anweisung oder mit dem CPI-C-Aufruf S*pecify_Local_TP_Name (CMSLTP)* definiert. In beiden Fällen besteht der TPSU Title aus einer maximal 64 Zeichen langen Zeichenkette. OpenCPIC akzeptiert daher beim Empfang einer *TP-BEGIN-DIALOGUE-indication* nur TPSU Title vom Typ *PrintableString* oder *T61String* und der Maximallänge 64. Dialogaufbauwünsche für einen lokalen TPSU Title vom Typ *Integer* werden durch den OpenCPIC-Manager abgelehnt.

7. OSI-TP erlaubt, dass mehrere APDUs (Application Protocol Data Units) zu einer PSDU (Presentation Service Data Unit) verkettet werden können (Concatenation). Damit können prinzipiell beliebig lange PSDUs entstehen. OpenCPIC kann jedoch nur PSDUs bis zu einer Länge von 33 KByte bearbeiten. Wenn längere PSDUs ankommen, wird die Verbindung zu diesem Partner abgebrochen.
   Umgekehrt sendet OpenCPIC bis zu 33 KByte lange PSDUs.

   Eine OSI-TP-Implementierung von Fremdherstellern darf somit maximal 33 KByte lange PSDUs senden und muss ankommende PSDUs bis zu einer Länge von 33 KByte bearbeiten können, wenn sie mit OpenCPIC kommunizieren will.

8 Globale Transaktionen

Dieses Kapitel beschreibt Eigenschaften und Funktionen von OpenCPIC, die speziell im Zusammenhang mit globalen Transaktionen zum Tragen kommen. Für eine ausführliche Definition des Modells der globalen Transaktionen verweisen wir auf die X/Open Dokumente ([24],[24],[25],[26]).

8.1 Lokale Resource Manager - XA-Anschluss

Lokale Resource Manager steuern den Zugriff auf und die Modifikation von lokalen Ressourcen, wie z. B. Dateisystemen oder Datenbanken. Um globale Transaktionen im Sinne des X/Open DTP Referenzmodells durchführen zu können, ist es erforderlich, dass alle beteiligten Resource Manager das Zwei-Phasen-Commit-Protokoll beherrschen. Ein Resource Manager muss in der Lage sein, alle durchgeführten Modifikationen in lokalen Ressourcen bis zu einem bestimmten Zeitpunkt der Transaktion rückgängig zu machen. Erst wenn eine Bestätigung aller an der Transaktion beteiligten Kommunikationspartner erfolgt ist, werden die Modifikationen gültig.

X/Open hat zu diesem Zweck die Schnittstelle XA definiert. Über diese Schnittstelle kommunizieren der Transaction Manager (TM) und die TX-Bibliothek in OpenCPIC mit einem lokalen Resource Manager.

Die Implementierung der XA-Funktionen ist normalerweise Bestandteil des Resource Manager Produkts und wird üblicherweise in Form einer oder mehrerer Bibliotheken zur Verfügung gestellt. Diese Bibliotheken müssen zum OpenCPIC-Manager gebunden werden, damit der TM Zugriff auf den lokalen Resource Manager erhält. Wie Sie dies bewerkstelligen, ist nachfolgend beschrieben. Außerdem müssen die XA-Bibliotheken auch zu den lokalen Anwendungsprogrammen, die über den lokalen Resource Manager im Rahmen einer Transaktion auf lokale Ressourcen zugreifen wollen, gebunden werden. Dies ist im “Abschnitt „Anwendungsprogramme binden“ auf Seite 87“ beschrieben.

Prinzipiell ist es im X/Open DTP Referenzmodell möglich, dass eine Anwendung mit mehreren lokalen Resource Managern gleichzeitig arbeitet. OpenCPIC unterstützt jedoch derzeit nur einen Resource Manager pro Anwendung.

Die aktuell unterstützten Datenbank-Produkte entnehmen Sie bitte Ihrer Freigabemitteilung.

Die Datei *openCPIC-pfad/xa/ocpic_xa_connect* enthält den Anschluss für die XA-Schnittstelle in Form einer dynamischen Bibliothek. Diese Datei muss den XA-Anschluss für den aktuell benutzten Resource Manager enthalten. *ocpic_xa_connect* wird von *ocpic_mgr* zur Laufzeit nachgeladen, um den Resource-Manager anzusprechen.

Den Openstring für die Datenbank müssen Sie im Verzeichnis *openCPIC-pfad* in einer Datei mit dem Namen *op.<XA-Switch-Name>* hinterlegen. Den XA-Switch-Namen für die betreffenden Datenbank entnehmen Sie bitte der Datei *openCIPC-pfad/xa/xa_info.c.* Für die Datenbank Oracle lautet der Name z.B. *op.xaosw.*

8.1.1 Standard XA-Anschluss ohne Resource Manager

Mit *openCPIC-pfad/xa/xa_connection.without_rm* wird eine dynamische Bibliothek ausgeliefert, die ohne Anschluss zu einem Resource Manager erstellt wurde. Diese Bibliothek muss verwendet werden, falls OpenCPIC ohne Resource Manager betrieben wird.

Bei der Installation von OpenCPIC wird diese Bibliothek standardmäßig als dynamische Bibliothek *ocpic_xa_connect* zur Verfügung gestellt. D.h. OpenCPIC wird standardmäßig ohne Resource Manager betrieben.

8.1.2 Standard XA-Anschluss mit Test Resource Manager

Zusätzlich wird mit *openCPIC-pfad/xa/xa_connection.test_rm* eine dynamische Bibliothek ausgeliefert, die mit einem Anschluss zu einem Test Resource Manager erstellt wurde. Bei der Installation von OpenCPIC wird diese Bibliothek standardmäßig ausgeliefert.

Wenn Sie OpenCPIC mit diesem Test Resource Manager betreiben wollen, dann müssen Sie die Datei *ocpic_xa_connect* durch diese Bibliothek ersetzen.

8.1.3 Erzeugen des XA-Anschluss-Moduls

Falls OpenCPIC mit einem Resource Manager gekoppelt werden soll, muss ein XA-Anschlussmodul in Form einer dynamischen Bibliothek mit dem Namen *ocpic_xa_connect* erzeugt werden. Gehen Sie wie folgt vor:

1. Falls Sie einen anderen Resource Manager als Oracle, Informix oder MS SQL-Server verwenden, passen Sie den Namen des XA-Switches in *openCPIC-pfad/xa/xa_info.c* an.
2. Übersetzen Sie mit dem C-Compiler den Source-Code unter *openCPIC-pfad/xa/xa_info.c*.

Dabei müssen Sie die Präprozessor Variable XA_ SWITCH entsprechend ihrem XA Resource Manager setzen. Mögliche Werte sind:

XA_SWITCH=0	kein Resource Manager
XA_SWITCH=1	Oracle
XA_SWITCH=2	Informix
XA_SWITCH=3	MS SQL-Server
XA_SWITCH=4	anderer Resource Manager. In diesem Fall muss der Name des XA-Switches in *xa_info.c* angepasst werden.

3. Binden Sie dieses Objekt zusammen mit den XA-Bibliotheken des verwendeten Resource Managers zu einer dynamischen Bibliothek.
4. Stellen Sie diese Bibliothek unter *openCPIC-pfad/xa* mit dem Namen *ocpic_xa_connect* zur Verfügung.

Beim nächsten Start von *ocpic_mgr* koppelt OpenCPIC mit diesem Resource Manager.

Falls auf Ihrem System mehrere Resource Manager installiert sind, ist unbedingt darauf zu achten, dass der OpenCPIC-Manager mit demselben Resource Manager arbeitet wie die lokalen Anwendungsprogramme. Das heißt, die Programmdatei *ocpic_mgr* und die lokalen Anwendungsprogramme müssen denselben XA-Anschluss verwenden und denselben Resource Manager referenzieren.
Wie Sie lokale Anwendungsprogramme binden, ist im ”Abschnitt „Anwendungsprogramme binden“ auf Seite 87 beschrieben.

Dynamische Registrierung (xa-Aufrufe *ax_reg(), ax_unreg()*) wird von OpenCPIC nicht unterstützt.

8.2 Wiederanlauf (Recovery)

Die Fähigkeit zum Wiederanlauf (Recovery) nach Fehlersituationen (z. B. einem Programmausfall oder Systemabsturz) ist ein wichtiges Merkmal globaler Transaktionen. Sie gewährleistet ein hohes Maß an Datenkonsistenz und Ausfallsicherheit.

8.2.1 Log-Records

Zum Zwecke des Wiederanlaufs nach Fehlersituationen legt der OpenCPIC-Manager im Verlaufe einer Transaktion sogenannte Log-Records an. Ein Log-Record enthält neben einem eindeutigen Transaktions-Identifier (Transaction Id) Informationen über die beteiligten Resource Manager, den aktuellen Stand der Transaktion und aufgetretene Fehler. Der OpenCPIC-Manager speichert die Log-Records in Form von Dateien im Verzeichnis *$OCPICDIR/SYNC*. Beim Abschluss einer Transaktion werden die Log-Records gelöscht. Wird eine Transaktion vor ihrer Beendigung durch einen Programm- oder Systemfehler unterbrochen, bleibt der zugehörige Log-Record stehen. Beim Wiederanlauf (Warmstart des OpenCPIC-Managers) geben die vorhandenen Log-Records dem OpenCPIC-Manager Aufschluss über die zum Zeitpunkt des Fehlers offenen Transaktionen und deren Bearbeitungsstand.

Log-Record-Dateien sind Binärdateien. Ihr Format ist in der XAP-TP-Spezifikation [25] beschrieben.

8.2.2 Wiederanlaufmaßnahmen des OpenCPIC-Managers

Der OpenCPIC-Manager führt Wiederanlaufmaßnahmen nach folgenden Fehlersituationen durch:

1. Ausfall eines an einer Transaktion beteiligten Anwendungsprogramms

   Je nachdem, zu welchem Zeitpunkt der Transaktion der Ausfall auftrat, führt der OpenCPIC-Manager selbständig einen Commit oder Rollback durch. Befinden sich alle beteiligten Dialoge noch vor oder in der Precommit-Phase, wird immer ein Rollback durchgeführt. Ist das Anwendungsprogramm vor dem Ausfall bereits mit *tx_commit()* in die Commit-Phase eingetreten, hängt es davon ab, an welcher Stelle der Commit-Abwicklung der Ausfall erfolgte. Wenn die TX-Bibliothek dem lokalen Resource Manager bereits mit *xa_end()* das Ende der Transaktion bekannt gemacht hat, erfolgt ein Commit, andernfalls ein Rollback.
   Der OpenCPIC-Manager nimmt nach einem Programmausfall dem lokalen Resource Manager gegenüber die Stelle des Anwendungsprogramms ein. Um zu vermeiden, dass der OpenCPIC-Manager durch die XA-Aufrufe, die für die Beendigung der Transaktion nötig sind, unverhältnismäßig lange blockiert wird, werden diese Recovery-

Maßnahmen in einem separaten Prozess durchgeführt. Der OpenCPIC-Manager erzeugt dazu mit dem Systemaufruf *fork()* einen Kindprozess, welcher sich nach Abschluss der Transaktion selbständig beendet.

2. Ausfall des OpenCPIC-Managers, z. B. aufgrund eines Systemabsturzes oder bei Empfang des Signals *SIGKILL*

   Ist es während einer Transaktion zum Ausfall des OpenCPIC-Managers oder des gesamten Systems gekommen, so sollte anschließend ein Warmstart durchgeführt werden. Dazu rufen Sie *ocpic_start* **ohne** den Schalter *-c* auf (siehe “Abschnitt „OpenCPIC-Manager starten“ auf Seite 56”). Der OpenCPIC-Manager liest dann zunächst die Log-Records aus den Dateien im Verzeichnis *$OCPICDIR/SYNC* und führt aufgrund der gelesenen Informationen Wiederanlauf-Maßnahmen für alle zum Zeitpunkt des Systemausfalls aktiven Transaktionen durch. Es hängt wiederum vom Zeitpunkt des Systemausfalls innerhalb der Transaktion ab, ob eine Transaktion mit Commit oder Rollback beendet wird.
   Auch bei einem Warmstart erzeugt der OpenCPIC-Manager für jede zu beendende Transaktion mit *fork()* einen eigenen Kindprozess.
   Wenn die Recovery-Maßnahmen nach einem Warmstart beendet sind, gibt der OpenCPIC-Manager eine entsprechende Meldung in die Protokolldatei *$OCPICDIR/PROT/prot.mgr* aus.

Nach erfolgreicher Durchführung der Wiederanlaufmaßnahmen werden die Log-Record-Dateien im Verzeichnis *$OCPICDIR/SYNC* gelöscht. Ausgenommen davon sind sogenannte Log-Damage-Records, die auf mögliche Inkonsistenzen bei der Transaktionsbeendigung (heuristische Entscheidungen) hinweisen (siehe nachfolgenden Abschnitt „Heuristische Entscheidungen“).

Da bei jedem Warmstart des OpenCPIC-Managers **alle** Log-Records im Verzeichnis *$OCPICDIR/SYNC* bearbeitet werden, sollten von Zeit zu Zeit nicht mehr benötigte Log-Damage-Records durch den Systemadministrator gelöscht bzw. in einem anderen Verzeichnis gesichert werden, um unnötig lange Wiederanlaufzeiten zu vermeiden.

8.2.3 Heuristische Entscheidungen

Falls es infolge heuristischer Entscheidungen (siehe Seite 27) zu Inkonsistenzen beim Transaktionsabschluss kommt, wird dies den beteiligten Transaction Managern durch das OSI-TP-Protokollelement *TP-HEURISTIC-REPORT-indication* oder die XAP-TP-Primitive *TP_LOG_DAMAGE_IND* mitgeteilt.

Der OpenCPIC-Manager gibt daraufhin eine entsprechende Meldung (425) in die Protokolldatei aus und sichert einen sogenannten Log-Damage-Record im Verzeichnis *$OCPICDIR/SYNC*. Da der binäre Log-Damage-Record nicht ohne Weiteres lesbar ist, gibt der OpenCPIC-Manager gleichzeitig die wichtigsten Informationen in eine separate ASCII-Log-Datei aus.

Den Anwendungsprogrammen wird eine heuristische Entscheidung im Transaktionsbaum durch den Returnwert beim *tx_commit()* bzw. *tx_rollback()* mitgeteilt. Bitte informieren Sie sich hierzu in der Beschreibung der TX-Schnittstelle [26].

Auswerten der Log-Damage-Informationen des OpenCPIC-Managers

Die entstandenen Inkonsistenzen müssen i. a. durch geeignete Maßnahmen des Systemadministrators behoben werden. Dazu stehen folgende Informationsquellen zur Verfügung:

– Die Log-Dateien des Resource Managers. Hinweise dazu finden Sie in Ihrer Resource Manager Dokumentation.

– Die Log-Damage-Records des OpenCPIC-Managers. Die enthaltenen Informationen finden Sie in lesbarer Form in der zugehörigen ASCII-Log-Datei.

Die Namen der binären Log-Record-Dateien sind nach folgendem Schema aufgebaut:

log.trans_id

trans_id ist eine aus mehreren Bestandteilen zusammengesetzte eindeutige Transaktionskennung.
Jedem binären Log-Damage-Record ist eine ASCII-Log-Datei mit folgendem Namen zugeordnet:

logasc.trans_id

Dabei ist *trans_id* für zusammengehörige Dateien gleich.

Die Informationen aus dem Log-Damage-Record können eine wichtige Hilfe zur Behebung der Inkonsistenzen darstellen. Löschen Sie diese Dateien daher erst, wenn Sie sicher sind, dass sich alle Ressourcen wieder in einem konsistenten Zustand befinden.

Format einer ASCII-Log-Datei

```
ASCII-Daten aus der Log-Damage-Record-Datei %s


XID
     format ID       : %d
     gtrid_length    : %d
     bqual_length    : %d
     data            : %s


Lokaler Resource Manager
     RM ID         : %s
     Open string   : %s
     Close string  : %s


Lokaler Knoten
     AP PID        : %d
     AEQ           : %d
     APT           : %s
     Node State    : %d
     Log Damage State: %d
     Superior      : %d
     Subordinates  : %d


Superior Knoten
     AEQ           : %d
     APT           : %s


Subordinate-Knoten
     AEQ           : %d
     APT           : %s
     Log Damage State: %d
```

Bedeutung der Ausgabefelder

Log-Damage-Record-Datei
: Name der zugehörigen binären Log-Damage-Record-Datei

XID
: XID der betreffenden Transaktion. Die Struktur einer XID ist in der Datei *tx.h* definiert (siehe auch [26]). Sie enthält folgende Elemente:

format ID
: Format-Identifikator

gtrid_length	Länge des Global Transaction Identifiers in Byte (1-64) Der Global Transaction Identifier beginnt mit dem ersten Byte des nachfolgenden Datenteils *data*.
bqual_length	Länge des Branch Qualifiers in Byte (1-64) Der Branch Qualifier beginnt unmittelbar nach dem letzten Byte des Global Transaction Identifiers im nachfolgenden Datenteil *data*.
data	Datenteil, bestehend aus Global Transaction Identifier und Branch Identifier der jeweils angegebenen Länge. Die Bytes des Datenteils werden in hexadezimaler Form ausgegeben.

Lokaler Resource Manager

Dieser Abschnitt enthält Informationen zum lokalen Resource Manager.

RM ID	Interner Identifikator des Resource Managers
Open String	XA-Open-String des Resource Managers
Close String	XA-Close-String des Resource Managers

Lokaler Knoten — Dieser Abschnitt enthält Informationen zum lokalen Knoten des Transaktionsbaums.

AP PID	Prozess-Id des Anwendungsprogramms
AEQ	Application Entity Qualifier der lokalen Anwendung
APT	Application Process Title der lokalen Anwendung
Node State	kann einen der folgenden Werte annehmen:
= 0	AP_TP_NONE Die Transaktion befand sich weder im Ready-Zustand (Phase 1 des Zwei-Phasen-Commit-Protokolls) noch im Commit-Zustand (Phase 2 des Zwei-Phasen-Commit-Protokolls), als die heuristische Entscheidung gemeldet wurde.
= 1	AP_TP_READY Die Transaktion befand sich im Ready-Zustand (Phase 1 des Zwei-Phasen-Commit-Protokolls), als die heuristische Entscheidung gemeldet wurde.
= 2	AP_TP_COMMIT Die Transaktion befand sich im Commit-Zustand (Phase 2 des Zwei-Phasen-Commit-Protokolls), als die heuristische Entscheidung gemeldet wurde.

Log Damage State
kann einen der folgenden Werte annehmen:

= 1 AP_TP_HEUR_MIX
Es wurde eine “Heuristic Mix Condition” im Teilbaum unterhalb des lokalen Knotens gemeldet.

= 2 AP_TP_HEUR_HAZ
Es wurde eine “Heuristic Hazard Condition” im Teilbaum unterhalb des lokalen Knotens des lokalen Knotens gemeldet.

Superior gibt an, ob der lokale Knoten einen Superior besitzt.

= 0 Der lokale Knoten besitzt keinen Superior. Er ist somit der Root-Knoten der Transaktion.

= 1 Der lokale Knoten besitzt einen Superior. Er ist somit ein Intermediate- oder Leaf-Knoten der Transaktion.

Subordinates gibt an, wieviel Subordinate-Knoten der lokale Knoten besitzt.

Superior-Knoten Dieser Abschnitt erscheint nur, wenn der lokale Knoten einen Superior besitzt. Er enthält Informationen zum Superior-Knoten.

AEQ Application Entity Qualifier des Superior-Knotens

APT Application Process Title des Superior-Knotens

Subordinate-Knoten Dieser Abschnitt erscheint einmal pro Subordinate. Er enthält Informationen zum jeweiligen Subordinate-Knoten.

AEQ Application Entity Qualifier des Subordinate-Knotens

APT Application Process Title des Subordinate-Knotens

Log Damage State
kann einen der folgenden Werte annehmen

= 0 AP_TP_NONE
Im Teilbaum unterhalb des Subordinate trat keine heuristische Entscheidung ” auf.

= 1 AP_TP_HEUR_MIX
Es wurde eine “Heuristic Mix Condition” im Teilbaum unterhalb des Subordinate gemeldet.

= 2 AP_TP_HEUR_HAZ
Es wurde eine “Heuristic Hazard Condition” im Teilbaum unterhalb des Subordinate gemeldet.

9 Meldungen der OpenCPIC-Programme

Dieses Kapitel enthält alle Meldungen der OpenCPIC-Programme sowie deren Bedeutung und evtl. durchzuführende Maßnahmen.

Meldungen des OpenCPIC-Managers werden normalerweise in die Datei *prot.mgr* im Verzeichnis *$OCPICDIR/PROT* (bzw. *openCPIC-pfad/PROT*, falls *OCPICDIR* nicht gesetzt ist) ausgegeben. Nur während seiner Initialisierungsphase gibt der OpenCPIC-Manager Meldungen direkt am Bildschirm aus.

Die Programme *ocpic_gen*, *ocpic_adm*, *ocpic_sta*, *ocpic_logdump* und *xatmigen* geben ihre Meldungen nach *stderr* aus.

OpenCPIC verwendet das Verfahren NLS (Native Language Support) zur sprachabhängigen Meldungsausgabe. Es ist eine Ausgabe in deutscher oder englischer Sprache möglich. Die Steuerung erfolgt über die Umgebungsvariable *$LANG*. Ist *$LANG* nicht gesetzt oder ungültig, erfolgt die Ausgabe in englisch.
Die NLS-Meldungskataloge werden bei der Installation in den Verzeichnissen */opt/lib/nls/msg/De* und */opt/lib/nls/msg/En* abgelegt.

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.

Bei der Beschreibung der Meldungen ist mit *$OCPICDIR* immer der Wert der Umgebungsvariablen *OCPICDIR* gemeint, bzw.das Standardverzeichnis *openCPIC-pfad*, falls diese Variable nicht gesetzt ist.

9.1 Meldungen des OpenCPIC-Managers

Alle Meldungen des OpenCPIC-Managers enthalten eine Kennung, die die Komponente des OpenCPIC-Managers angibt, von der die Meldung stammt. Die Kennungen haben folgende Bedeutung:

Kennung	Meldungsnummern	Komponente
BD	001 - 299	Boundary (IPC-Handler, Schnittstelle zum XAP-TP-Baustein)
TPM	300 - 399	TP-Manager (Communication Resource Manager)
TM	400 - 499	TM (Transaction Manager)

Die Meldungen des XAP-TP-Bausteins werden von der Komponente BD ausgegeben, jeweils eingeleitet durch die Meldung 100. Sie besitzen jedoch eigene Meldungsnummern und werden daher in einem separaten Abschnitt (Seite 177) aufgelistet.

9.1.1 Meldungen der Komponente BD

```
OpenCPIC [BD] 001 Start des OpenCPIC-Managers mit PID &MGRPID
```

Bedeutung
Initialisierungsmeldung des OpenCPIC-Managers. Sie wird unmittelbar nach dem Start ausgegeben und enthält die PID *&MGRPID* des OpenCPIC-Managers.

```
OpenCPIC [BD] 002 Version: &VERSION
```

Bedeutung
Initialisierungsmeldung des OpenCPIC-Managers. Sie enthält die aktuelle Versionsnummer *&VERSION* des OpenCPIC-Managers.

```
OpenCPIC [BD] 003 Trace-Option(en) = &TROPTS
```

Bedeutung
Initialisierungsmeldung des OpenCPIC-Managers. Diese Meldung gibt die beim Start gültigen Traceeinstellungen (*&TROPTS*) aus (siehe Seite 56).

```
OpenCPIC [BD] 004 Aufrufsyntax: ocpic_start [-tp] [-tm] [-c]
          -tp : mit XAP-TP-Trace
          -tm: mit Manager-Trace
          -c: Kaltstart
```

Bedeutung
SIe haben den OpenCPIC-Manager mit ungültigen Schaltern oder Parametern aufgerufen.

Maßnahme
Informieren Sie sich im “Abschnitt „OpenCPIC-Manager starten“ auf Seite 56” über die korrekte Aufrufsyntax.

`OpenCPIC [BD] 007 Fehler beim Verzeichniswechsel nach &DIRNAME`

Bedeutung
Der OpenCPIC-Manager kann nicht in das Arbeitsverzeichnis *&DIRNAME* (*$OCPICDIR* bzw. *openCPIC-pfad*, wenn *OCPICDIR* nicht gesetzt ist) wechseln.

Maßnahme
Überprüfen Sie den Inhalt der Umgebungsvariablen *OCPICDIR* und vergewissern Sie sich, ob das angegebene Verzeichnis existiert. Starten Sie dann den OpenCPIC-Manager erneut.

`OpenCPIC [BD] 008 Abbruch wegen Speicherplatzmangels`

Bedeutung
Der Versuch, temporären Speicherplatz zu beschaffen, ist fehlgeschlagen. Aufgrund dessen beendet sich der OpenCPIC-Manager.

Maßnahme
Reduzieren Sie den Speicherverbrauch anderer Programme oder vergrößern Sie den Hauptspeicher Ihres Systems. (Den Speicherbedarf des OpenCPIC-Managers können Sie reduzieren, indem Sie mit beim Parameter ASSOCIATIONS in den PARTAPPL-Anweisungen weniger Associations generieren.) Starten Sie den OpenCPIC-Manager erneut.

`OpenCPIC [BD] 009 OpenCPIC-Manager laeuft bereits`

Bedeutung
Sie haben den OpenCPIC-Manager bereits gestartet. Es darf nur ein OpenCPIC-Manager in einer Arbeitsumgebung laufen. Wollen Sie mehrere OpenCPIC-Manager auf einem System starten, müssen Sie jeden OpenCPIC-Manager in einer eigenen Umgebung starten und die Umgebungsvariable *OCPICDIR* für jeden OpenCPIC-Manager unterschiedlich setzen.

Maßnahme
Mit *ocpic_sta* können Sie sich den aktuellen Zustand des OpenCPIC-Managers anschauen. Das Kommando *ps* zeigt Ihnen die momentan aktiven Prozesse an. Wenn Sie den OpenCPIC-Manager in derselben Arbeitsumgebung neu starten wollen, beenden Sie den laufenden zunächst mit *ocpic_adm -e* oder mit *kill -15*. Wollen Sie einen anderen OpenCPIC-Manager parallel betreiben, wechseln Sie die Arbeitsumgebung.
Sollte diese Meldung erscheinen, obwohl tatsächlich kein Prozess mit dem Namen *ocpic_mgr* in Ihrer Arbeitsumgebung aktiv ist, schauen Sie bitte im Verzeichnis *$OCPICDIR/PROT* nach, ob es dort eine Datei namens *pid_mgr* gibt. Löschen Sie diese ggf. und starten Sie den OpenCPIC-Manager erneut.

```
OpenCPIC [BD] 010 Anlegen der Datei pid_mgr mißlungen, errno = &ERRNO
```

Bedeutung
Der OpenCPIC-Manager kann eine seiner Systemdateien im Verzeichnis *$OCPICDIR/PROT* nicht anlegen. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Überprüfen Sie den Inhalt der Umgebungsvariablen *OCPICDIR* und vergewissern Sie sich, ob das angegebene Verzeichnis existiert. Starten Sie dann den OpenCPIC-Manager erneut.

```
OpenCPIC [BD] 011 Fehler beim Oeffnen der Input-Pipe, errno = &ERRNO
```

Bedeutung
Die Named Pipe *$OCPICDIR/.pipes/.pipe_ocpm* kann nicht geöffnet werden. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.
Wenn die Input-Pipe nicht geöffnet werden kann, ist keine Kommunikation zwischen OpenCPIC-Manager und den Anwendungsprogrammen möglich und der OpenCPIC-Manager beendet sich.

Maßnahme
Abhängig von *&ERRNO*. Verständigen Sie ggf. Ihren Systemverwalter.

```
OpenCPIC [BD] 012 Fehler beim Oeffnen der Konfigurationsdatei, errno = &ERRNO
```

Bedeutung
Die Konfigurationsdatei *$OCPICDIR/conffile* kann nicht geöffnet werden. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Erzeugen Sie ggf. mit *ocpic_gen* (siehe Seite 50) eine neue Konfigurationsdatei und starten Sie dann den OpenCPIC-Manager erneut.

```
OpenCPIC [BD] 013 Fehler beim Lesen der Konfigurationsdatei, errno = &ERRNO
```

Bedeutung
Die Konfigurationsdatei *$OCPICDIR/conffile* kann nicht gelesen werden. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Erzeugen Sie ggf. mit *ocpic_gen* (siehe Seite 50) eine neue Konfigurationsdatei und starten Sie dann den OpenCPIC-Manager erneut.

```
OpenCPIC [BD] 014 Falsche Version in der Konfigurationsdatei
```

Bedeutung
Die mit *ocpic_gen* erzeugte Konfigurationsdatei *conffile* enthält eine falsche Versionsnummer. Sie wurde mit einer Version des Programms *ocpic_gen* erstellt, die nicht zum OpenCPIC-Manager passt.

Maßnahme
Rufen Sie *ocpic_gen* (siehe Seite 50) auf und starten Sie dann den OpenCPIC-Manager erneut.

`OpenCPIC [BD] 015 Es konnten nur &NUMBER von den gewünschten Instanzen erzeugt werden.`

Bedeutung
Während der Initialisierungsphase versucht der OpenCPIC-Manager, pro gewünschter Association eine XAP-TP-Dialog-Instanz zu erzeugen (zuzüglich einer Kontroll-Instanz für die Überwachung von Commitment, Rollback und Recovery). Die Anzahl der zu erzeugenden Dialog-Instanzen ergibt sich aus der Summe aller gewünschten Associations (Parameter ASSOCIATIONS) aller konfigurierten Partner (PARTAPPL-Anweisungen). Die Meldung besagt, dass nicht alle gewünschten Instanzen vom XAP-TP-Baustein bereitgestellt wurden. Dies kann auf einen internen Fehler im XAP-TP-Baustein hinweisen.

Maßnahme
Sichern Sie den Inhalt des Verzeichnisses *$OCPICDIR/PROT*, insbesondere die Protokolldateien des XAP-TP-Trace, und verständigen Sie Ihren Service.

`OpenCPIC [BD] 016 Initialisierung des Managers abgeschlossen`

Bedeutung
Wenn diese Meldung am Bildschirm erscheint, hat der OpenCPIC-Manager seine Initialisierungsphase beendet und ist nun bereit, Anforderungen von lokalen Anwendungsprogrammen oder entfernten Anwendungen entgegenzunehmen.

`OpenCPIC [BD] 017 OpenCPIC-Manager beendet`

Bedeutung
Diese Meldung wird als letzte in die Protokolldatei geschrieben, wenn sich der OpenCPIC-Manager normal (d. h. aufgrund von *ocpic_adm -e* oder *kill -15*) beendet.

`OpenCPIC [BD] 018 Abbruch des OpenCPIC-Managers im Modul &MNAME , Error Nummer = &ERRNUM, Error Code = &ERRCODE`

Bedeutung
Der OpenCPIC-Manager hat sich aufgrund eines schwerwiegenden internen Fehlers beendet.

Maßnahme
Sichern Sie den Inhalt des Verzeichnisses *$OCPICDIR/PROT* und starten Sie den OpenCPIC-Manager erneut. Verständigen Sie Ihren Service.

`OpenCPIC [BD] 019 Abbruch durch den XAP-TP-Provider: &NAME`

Bedeutung
Der OpenCPIC-Manager hat sich aufgrund eines schwerwiegenden Fehlers im XAP-TP-Baustein beendet.

Maßnahme
Sichern Sie den Inhalt des Verzeichnisses *$OCPICDIR/PROT* und starten Sie den OpenCPIC-Manager erneut. Verständigen Sie Ihren Service.

```
OpenCPIC [BD] 020 Fehler bei Signalbehandlung; errno = &ERRNO
```

Bedeutung
Es trat ein interner Fehler bei der Initialisierung der Signalbehandlung des OpenCPIC-Managers auf. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Verständigen Sie ggf. Ihren Systemverwalter.

```
OpenCPIC [BD] 025 Die synchrone Pipe zum AP mit PID &APPID ist voll
```

Bedeutung
Bei der Kommunikation zwischen OpenCPIC-Manager und einem lokalen Anwendungsprogramm ist es zum Überlauf der Named Pipe gekommen. Der OpenCPIC-Manager versucht nach Ablauf von 1 Sekunde, erneut in die Pipe zu schreiben.

Maßnahme
Sollte diese Meldung gehäuft auftreten, prüfen Sie, ob das lokale AP mit der angegebenen PID noch aktiv ist.

```
OpenCPIC [BD] 030 Dump-Datei &FILENAME kann nicht geöffnet werden
```

Bedeutung
Der OpenCPIC-Manager hat versucht aufgrund eines internen Fehlers oder eines Administrationskommandos, eine Dump-Datei *&FILENAME* anzulegen. Die Datei konnte nicht geöffnet werden. Dies bedeutet, dass dem Service keine Dump-Datei zur Fehlerdiagnose zur Verfügung steht.

```
OpenCPIC [BD] 035 Unbekannte interne Meldung empfangen
OpenCPIC [BD] 040 Zu lange interne Meldung (&NUMBER Bytes) empfangen
```

Bedeutung
Diese Meldungen deuten auf einen Fehler bei der Kommunikation zwischen dem OpenCPIC-Manager und den lokalen Anwendungsprogrammen bzw. Administrationsprogrammen hin.

Maßnahme
Sichern Sie den Inhalt des Verzeichnisses *$OCPICDIR/PROT* und verständigen Sie Ihren Service.

```
OpenCPIC [BD] 045 Read an Input-Pipe liefert Fehler ENOBUFS
```

Bedeutung
Beim Lesen mit *read()* aus einer Named Pipe trat der Fehler *errno = NOBUFS* auf. Das bedeutet, es besteht ein momentaner Speicherengpass. Der OpenCPIC-Manager wiederholt den Leseversuch.

Maßnahme
Bei häufigem Auftreten dieser Meldung bzw. wenn sich der OpenCPIC-Manager aufgrund der Überschreitung der Maximalzahl von 25 Wiederholungen beendet, verständigen Sie Ihren Systemverwalter.

```
OpenCPIC [BD] 046 Write in Pipe zum AP mit PID &APPID liefert Fehler ENOBUFS
```

Bedeutung
Beim Schreiben mit *write()* in eine Named Pipe trat der Fehler *errno = ENOBUFS* auf. Das bedeutet, es besteht ein momentaner Speicherengpass. Der OpenCPIC-Manager wiederholt den Schreibversuch.

Maßnahme
Bei häufigem Auftreten dieser Meldung bzw. wenn sich der OpenCPIC-Manager aufgrund der Überschreitung der Maximalzahl von 25 Wiederholungen beendet, verständigen Sie Ihren Systemverwalter.

```
OpenCPIC [BD] 050 ap_set_env fuer Instanz &INSTNO fehlgeschlagen, ap_errno = &APERRNO:
          &ERRTXT
OpenCPIC [BD] 051 ap_get_env fuer Instanz &INSTNO fehlgeschlagen, ap_errno = &APERRNO:
          &ERRTXT
OpenCPIC [BD] 052 ap_free fuer Instanz &INSTNO fehlgeschlagen, ap_errno = &APERRNO: &ERRTXT
OpenCPIC [BD] 053 ap_bind fuer Instanz &INSTNO fehlgeschlagen, ap_errno = &APERRNO: &ERRTXT
OpenCPIC [BD] 054 ap_close fuer Instanz &INSTNO fehlgeschlagen, ap_errno = &APERRNO:
          &ERRTXT
OpenCPIC [BD] 055 ap_init_env fuer Instanz &INSTNO fehlgeschlagen, mit ap_errno = &APERRNO:
          &ERRTXT
OpenCPIC [BD] 056 ap_open fuer Instanz &INSTNO fehlgeschlagen, ap_errno = &APERRNO: &ERRTXT
OpenCPIC [BD] 057 ap_rcv fuer Instanz &INSTNO fehlgeschlagen, ap_errno = &APERRNO: &ERRTXT
OpenCPIC [BD] 058 ap_snd fuer Instanz &INSTNO fehlgeschlagen, ap_errno = &APERRNO: &ERRTXT
OpenCPIC [BD] 059 ap_poll fuer Instanz &INSTNO fehlgeschlagen, ap_errno = &APERRNO: &ERRTXT
OpenCPIC [BD] 060 apext_adm (operation code &OPCODE fehlgeschlagen, ap_errno = &APERRNO:
          &ERRTXT
OpenCPIC [BD] 061 apext_att fehlgeschlagen, ap_errno = &APERRNO: &ERRTXT
OpenCPIC [BD] 062 apext_det fehlgeschlagen, ap_errno = &APERRNO: &ERRTXT
OpenCPIC [BD] 063 apext_init fuer Typ &INITTYPE fehlgeschlagen, ap_errno = &APERRNO:
          &ERRTXT
```

Bedeutung
Diese Meldungen deuten auf einen internen Fehler beim Zugriff auf die XAP-TP-Schnittstelle hin.

&INSTNO	Nummer der XAP-TP-Instanz
&APERRNO	XAP-TP-Fehlernummer
&ERRTXT	XAP-TP-Fehlertext

Maßnahme
Sichern Sie den Inhalt des Verzeichnisses *$OCPICDIR/PROT* und verständigen Sie Ihren Service.

`OpenCPIC [BD] 070 Lokales AP &APNAME kann nicht gestartet werden: Programm nicht vorhanden`

Bedeutung
Der OpenCPIC-Manager hat versucht, aufgrund eines empfangenen Dialogaufbauwunsches, das lokal konfigurierte Anwendungsprogramm *&APNAME* zu starten. Das Programm, welches Sie in der Generierungsanweisung LOCAPRO (Parameter PATH) angegeben haben, existiert nicht auf Ihrem System.

Maßnahme
Installieren Sie das entsprechende Anwendungsprogramm unter dem angegebenen Pfadnamen oder korrigieren Sie Ihre Generierungsdatei. Im zweiten Fall müssen Sie die Konfigurationsdatei mit *ocpic_gen* neu erzeugen und den OpenCPIC-Manager neu starten.

`OpenCPIC [BD] 071 Lokales AP &APNAME kann nicht gestartet werden: Login Name &LGNAME nicht vorhanden`

Bedeutung
Der OpenCPIC-Manager hat versucht, aufgrund eines empfangenen Dialogaufbauwunsches, das lokal konfigurierte Anwendungsprogramm *&APNAME* unter der Benutzerkennung *&LGNAME* zu starten. Die Benutzerkennung, die Sie in der Generierungsanweisung LOCAPRO (Parameter LOGIN) angegeben haben, existiert nicht auf Ihrem System.

Maßnahme
Tragen Sie die Benutzerkennung auf Ihrem lokalen System ein oder korrigieren Sie Ihre Generierungsdatei. Im zweiten Fall müssen Sie die Konfigurationsdatei mit *ocpic_gen* neu erzeugen und den OpenCPIC-Manager neu starten.

`OpenCPIC [BD] 072 Lokales AP &APNAME kann nicht gestartet werden: Systemaufruf fork() fehlgeschlagen, errno = &ERRNO`

Bedeutung
Der OpenCPIC-Manager hat versucht, aufgrund eines empfangenen Dialogaufbauwunsches, das lokal konfigurierte Anwendungsprogramm *&APNAME* zu starten. Beim Start eines Kindprozesses mit *fork()* hat es einen Systemfehler gegeben. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Verständigen Sie ggf. Ihren Systemverwalter.

```
OpenCPIC [BD] 073 Lokales AP &APNAME kann nicht gestartet werden: Verzeichniswechsel ins
          HOME-Verzeichnis &DIRNAME fehlgeschlagen, errno = &ERRNO
```

Bedeutung
Der OpenCPIC-Manager hat aufgrund eines empfangenen Dialogaufbauwunsches versucht, das lokal konfigurierte Anwendungsprogramm *&APNAME* zu starten. Im Kindprozess ist der Versuch, mit *chdir* in das *HOME*-Verzeichnis *&DIRNAME* der konfigurierten Benutzerkennung (Parameter LOGIN in der Generierungsanweisung LOCAPRO) zu wechseln, fehlgeschlagen. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Ändern Sie ggf. die Zugriffsrechte.

```
OpenCPIC [BD] 074 Lokales AP &APNAME kann nicht gestartet werden:
          Set group Id to &GRPNAME fehlgeschlagen, errno = &ERRNO
```

Bedeutung
Der OpenCPIC-Manager hat aufgrund eines empfangenen Dialogaufbauwunsches das lokal konfigurierte Anwendungsprogramm *&APNAME* mittels *fork()* gestartet. Im Kindprozess ist der Versuch, mit *setgid()* die Benutzergruppe entsprechend der konfigurierten Benutzerkennung *&GRPNAME* (Parameter LOGIN in der Generierungsanweisung LOCAPRO) zu setzen, fehlgeschlagen. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Verständigen Sie ggf. Ihren Systemverwalter.

```
OpenCPIC [BD] 075 Lokales AP &APNAME kann nicht gestartet werden:
          Set user Id to &USRNAME fehlgeschlagen, errno = &ERRNO
```

Bedeutung
Der OpenCPIC-Manager hat aufgrund eines empfangenen Dialogaufbauwunsches versucht, das lokal konfiguriertes Anwendungsprogramm zu starten. Im Kindprozess ist der Versuch, mit *setuid()* die Benutzer-Id entsprechend der konfigurierten Benutzerkennung (Parameter LOGIN in der Generierungsanweisung LOCAPRO) zu setzen, fehlgeschlagen. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Verständigen Sie ggf. Ihren Systemverwalter.

```
OpenCPIC [BD] 076 Lokales AP &APNAME kann nicht gestartet werden: execve fehlgeschlagen,
          errno = &ERRNO
```

Bedeutung
Der OpenCPIC-Manager hat aufgrund eines empfangenen Dialogaufbauwunsches versucht, das lokal konfigurierte Anwendungsprogramm *&APNAME* zu starten. Im Kindprozess ist der Versuch, mit *execve()* das Programm (Parameter PATH in der Generierungsanweisung LOCAPRO) zu starten, fehlgeschlagen. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Prüfen Sie, ob die Ausführungsrechte für Ihr lokales Anwendungsprogramm richtig vergeben sind.

```
OpenCPIC [BD] 077 Kindprozeß beendet sich
```

Bedeutung
Diese Meldung erscheint nach einer der Meldungen 073 - 076. Aufgrund eines Fehlers beim Starten eines lokalen Anwendungsprogramms wird der mittels *fork()* erzeugte Kindprozess wieder beendet. Über die Fehlerursache gibt die vorhergehende Meldung Auskunft.

```
OpenCPIC [BD] 100 Meldung &XAPNUM des XAP-TP-Providers: &MSGNUM ...
```

Bedeutung
Diese Meldung leitet eine interne Systemmeldung des XAP-TP-Bausteins ein. Der Folgetext ist abhängig von der Meldungsnummer *&MSGNUM*. Die XAP-TP-Baustein-Meldungen sind in einem separaten Abschnitt (Seite 177) aufgelistet.
&XAPNUM interne Fehlernummer
&MSGNUM Nummer der Meldung

9.1.2 Meldungen der Komponente TPM

```
OpenCPIC [TPM] 301 CPI-C-Request von unbekanntem Absender (PID &APPID) empfangen
OpenCPIC [TPM] 302 CPI-C-Request mit unbekannter Conversation ID &CONVID empfangen von
          Absender
          mit PID &APPID
```

Bedeutung
Der OpenCPIC-Manager hat einen CPI-C-Request von einem unbekannten Anwendungsprogramm und/oder mit einer ungültigen Conversation Id erhalten. Dies deutet auf einen internen Fehler hin.
&APPID PID des Anwendungsprogramms
&CONVID Conversation-Id

Maßnahme
Prüfen Sie, ob Ihr Anwendungsprogramm mit der aktuellen Version der Bibliothek *libocpic.a* gebunden wurde. Starten Sie das Anwendungsprogramm erneut. Bei wiederholtem Auftreten schalten Sie den CPI-C- und den Manager-Trace ein. Sichern Sie den Inhalt des Verzeichnisses *$OCPICDIR/PROT* und verständigen Sie Ihren Service.

```
OpenCPIC [TPM] 303 Conversation kann nicht aufgebaut werden, maximale Anzahl von
           Associations erreicht
           (AP PID &APPID, Conversation ID &CONVID)
```

Bedeutung
Es ist bereits die maximal mögliche Anzahl von Associations aufgebaut und alle Associations sind momentan belegt. Da keine weitere Association aufgebaut werden kann, kann der OpenCPIC-Manager zur Zeit keine Conversation aufbauen.
&APPID PID des Anwendungsprogramms
&CONVID Conversation Id

Maßnahme
Die Anzahl der maximal möglichen Associations generieren Sie in der PARTAPPL-Anweisung mit dem Parameter ASSOCIATIONS. Wenn Sie diese Anzahl erhöhen wollen, müssen Sie die Generierungsdatei ändern, mit *ocpic_gen* eine neue Konfigurationsdatei erzeugen und den OpenCPIC-Manager anschließend neu starten.

```
OpenCPIC [TPM] 304 Conversation kann nicht aufgebaut werden
           Grund: Associationswunsch abgelehnt
           (AP PID &APPID, Conversation ID &CONVID)
```

Bedeutung
Der OpenCPIC-Manager hat aufgrund eines *Allocate (CMALLC)* eines lokalen AP mit *return_control = CM_IMMEDIATE* versucht, eine Association zu belegen, und nicht sofort eine Association vom XAP-TP-Baustein erhalten. Der *CMALLC* wird deshalb mit *CM_UNSUCCESSFUL* beantwortet.
&APPID PID des Anwendungsprogramms
&CONVID Conversation ID

```
OpenCPIC [TPM] 311 Allocate() von lokalem Anwendungsprogramm (PID &APPID) abgelehnt
           Partneranwendung: &PARTNAME
           Grund: Partneranwendung nicht konfiguriert
```

Bedeutung
Ein lokales AP hat versucht, mit *Allocate (CMALLC)* eine Conversation zu einer Partneranwendung aufzubauen, die nicht konfiguriert ist. Sie haben entweder beim *Initialize_Conversation (CMINIT)* einen falschen *symdest_name* angegeben oder mit *Set_Partner_LU_Name (CMSPLN)* eine nicht generierte Partneranwendung angegeben.
&APPID PID des lokalen Anwendungsprogramms
&PARTNAME Name der gewünschten Partneranwendung

Maßnahme
Korrigieren Sie Ihr Anwendungsprogramm oder generieren Sie die gewünschte Partneranwendung. Wenn Sie Ihre Generierung ändern, müssen Sie mit *ocpic_gen* eine neue Konfigurationsdatei erzeugen sowie den OpenCPIC-Manager beenden und anschließend neu starten.

```
OpenCPIC [TPM] 313 Allocate() von lokalem Anwendungsprogramm (PID &APPID, Conversation ID
          &CONVID)
          wird abgelehnt
          Grund: Aktuelle Transaktion wird beendet
```

Bedeutung
Ein lokales AP hat versucht, mit *Allocate (CMALLC)* eine Protected Conversation aufzubauen, während sich die aktuelle Transaktion bereits in der Terminierungsphase befindet. Zu diesem Zeitpunkt können keine weiteren Conversations bzw. Dialoge mehr an die Transaktion angeschlossen werden.
&APPID PID des lokalen Anwendungsprogramms
&CONVID Conversation ID

```
OpenCPIC [TPM] 314 Allocate() von lokalem Anwendungsprogramm (PID &APPID) abgelehnt
          Partneranwendung: &PARTNAME
          Grund: Fehlerhafte Security-Parameter
```

Bedeutung
Ein lokales AP hat versucht, mit *Allocate (CMALLC)* eine Conversation mit *conversation_security_type* = *CM_SECURITY_PROGRAM* aufzubauen, wobei entweder *security_user_ID* oder *security_password* oder beide nicht gesetzt waren. Der *CMALLC* kehrt deshalb mit *CM_PARAMETER_ERROR* zurück.
&APPID PID des lokalen Anwendungsprogramms
&PARTNAME Name der gewünschten Partneranwendung

Maßnahme
Geben Sie eine Benutzerkennung und ein Passwort an oder ändern Sie den Wert von *conversation_security_type*. Benutzerkennung und Passwort definieren Sie entweder in der Generierungsanweisung SYMDEST (Parameter SEC-UID und SEC-PASS) oder mit den CPI-C-Aufrufen S*et_Conversation_Security_Password (CMSCSP)* und S*et_Conversation_Security_User_ID (CMSCSU)*. Den *conversation_security_type* können Sie ebenfalls entweder in der Generierungsdatei setzen (SYMDEST-Anweisung, Parameter SEC-TYPE) oder mit dem CPI-C-Aufruf S*et_Converstaion_Security_Type (CMSCST)*.

```
OpenCPIC [TPM] 315 Allocate() von lokalem Anwendungsprogramm (PID &APPID) abgelehnt
           Partneranwendung: &PARTNAME
           Grund: CCR-Synatx nicht im Application-Context vereinbart
```

Bedeutung
Ein lokales Anwendungsprogramm hat versucht, mit *Allocate (CMALLC)* eine Protected-Conversation zu einer Partneranwendung aufzubauen, in deren Application Context die CCR-Syntax nicht enthalten ist.
&APPID PID des lokalen Anwendungsprogramms
&PARTNAME Name der gewünschten Partneranwendung

Maßnahme
Wenn Sie zu dieser Partneranwendung Protected-Conversations betreiben wollen, müssen Sie die aktuelle Generierung ändern. Sie können entweder einen anderen Application-Context vereinbaren, der die CCR-Syntax enthält (Parameter APPL-CONTEXT in der PARTAPPL-Anweisung), oder Sie müssen explizit angeben, dass der von Ihnen generierte Application-Context die CCR-Syntax enthalten soll (Parameter CCR=YES). Wenn Sie Ihre Generierung ändern, müssen Sie mit *ocpic_gen* eine neue Konfigurationsdatei erzeugen sowie den OpenCPIC-Manager beenden und anschließend neu starten.

```
OpenCPIC [TPM] 320 Passiver Dialogwunsch (CMACCI/CMACCP) abgelehnt (AP PID &APPID)
           Grund: Unzulaessiger Multiple-Accept
```

Bedeutung
Ein lokales AP, welches bereits eine Conversation mit dem *sync_level CM_SYNC_POINT* oder *CM_SYNC_POINT_NO_CONFIRM* zu einem Superior-AP unterhält, hat versucht, mit *Accept_Conversation (CMACCP)* oder *Accept_Incoming (CMACCI)* eine weitere Conversation zu akzeptieren. Dies ist in OpenCPIC nicht erlaubt, siehe dazu auch Anmerkung 1 auf Seite 83. Der Aufruf kehrt mit dem Returnwert *CM_PRODUCT_SPECIFIC_ERROR* zurück. Im Falle von *CMACCI* ändert sich der Wert des *conversation_state* dadurch nicht.
&APPID PID des betreffenden Anwendungsprogramms

```
OpenCPIC [TPM] 321 Passiver Dialogwunsch (CMACCI/CMACCP) abgelehnt (AP PID &APPID)
           Grund: Unzulaessiger Accept im Root-Knoten
```

Bedeutung
Diese Meldung wird ausgegeben, wenn ein lokales Anwednungsprogramm versucht, mit *Accept_Conversation (CMACCP)* oder *Accept_Incoming (CMACCI)* eine Conversation zu akzeptieren und mindestens eine der folgenden Bedingungen wahr ist:
- Das AP hat bereits Dialoge zu Subordinate-Partnern mit *sync_level CM_SYNC_POINT* oder *CM_SYNC_POINT_NO_CONFIRM*
- Das AP bereits hat *tx_begin()* aufgerufen
Damit ist das AP der Root-Knoten der aktuellen bzw. nächsten Transaktion. Ein AP kann zu einem Zeitpunkt nur an einer Transaktion beteiligt sein und Transaktionsbäume können nur nach unten (d. h. durch Einbeziehen weiterer Subordinates) erweitert werden. Siehe dazu auch Anmerkung 1 auf Seite 83. Der *CMACCP* bzw. *CMACCI*-Aufruf wird mit dem

Returnwert *CM_PRODUCT_SPECIFIC_ERROR* abgelehnt. Im Falle von *CMACCI* ändert sich der Wert des *conversation_state* dadurch nicht.

&APPID PID des betreffenden Anwendungsprogramms

```
OpenCPIC [TPM] 350 XAP-TP-Provider lehnt Associationswunsch ab
          (XAP-TP Dialog &INSTNO, AP PID &APPID, Conversation ID &CONVID)
          result = &RESULT, source = &SOURCE, reason = &REASON
```

Bedeutung

Der OpenCPIC-Manager hat aufgrund eines *Allocate (CMALLC)* eines lokalen AP versucht, eine Association zu belegen . Der lokale XAP-TP-Baustein hat den Association-Allocate-Request abgelehnt und der *CMALLC* kehrt mit dem Returnwert *CM_ALLOCATE_FAILURE(_RETRY)* oder *CM_UNSUCCESSFUL* zurück.

&INSTNO Nummer der XAP-TP-Instanz
&APPID PID des betroffenen Anwendungsprogramms
&CONVID Conversation ID
&RESULT Ergebnis des Association-Allocate-Request (siehe unten, Tabelle A)
&SOURCE Quelle der Ablehnung (siehe unten, Tabelle B)
&REASON Grund der Ablehnung (siehe unten, Tabelle C)

Tabelle A: Bedeutung von *result = &RESULT*

result	XAP Code	Bedeutung
1	AP_REJ_PERM	Association-Allocation-Request dauerhaft abgelehnt
2	AP_REJ_TRAN	Association-Allocation-Request vorübergehend abgelehnt

Tabelle B: Bedeutung von *source = &SOURCE*)

source	XAP Code	Bedeutung
0	AP_ACSE_SERV_USER	ACSE Service User
1	AP_ACSE_SERV_PROV	ACSE Service Provider
2	AP_PRES_SERV_PROV	Presentation Service Provider
7	AP_APM_SERV_PROV	APM Service Provider(Association Pool Manager)

Tabelle C: Bedeutung von *reason = &REASON*)

reason	XAP/XAP-TP Code	Bedeutung
-1	AP_TP_NRSN	kein Grund angegeben
0	AP_TP_CCR_V2_NAVAIL	CCR-Version 2 ist nicht verfügbar
1	AP_TP_VER_NAVAIL	Versions-Inkompatibilität
2	AP_TP_CW_REJ	Contention Winner Eigenschaft abgelehnt
3	AP_TP_BM_REJ	Bid mandatory Wert abgelehnt

reason	XAP/XAP-TP Code	Bedeutung
6	AP_TP_BAD_AET	lokaler oder entfernter Application Entity Title (AET) ungültig
7	AP_TP_BAD_POOL	kein Association-Pool gefunden für den angegebenen AET
58	AP_TP_ASSOC_NVAIL	Alle Associations aus dem Association-Pool sind belegt, jedoch gestatten die Pool-Grenzen den Aufbau weiterer Associations. Der XAP-TP-Baustein hat mit dem Aufbau weitere Associations begonnen (nur möglich für *return_control = CM_IMMEDIATE*).
59	AP_TP_POOL_LIMIT	Alle Associations aus dem Association-Pool sind belegt und die Pool-Grenzen gestatten keinen Aufbau weiterer Associations (nur möglich für *return_control = CM_IMMEDIATE*).
60	AP_TP_POOL_TIMEOUT	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	AP_TP_DIALOGUE_REFUSED	Die Association soll für eine Protected Conversation belegt werden und die aktuelle Transaktion befindet sich in der Endphase. Zu diesem Zeitpunkt können keine weiteren Dialoge mehr an den Transaktionsbaum angegliedert werden.

```
OpenCPIC [TPM] 351 XAP-TP-Provider meldet Verlust der Association
           (XAP-TP Dialog &INSTNO, AP PID &APPID, Conversation ID &CONVID)
           source = &SOURCE, reason = &REASON, event = &EVENT
```

Bedeutung

Der lokale XAP-TP-Baustein hat den Verlust einer Association, die bereits durch einen Dialog belegt war, gemeldet.

&INSTNO Nummer der XAP-TP-Instanz

&APPID PID des betroffenen Anwendungsprogramms

&CONVID Conversation ID

&SOURCE Quelle des Associationsverlusts (siehe unten, Tabelle D)
Ist *&SOURCE* gleich 0 oder 1, wurde eines der XAP-Protokollelemente *A-ABORT-IND* oder *A-RELEASE-IND* empfangen. Ist *&SOURCE* gleich 2, wurde das XAP-Protokollelement *A-PABORT-IND* empfangen.

&REASON Grund des Associationsverlusts. Der Wert von *&REASON* ist abhängig von *&SOURCE*. Ist *&SOURCE* gleich 0 oder 1, wurde eines der XAP-Protokollelemente *A-ABORT-IND* oder *A-RELEASE-IND* empfangen

Ist *&SOURCE* gleich 2, wurde das XAP-Protokollelement *A-PABORT-IND* empfangen. (Siehe unten, Tabellen E.1 bis E.3).

&EVENT In dieser Version immer gleich -1

Tabelle D: Bedeutung von *source = &SOURCE*)

source	XAP Code	Bedeutung
0	AP_ACSE_SERV_USER	ACSE Service User
1	AP_ACSE_SERV_PROV	ACSE Service Provider
2	AP_PRES_SERV_PROV	Presentation Service Provider
7	AP_APM_SERV_PROV	APM Service Provider (Association Pool Manager)

Tabelle E.1: Bedeutung von *reason = &REASON* für *&SOURCE* gleich 0 oder 1)

reason	XAP-TP Code	Bedeutung
-1	AP_RSN_NOVAL	*A-RELEASE-IND*: kein Grund angegeben
0	AP_REL_NORMAL	*A-RELEASE-IND*: normaler Release-Request
1	AP_REL_URGENT	*A-RELEASE-IND*: dringender Release-Request
12	AP_TP_ABORTED	*A-ABORT-IND*
30	AP_REL_USER_DEF	*A-RELEASE-IND*: benutzerdefinierter Release-Request

Tabelle E.2: Bedeutung von *reason = &REASON* für *&SOURCE* gleich 2)

reason	XAP Code	Bedeutung
-1	AP_RSN_NOVAL	kein Abbruchgrund angegeben
0	AP_NSPEC	Abbruchgrund nicht bekannt
1	AP_UNREC_PPDU	unbekannte Presentation Protocol Data Unit empfangen
2	AP_UNEXPT_PPDU	unerwartete Presentation Protocol Data Unit empfangen
3	AP_UNEXPT_SSPRIM	unerwartete Session Service Primitive empfangen
4	AP_UNREC_PPDU_PARM	Presentation Protocol Data Unit mit unbekanntem Parameter empfangen
5	AP_UNEXPT_PPDU_PARM	Presentation Protocol Data Unit mit unerwartetem Parameter empfangen
6	AP_INVAL_PPDU_PARM	Presentation Protocol Data Unit mit ungültigem Parameter empfangen

Tabelle E.3: Bedeutung von *reason = &REASON* für *&SOURCE* gleich 7)

reason	XAP Code	Bedeutung
5	AP_TP_IN_DIALOGUE	Ein Dialog von einem Partner hat die Association belegt.

```
OpenCPIC [TPM] 352 Dialogwunsch durch den Partner (XAP-TP-User) abgelehnt
           (XAP-TP Dialog &INSTNO, AP PID &APPID, Conversation ID &CONVID)
```

Bedeutung

Der OpenCPIC-Manager hat vom Kommunikationspartner eine *TP-BEGIN-DIALOGUE-confirmation* mit *Result = rejected* empfangen. Die Ablehnung des Dialoges erfolgte durch den entfernten XAP-TP-User.

&INSTNO Nummer der XAP-TP-Instanz
&APPID PID des betroffenen Anwendungsprogramms
&CONVID Conversation ID

```
OpenCPIC [TPM] 353 Dialogwunsch durch den Partner (XAP-TP-Provider) abgelehnt
           (XAP-TP Dialog &INSTNO, AP PID &APPID, Conversation ID &CONVID)
           reason = &REASON
```

Bedeutung

Der OpenCPIC-Manager hat vom Kommunikationspartner eine *TP-BEGIN-DIALOGUE-confirmation* mit *Result = rejected* empfangen. Die Ablehnung des Dialoges erfolgte durch den entfernten XAP-TP-Baustein.

&INSTNO Nummer der XAP-TP-Instanz
&APPID PID des betroffenen Anwendungsprogramms
&CONVID Conversation ID
&REASON Grund der Ablehnung (siehe unten, Tabelle F)

Tabelle F: Bedeutung von *reason = &REASON*

reason	XAP-TP Code	Bedeutung
-1	AP_TP_NRSN	kein Grund angegeben
1	AP_TP_TPSUT_UNKNOWN	Der angegebeneTPSU Title ist beim Partner nicht bekannt.
2	AP_TP_TPSUT_NVAIL_PERM	Der angegebene TPSU Title ist beim Partner dauerhaft nicht verfügbar.
3	AP_TP_TPSUT_NVAIL_TRAN	Der angegebene TPSU Title ist beim Partner vorübergehend nicht verfügbar.
4	AP_TP_TPSUT_NEEDED	Die Partneranwendung hat eine *TP-BEGIN-DIALOGUE-indication* ohne Angabe eines TPSU Title empfangen.
5	AP_TP_FU_NSUP	Eine oder mehrere der gewünschten Funktionseinheiten (Functional Units) werden vom Partner nicht unterstützt.
6	AP_TP_FU_COMB_NSUP	Die gewünschte Kombination von Funktionseinheiten (Functional Units) wird vom Partner nicht unterstützt.
7	AP_TP_ASSOC_RES	Die Association, die für den Dialog belegt wurde, wird bereits von der Partneranwendung benutzt.

reason	XAP-TP Code	Bedeutung
8	AP_TP_RECIPIENT_UNKNOWN	Die Application Entity Parameter, die im *TP-BEGIN-DIALOGUE-request* angegeben wurden, identifizieren keine bekannte Application Entity Invocastion (AEI).

`OpenCPIC [TPM] 360 Ankommender Dialogwunsch abgelehnt: Shared Control gewünscht`

Bedeutung
Der OpenCPIC-Manager hat vom Partner eine *TP-BEGIN-DIALOGUE-indication* empfangen, mit der die Funktionseinheit (Functional Unit) *Shared Control* gewünscht wurde. Da OpenCPIC diese Funktionseinheit nicht unterstützt, wird der Dialog durch den OpenCPIC-Manager abgelehnt.

`OpenCPIC [TPM] 361 Ankommender Dialogwunsch abgelehnt: kein lokaler TPSU Title angegeben`

Bedeutung
Der OpenCPIC-Manager hat vom Partner eine *TP-BEGIN-DIALOGUE-indication* ohne Angabe eines TPSU Title empfangen. Der Dialogwunsch wird deshalb durch den lokalen OpenCPIC-Manager abgelehnt.

`OpenCPIC [TPM] 362 Ankommender Dialogwunsch abgelehnt: Typ des lokalen TPSU Title ungueltig`

Bedeutung
Der OpenCPIC-Manager hat vom Partner eine *TP-BEGIN-DIALOGUE-indication* mit einem ungültigen TPSU Title Typ empfangen. Der Dialogwunsch wird deshalb durch den lokalen OpenCPIC-Manager abgelehnt.

Maßnahme
Der OpenCPIC-Manager akzeptiert für TSPU Title nur die Typen *PRINTABLE_STRING* und *T61_STRING*. Ändern Sie den Typ des TPSU Title in der Konfigurierung oder in den Anwendungsprogrammen auf der Partnerseite. Falls die Partneranwendung ein OpenCPIC-Manager ist, korrigieren Sie den Parameter PARTNER-TYPE in der SYMDEST-Anweisung.

`OpenCPIC [TPM] 363 Ankommender Dialogwunsch abgelehnt: lokales AP &APNAME nicht konfiguriert`

Bedeutung
Der OpenCPIC-Manager hat vom Partner eine *TP-BEGIN-DIALOGUE-indication* für den TPSU Title *&APNAME* empfangen, der nicht als lokaler Name bekannt ist. Der Dialogwunsch wird deshalb durch den lokalen OpenCPIC-Manager abgelehnt.

Maßnahme
Wenn Ihre lokale Anwendung Dialoge für den Namen *&APNAME* annehmen soll, müssen Sie diesen dem OpenCPIC-Manager entweder per Generierung (LOCAPRO-Anweisung) oder durch den CPI-C Aufruf *Specify_Local_TP_Name (CMSLTP)* bekannt machen.

```
OpenCPIC [TPM] 364 Ankommender Dialogwunsch abgelehnt: lokales AP &APNAME kann nicht
           gestartet werden
```

Bedeutung
Der OpenCPIC-Manager hat vom Partner eine *TP-BEGIN-DIALOGUE-indication* für den TPSU Title *&APNAME* empfangen, der dem OpenCPIC-Manager per Generierung (LOCAPRO-Anweisung) bekannt gemacht wurde. Der OpenCPIC-Manager hat daraufhin versucht, das lokale Anwendungsprogramm zu starten, was jedoch fehlgeschlagen ist. Der Dialogwunsch wird deshalb durch den lokalen OpenCPIC-Manager abgelehnt.

Maßnahme
Überprüfen Sie den Pfadnamen in der LOCAPRO-Anweisung Ihrer Generierung sowie die Ausführungsrechte der betreffenden Datei.

```
OpenCPIC [TPM] 365 Ankommender Dialogwunsch abgelehnt: Typ des entfernten Application
           Process Title ungueltig
```

Bedeutung
Der OpenCPIC-Manager hat vom Partner eine *TP-BEGIN-DIALOGUE-indication* empfangen und konnte den Application Process Title (APT) des Absenders nicht dekodieren. Der Dialogwunsch wird deshalb durch den lokalen OpenCPIC-Manager abgelehnt. Dieser Fehler sollte nur bei der Kommunikation mit Fremdsystemen auftreten.

Maßnahme
Ändern Sie wenn möglich das Verhalten der Partneranwendung. Eventuell ist eine Kommunikation mit dieser Partneranwendung nicht möglich.

```
OpenCPIC [TPM] 366 Ankommender Dialogwunsch abgelehnt: Typ des entfernten Application
           Entity Qualifier ungueltig
```

Bedeutung
Der OpenCPIC-Manager hat vom Partner eine *TP-BEGIN-DIALOGUE-indication* empfangen und konnte den Application Entity Qualifier (AEQ) des Absenders nicht dekodieren. Der Dialogwunsch wird deshalb durch den lokalen OpenCPIC-Manager abgelehnt. Dieser Fehler sollte nur bei der Kommunikation mit Fremdsystemen auftreten.

Maßnahme
Ändern Sie wenn möglich das Verhalten der Partneranwendung. Eventuell ist eine Kommunikation mit dieser Partneranwendung nicht möglich.

```
OpenCPIC [TPM] 367 Ankommender Dialogwunsch abgelehnt: Partneranwendung nicht bekannt
```

Bedeutung
Der OpenCPIC-Manager hat eine *TP-BEGIN-DIALOGUE-indication* von einer Partneranwendung empfangen, die in der lokalen Anwendung nicht bekannt ist. Der Dialogwunsch wird deshalb durch den lokalen OpenCPIC-Manager abgelehnt.

Maßnahme
Definieren Sie die Partneranwendung mit der PARTAPPL-Anweisung in Ihrer Generierungsdatei, erzeugen Sie mit *ocpic_gen* eine neue Konfigurationsdatei, beenden Sie den OpenCPIC-Manager und starten Sie ihn anschließend neu.

`OpenCPIC [TPM] 368 Ankommender Dialogwunsch abgelehnt: lokaler TPSU Title kann nicht dekodiert werden`

Bedeutung
Der OpenCPIC-Manager hat vom Partner eine *TP-BEGIN-DIALOGUE-indication* empfangen und konnte den lokalen TPSU Title nicht dekodieren. Der Dialogwunsch wird deshalb durch den lokalen OpenCPIC-Manager abgelehnt. Dieser Fehler sollte nur bei der Kommunikation mit Fremdsystemen auftreten.

Maßnahme
Ändern Sie wenn möglich das Verhalten der Partneranwendung. Eventuell ist eine Kommunikation mit dieser Partneranwendung nicht möglich.

`OpenCPIC [TPM] 369 Ankommender Dialogwunsch abgelehnt: CCR-Syntax für die Partneranwendung nicht vereinbart`

Bedeutung
Der OpenCPIC-Manager hat vom Partner eine *TP-BEGIN-DIALOGUE-indication* empfangen, mit der die Funktionseinheit (Functional Unit) *Commit* gewünscht wurde. In der lokalen Generierung wurde jedoch für diese Partneranwendung die CCR-Syntax nicht vereinbart (Parameter APPL-CONTEXT bzw. CCR in der PARTAPPL-Anweisung). Der Dialogwunsch wird deshalb durch den lokalen OpenCPIC-Manager abgelehnt.

Maßnahme
Wenn Sie mit dieser Partneranwendung Dialoge unterhalten wollen, die die Funktionseinheit *Commit* unterstützen, müssen Sie dies dem OpenCPIC-Manager per Generierung bekannt machen. Ändern Sie die Generierungsdatei, erzeugen Sie mit *ocpic_gen* eine neue Konfigurationsdatei, beenden Sie den OpenCPIC-Manager und starten Sie ihn anschließend neu.

`OpenCPIC [TPM] 370 XAP-TP Dialog &INSTNO durch den Partner (XAP-TP-Provider) abgebrochen`

Bedeutung
Der OpenCPIC-Manager hat für einen Dialog eine *TP-P-ABORT-indication* erhalten, d. h. der Dialog wurde durch den entfernten XAP-TP-Baustein abgebrochen
&INSTNO Nummer der XAP-TP-Instanz

`OpenCPIC [TPM] 371 XAP-TP Dialog &INSTNO durch den Partner (XAP-TP-User) abgebrochen`

Bedeutung
Der OpenCPIC-Manager hat für einen Dialog eine *TP-U-ABORT-indication* erhalten, d. h. der Dialog wurde durch den entfernten XAP-TP-User abgebrochen
&INSTNO Nummer der XAP-TP-Instanz

```
OpenCPIC [TPM] 372 Empfangene Log-Daten:
```

Bedeutung
Diese Meldung wird zusätzlich zur Meldung 370 bzw. 371 ausgegeben, falls das empfangene Protokollelement Log-Daten enthielt. OpenCPIC unterstützt keine Log-Daten, d. h. beim Abbrechen eines Dialogs von der lokalen Seite aus können dem Partner keine Log-Daten mitgegeben werden. Bei der Kommunikation mit Fremdsystemen ist es jedoch möglich, dass Log-Daten bei einem Dialogabbruch von der Partnerseite aus mitgesendet werden. Da der CPI-C Aufruf *Extract_Log_Data (CMELD)* nicht unterstützt wird, können die Log-Daten in diesem Fall nur der Datei *prot.mgr* entnommen werden.

9.1.3 Meldungen der Komponente TM

```
OpenCPIC [TM] 401 TX-Request von unbekanntem Absender (PID &APPID) empfangen (&MODNAME )
```

Bedeutung
Der OpenCPIC-Manager hat einen TX-Request von einem unbekannten Anwendungsprogramm erhalten. Der Request wird verworfen.
&APPID PID des Absender-Programms
&MODNAME Modul-Name

Maßnahme
Prüfen Sie, ob Ihr Anwendungsprogramm mit der aktuellen Version der Bibliothek *libocpic.a* gebunden wurde. Starten Sie das Anwendungsprogramm erneut. Bei wiederholtem Auftreten schalten Sie den CPI-C- und den Manager-Trace ein. Sichern Sie den Inhalt des Verzeichnisses *$OCPICDIR/PROT* und verständigen Sie Ihren Service.

```
OpenCPIC [TM] 402 Unzulaessige Zustaende fuer die Verarbeitung von tx_commit():
          Die Kombination des Zustands der CPI-C-Conversation (&CONVSTATE) mit dem Zustand
          des zugehoerigen XAP-TP-Dialogs (&DIASTATE) ist unzulaessig fuer Commit
          oder es liegt eine nicht abgeschlossene Operation vor (OI = &OPINC).
```

Bedeutung
Der OpenCPIC-Manager hat von einem lokalen Anwendungsprogramm einen *tx_commit()* Request in einem unzulässigen Zustand erhalten. Dies deutet auf einen internen Fehler in der TX-Bibliothek oder im OpenCPIC-Manager hin.
&CONVSTATE CPI-C-Zustand der Conversation
&DIASTATE XAP-TP-Zustand des Dialogs
&OPINC nicht abgeschlossene Operation liegt vor (1=TRUE, 0=FALSE)

Maßnahme
Sichern Sie den Inhalt des Verzeichnisses *$OCPICDIR/PROT* und verständigen Sie Ihren Service.

```
OpenCPIC [TM] 403 Unzulaessige Zustaende fuer die Verarbeitung von tx_rollback():
          Der XAP-TP-Zustand (&DIASTATE) des Dialoges auf der XAP-TP-Instanz &INSTNO
          ist unzulaessig fuer einen TP_ROLLBACK_REQ
          oder es liegt eine nicht abgeschlossene Operation vor (OI = &OPINC).
```

Bedeutung
Der OpenCPIC-Manager hat von einem lokalen Anwendungsprogramm einen *tx_rollback()*-Request in einem unzulässigen Zustand erhalten. Dies deutet auf einen internen Fehler in der TX-Bibliothek oder im OpenCPIC-Manager hin.

&DIASTATE	XAP-TP-Zustand des Dialogs
&INSTNO	Nummer der XAP-TP-Instanz
&OPINC	nicht abgeschlossene Operation (1 = TRUE, 0 = FALSE)

Maßnahme
Sichern Sie den Inhalt des Verzeichnisses *$OCPICDIR/PROT* und verständigen Sie Ihren Service.

```
OpenCPIC [TM] 404 Der TP Kontrollblock fuer die Primitive &PRIMTYPE kann nicht lokalisiert
          werden.
```

Bedeutung
Der OpenCPIC-Manager hat vom XAP-TP-Baustein ein Protokollelement erhalten, dessen Empfänger dem OpenCPIC-Manager nicht bekannt ist. Das Protokollelement wird verworfen. Dieser Fehler wird lediglich protokolliert, anschließend wird der normale Programmlauf fortgesetzt.

&PRIMTYPE	Typ des Protokollelements

Maßnahme
Bei gehäuftem Auftreten schalten Sie den XAP-TP- und den Manager-Trace ein, sichern Sie die Protokolldateien und verständigen Sie Ihren Service.

```
OpenCPIC [TM] 405 Unpassender Parameter ReqType in tm_send_reply():
          Erwarteter ReqType = &REQEXP (tp), uebergebener ReqType = &REQRECV.
```

Bedeutung
Der OpenCPIC-Manager hat versucht, einen TX-Request eines lokalen Anwendungsprogrammes mit einem falschen Request-Typ zu beantworten. Dies deutet auf einen internen Fehler im OpenCPIC-Manager hin. *&REQEXP* und *&REQRECV* geben den erwarteten und den tatsächlichen Request-Typ in Hexadezimaldarstellung an.

Maßnahme
Sichern Sie den Inhalt des Verzeichnisses *$OCPICDIR/PROT* und verständigen Sie Ihren Service.

```
OpenCPIC [TM] 406 Die folgende Log-Record-Datei enthaelt einen Log-Damage Record und wird
          nicht geloescht: &FILENAME
```

Bedeutung
Diese Meldung kann sowohl bei einem Warmstart des OpenCPIC-Managers (Wiederanlauf) als auch während des normalen Betriebs auftreten.
Beim Warmstart versucht der OpenCPIC-Manager, anhand der im Verzeichnis *$OCPICDIR/SYNC* gesicherten Log-Records, die unterbrochenen Transaktionen zu beenden und löscht anschließend die Log-Record-Dateien, mit Ausnahme der Dateien, die einen Log-Damage-Record enthalten.
Im Normalbetrieb löscht OpenCPIC-Manager die Log-Record-Dateien bei Beendigung einer Transaktion, d. h. bei Empfang der *TP-COMMIT-COMPLETE-indication* bzw. *TP-ROLLBACK-COMPLETE-indication* für alle beteiligten Instanzen. Auch in diesem Fall werden jedoch im Laufe der Transaktion empfangene Log-Damage-Records nicht beseitigt.
Ein Log-Damage-Record weist auf mögliche Inkonsistenzen zwischen den an einer verteilten Transaktion beteiligten Resource Managern hin. Da er ggf. wichtige Informationen zum Ablauf und letztem Status der unterbrochenen Transaktion enthält, kann er nur vom Systemadministrator gelöscht werden (siehe auch “Abschnitt „Heuristische Entscheidungen“ auf Seite 146“).

Maßnahme
Prüfen Sie, ob Sie die Informationen in den gesicherten Log-Record-Dateien benötigen. Log-Record-Dateien sind Binärdateien und daher nicht ohne Weiteres lesbar. Ihr Aufbau ist in der XAP-TP-Spezifikation [25] beschrieben. Der OpenCPIC-Manager gibt bei Empfang eines Log-Damage-Records alle benutzerrelevanten Daten aus dem Record in eine ASCII-Log-Datei aus (siehe Meldung 425). Mit Hilfe dieser Informationen sowie evtl. vorhandenen Logging-Informationen Ihres Resource Managers können Sie als Systemverwalter den konsistenten Zustand Ihrer Daten wiederherstellen.

```
OpenCPIC [TM] 407 Kein Zugriff auf das Verzeichnis &DIRNAME waehrend des Restarts, errno =
          &ERRNO
```

Bedeutung
Der Versuch des OpenCPIC-Managers, das Verzeichnis *&DIRNAME ($OCPICDIR/sync)* mit Hilfe des Systemaufrufs *opendir()* zu öffnen, ist fehlgeschlagen. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben. Nach Ausgabe dieser Meldung beendet sich der OpenCPIC-Manager.

Maßnahme
Abhängig von *&ERRNO*. Falls das angegebene Verzeichnis bereits existiert, prüfen bzw. ändern Sie ggf. dessen Zugriffsrechte. Prüfen bzw. ändern Sie ggf. die Zugriffsrechte der übergeordneten Verzeichnisse. Starten Sie dann den OpenCPIC-Manager erneut.

```
OpenCPIC [TM] 408 Der Log-Record fuer das AP mit der PID &APPID kann nicht aktualisiert
           werden.
           Die bearbeitete Primitive ist &PRIMTYPE.
```

Bedeutung
Der OpenCPIC-Manager hat versucht aufgrund eines empfangenen Protokollelements, einen vorhandenen Log-Record zu aktualisieren. Dies ist z. B. notwendig, wenn dem Transaction Manager in der Commit-Phase eine heuristische Entscheidung gemeldet wird und der aktuelle Log-Ready-Record in einen Log-Damage-Record geändert werden muss. Die Meldung besagt, dass der Log-Record nicht geändert werden konnte, möglicherweise aufgrund eines Dateifehlers oder Speicherplatzmangels. Daraufhin initiiert der Transaction Manager einen Rollback der Transaktion.
&APPID PID des betrefeffenden Anwedungsprogramms
&PRIMTYPE Typ des momentan bearbeiteten XAP-TP-Protokollelements

```
OpenCPIC [TM] 409 Unbekannte Primitive empfangen, sptype = &PRIMTYPE
```

Bedeutung
Der OpenCPIC-Manager hat vom XAP-TP-Baustein ein unbekanntes Protokollelement vom Type *&PRIMTYPE* erhalten. Das Protokollelement wird verworfen. Dieser Fehler wird lediglich protokolliert, anschließend wird der normale Programmlauf fortgesetzt.
&PRIMTYPE Typ des XAP-TP-Protokollelements

Maßnahme
Bei gehäuftem Auftreten schalten Sie den XAP-TP- und den Manager-Trace ein, sichern Sie die Protokolldateien und verständigen Sie Ihren Service.

```
OpenCPIC [TM] 410 TP_READY_ALL_IND ohne Log-Record fuer das AP mit der PID &APPID empfangen
```

Bedeutung
Der Transaction Manager hat das Protokollelement *TP-READY-ALL-indication* empfangen und es wurde kein Log-Record mitgegeben. Dies deutet evtl. auf einen internen Fehler des XAP-TP-Bausteins hin. Der Transaction Manager initiiert daraufhin einen Rollback der Transaktion.

Maßnahme
Sichern Sie den Inhalt des Verzeichnisses *$OCPICDIR/PROT* und verständigen Sie Ihren Service.

```
OpenCPIC [TM] 411 log_req_c enthielt &NUMBER unerwartete Elemente bei der Bearbeitung einer
           TP_READY_ALL_IND fuer das AP mit der PID &APPID
```

Bedeutung
Diese Meldung deutet auf einen internen Fehler im OpenCPIC-Manager hin. Dieser Fehler wird lediglich protokolliert, anschließend wird der normale Programmlauf fortgesetzt.

Maßnahme
Sichern Sie den Inhalt des Verzeichnisses *$OCPICDIR/PROT* und verständigen Sie Ihren Service.

```
OpenCPIC [TM] 412 TP_READY_ALL_IND waehrend XAP-TP-Warmstart empfangen
```

Bedeutung
Der OpenCPIC-Manager hat während des Warmstarts das Protokollelement *TP-READY-ALL-indication* für eine zuvor unterbrochene Transaktion empfangen. Diese Situation sollte normalerweise nicht auftreten. Das Protokollelement wird verworfen. Dieser Fehler wird lediglich protokolliert, anschließend wird der normale Programmlauf fortgesetzt.

```
OpenCPIC [TM] 413 Kein Zugriff auf die Log-Datei &FILENAME waehrend Restart; Systemaufruf
          &SYSCALL ,
          errno = &ERRNO
```

Bedeutung
Während des Warmstarts trat ein Fehler beim Öffnen oder Lesen der Log-Record-Datei *&FILENAME* auf *&SYSCALL* ist der Name des Systemaufrufs, bei welchem der Fehler auftrat (z. B. *open*, *read*). Die genauere Fehlerursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben. Die Log-Record-Datei wird übergangen, das heißt, für die betreffende Transaktion können keine Wiederanlaufmaßnahmen durchgeführt werden.

Maßnahme
Abhängig von *&ERRNO*. Prüfen Sie die Zugriffsrechte der angegebenen Datei und der übergeordneten Verzeichnisse. Führen Sie anschließend einen erneuten Warmstart durch.

```
OpenCPIC [TM] 414 Kein XAP-TP-Record in der Log-Datei &FILENAME
```

Bedeutung
Die Datei *&FILENAME* aus dem Verzeichnis *$OCPICDIR/sync* enthält keinen Log-Record von XAP-TP, sondern nur lokale Daten zur betreffenden Transaktion. Dies bedeutet, dass an der Transaktion nur lokale Resource Manager beteiligt waren, jedoch keine Partneranwendungen. Demzufolge führt der Transaction Manager nur lokale Wiederanlaufmaßnahmen durch, ohne Beteiligung des XAP-TP-Bausteins. Diese Meldung wird lediglich zur Information ausgegeben.

```
OpenCPIC [TM] 415 Recovery (Warmstart) wird durchgefuehrt fuer die Transaktion,
          repraesentiert durch die Log-Datei: &FILENAME
```

Bedeutung
Diese Meldung wird zu Beginn der Wiederanlaufmaßnahme für eine bestimmte Transaktion ausgegeben. Der Transaction Manager versucht, die durch einen Programm- oder Systemfehler unterbrochene Transaktion zu einem geordneten Ende zu bringen. Über das Ergebnis der Wiederanlaufmaßnahme werden Sie durch weitere Meldungen informiert.

```
OpenCPIC [TM] 417 Fehler bei fork() waehrend Recovery, errno = &ERRNO
```

Bedeutung
Der OpenCPIC-Manager hat versucht, während des Warmstarts einen Kindprozess für die separate Durchführung von Wiederanlaufmaßnahmen zu starten. Beim Start des Kindprozesses mit *fork()* hat es einen Systemfehler gegeben. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben

Maßnahme
Abhängig von *&ERRNO*. Verständigen Sie ggf. Ihren Systemverwalter.

```
OpenCPIC [TM] 418 Transaktionsende mit TX_UNCHAINED erfordert Dialogabbruch durch den
          OpenCPIC-Manager
          (XAP-TP Dialog &INSTNO, AP PID &APPID, Conversation ID %d)
```

Bedeutung
Diese Meldung erscheint bei Beendigung einer Transaktion, wenn *tx_transaction_control = TX_UNCHAINED* ist und für eine der beteiligten Conversations die CPI-C-Variable *transaction_control = CM_CHAINED_TRANSACTIONS* gesetzt ist. In diesem Fall erfolgt, wie in der X/Open CPI-C Spezifikation [24] beschrieben, ein automatischer Dialogabbau durch den OpenCPIC-Manager. Der nächste CPI-C-Aufruf auf der betreffenden Conversation wird mit *CM_RESOURCE_FAILURE_RETRY* beantwortet.

&INSTNO	Nummer der XAP-TP-Instanz
&APPID	PID des betroffenen Anwendungsprogramms
&CONVID	Conversation ID

Maßnahme
Sie können diese Situation vermeiden, indem Sie *tx_transaction_control* (TX) und *transaction_control* (CPI-C) aufeinander abstimmen. Wenn Sie mit Transaktionsbeendigung sofort eine neue Transaktion beginnen wollen, setzen Sie beide Werte auf *TX_CHAINED* (TX) bzw. *CM_CHAINED_TRANSACTIONS* (CPI-C). Wünschen Sie dies nicht, setzen Sie beide Werte auf *TX_UNCHAINED* (TX) bzw. *CM_UNCHAINED_TRANSACTIONS* (CPI-C).

```
OpenCPIC [TM] 419 tx_begin von AP mit PID &APPID voruebergehend abgelehnt
          Grund: Interner Rollback wegen Transaction Control Differenzen noch nicht
          beendet.
```

Bedeutung
Aufgrund des wie bei Meldung 418 beschriebenen notwendigen Dialogabbruchs führt der OpenCPIC-Manager intern einen Rollback durch. Bis zur Beendigung dieses Rollback kann keine neue Transaktion begonnen werden. Ein *tx_begin()*-Aufruf in dieser Situation wird daher mit dem Returnwert *TX_ERROR* beantwortet, welcher, wie in [26] beschrieben, einen vorübergehenden Fehler anzeigt.

&APPID	PID des betroffenen Anwendungsprogramms

Maßnahme
Rufen Sie den *tx_begin()* zu einem späteren Zeitpunkt nochmals auf. Im übrigen gelten die bei Meldung 418 gegebenen Hinweise.

```
OpenCPIC [TM] 425 Log-Damage-Record erhalten mit XAP-TP-Primitive &PRIMTYPE
```

Bedeutung
Der OpenCPIC-Manager hat einen Log-Damage-Record erhalten. Die Informationen aus dem Log-Damage-Record werden in lesbarer Form in eine spezielle ASCII-Log-Datei ausgegeben (siehe “Abschnitt „Heuristische Entscheidungen“ auf Seite 146“). Diese Infor-

mationen dienen dazu, nach heuristischen Entscheidungen, den konsistenten Zustand aller an der Transaktion beteiligten Resource Manager wiederherzustellen.
&PRIMTYPE Typ des XAP-TP-Protokollelements

9.1.4 Meldungen des XAP-TP-Bausteins

Die Meldungen des XAP-TP-Bausteins besitzen eine separate Zählung. Sie werden jeweils von der Meldung 100 der Komponente BD eingeleitet.

Die Meldungen mit Ausnahme von P100 und P101 sind im openUTM-Handbuch „Meldungen, Test und Diagnose auf Unix- und Windows-Systemen“ beschrieben. Dort finden Sie auch die Beschreibung der zugehörigen Fehlercodes und der in den Meldungen enthaltenen Inserts (variable Elemente, deren Name mit einem ‚&‘ beginnt).

P100 `Instance Allocation Timeout:`
`&ACPNT,`
`&OSLPAP,`
`(&REFINST)`

Bedeutung
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
&REFINST interne Kennzeichnung der XAP-TP-Instanz

P101 `CMX Error:`
`&ACPNT,`
`&OSLPAP`

Bedeutung
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

9.2 Meldungen des Generierungsprogramms ocpic_gen

In den folgenden Meldungen haben die Platzhalter *&LINENO* und *&INVTXT* immer folgende Bedeutung:

&LINENO Nummer der Zeile in der Generierungsdatei, in der ein Fehler auftrat.

&INVTXT Fehlerhafter Text in der Generierungsdatei

```
ocpic_gen 00 Message catalog ocpic3.cat not opened: errno = &ERRNO (&ERRNAME)
```

Bedeutung
Der NLS-Meldungskatalog *ocpic3.cat* konnte nicht geöffnet werden. Er wird abhängig vom Inhalt der Umgebungsvariablen *$LANG* im Verzeichnis */opt/lib/nls/msg/De* bzw. */opt/lib/nls/msg/En* gesucht. Die genauere Fehlerursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Stellen Sie ggf. den Meldungskatalog wieder her, evtl. durch Neuinstallation des Produktes, oder ändern Sie den Inhalt von *$LANG*.

```
ocpic_gen 02 Unbekannter Text in Zeile &LINENO - ab &INVTXT ueberlesen
ocpic_gen 03 Kein Komma am Anfang der Zeile erlaubt (Fehler in Zeile &LINENO)
ocpic_gen 04 Kein Komma am Ende einer Anweisung erlaubt (Fehler in Zeile &LINENO)
ocpic_gen 05 Komma erwartet in Zeile &LINENO oder &LINENO
ocpic_gen 06 Komma an der falschen Stelle in Zeile &LINENO
```

Bedeutung
Diese Meldungen weisen auf einen syntaktischen Fehler in der Generierungsdatei hin. Die korrekte Syntax der Generierungsdatei ist im Kapitel „OpenCPIC generieren“ auf Seite 29 beschrieben.

Maßnahme
Korrigieren Sie die Generierungsdatei und starten Sie *ocpic_gen* erneut.

```
ocpic_gen 07 Pfadname muss mit / beginnen (Fehler in Zeile &LINENO - ab &INVTXT)
```

Bedeutung
Sie haben in der LOCAPRO-Anweisung mit dem Parameter PATH einen fehlerhaften Pfadnamen *&INVTXT* angegeben. Der Pfadname muss vollqualifiziert sein, d. h., er muss mit eine Schrägstrich (/) beginnen.

```
ocpic_gen 08 Unzulaessiger symdest_name (Fehler in Zeile &LINENO - ab &INVTXT
```

Bedeutung
Der *symdest_name* in der SYMDEST-Anweisung enthält unzulässige Zeichen. Zulässig sind nur Zeichen aus dem Character-Set 01134, d. h. Großbuchstaben und Ziffern.

```
ocpic_gen 09 Zahl erwartet (Fehler in Zeile &LINENO ab &INVTXT)
```

Bedeutung
Sie haben als Wert für einen numerischen Parameter ein nicht numerisches Zeichen (*&INVTXT*) eingegeben.

```
ocpic_gen 10 Unzulaessiger locapro_name (Fehler in Zeile &LINENO – ab &INVTXT)
```

Bedeutung
Der *locapro_name* enthält unzulässige Zeichen. Zulässige Zeichen sind Buchstaben, Ziffern, runde Klammern, Punkt(.), Pluszeichen(+), Bindestrich(-), Schrägstrich(/), Fragezeichen (?), Doppelpunkt (:), Hochkomma (‘) und Gleichheitszeichen (=).

```
ocpic_gen 11 Unzulaessiger Name (Fehler in Zeile &LINENO – ab &INVTXT)
```

Bedeutung
Der Name enthält unzulässige Zeichen. Zulässige Zeichen sind alle druckbaren Zeichen außer Leerzeichen, Semikolon(;), Komma (,) und Stern (*).

```
ocpic_gen 12 PARTAPPL muss zusammen mit partappl_name in einer Zeile stehen (Fehler in
           Zeile &LINENO)
ocpic_gen 13 LOCAPPL muss zusammen mit locappl_name in einer Zeile stehen (Fehler in Zeile
           &LINENO)
ocpic_gen 14 LOCAPRO muss zusammen mit locapro_name in einer Zeile stehen (Fehler in Zeile
           &LINENO)
ocpic_gen 15 SYMDEST muss zusammen mit symdest_name in einer Zeile stehen (Fehler in Zeile
           &LINENO)
ocpic_gen 16 APPLICATION-CONTEXT muss zusammen mit application_context_name in einer Zeile
           stehen
           (Fehler in Zeile &LINENO)
```

Bedeutung
Das Schlüsselwort einer Anweisung darf nicht alleine in einer Zeile stehen, sondern es muss der nachfolgende Name auch noch in derselben Zeile stehen.

```
ocpic_gen 20 LOCAPPL-Anweisung darf nur einmal in der Generierung auftreten (Fehler in
           Zeile &LINENO)
```

Bedeutung
Die Generierungsdatei enthält mehr als eine LOCAPPL-Anweisung. Dies ist nicht zulässig.
&LINENO Nummer der Zeile in der Generierungsdatei, in der der Fehler auftrat

```
ocpic_gen 21 Unzulaessige Anzahl von Elementen im OBJECT-IDENTIFIER (Fehler in Zeile
           &LINENO)
```

Bedeutung
Ein Parameter vom Typ Objektbezeichner (siehe Seite 31) muss mindestens zwei und maximal zehn Elemente enthalten.

```
ocpic_gen 22 Syntaxfehler in Zeile &LINENO
```

Bedeutung
Diese Meldung weist auf einen syntaktischen Fehler in der Generierungsdatei hin. Sie erscheint, wenn z. B. Parameter fehlen bzw. zu einer anderen Anweisung gehören oder als Folge eines Kommafehlers. Nach diesem Fehler wird der Rest bis zum Ende der Anweisung überlesen.

```
ocpic_gen 24 Name &INVTXT zu lang (Fehler in Zeile &LINENO)
ocpic_gen 25 Name &INVTXT muss genau acht Zeichen lang sein (Fehler in Zeile &LINENO)
```

Bedeutung
Der Name *&INVTXT* hat die falsche Länge.

```
ocpic_gen 26 Wert &INVTXT ausserhalb des Wertebereiches (Fehler in Zeile &LINENO)
```

Bedeutung
Der angegebene Parameterwert liegt außerhalb des zulässigen Wertebereichs.

```
ocpic_gen 27 Wert fuer CONNECT zu gross (Fehler in der Anweisung vor Zeile &LINENO)
```

Bedeutung
Der Parameter CONNECT=*connect_number* in der PARTAPPL-Anweisung darf nicht größer sein als die maximale Anzahl von Associations, die Sie mit dem Parameter ASSOCIATIONS in derselben Anweisung generiert haben.

```
ocpic_gen 28 Wert fuer CONTWIN zu gross (Fehler in der Anweisung vor Zeile &LINENO)
```

Bedeutung
Beim Parameter CONTWIN=*(contwin_min contwin_max)* in der PARTAPPL-Anweisung darf *contwin_max* nicht größer sein als die maximale Anzahl von Associations, die Sie mit dem Parameter ASSOCIATIONS in derselben Anweisung generiert haben.

```
ocpic_gen 29 Name &INVTXT in keiner vorhergehenden PARTAPPL-Anweisung definiert (Fehler in
          Zeile &LINENO)
```

Bedeutung
Sie haben in einer *SYMDEST*-Anweisung einen *partappl_name* angegeben, der in keiner PARTAPPL-Anweisung generiert ist. Beachten Sie, dass die PARTAPPL-Anweisung vor der SYMDEST-Anweisung erscheinen muss.

```
ocpic_gen 30 Name &INVTXT bereits vergeben (Fehler in Zeile &LINENO)
```

Bedeutung
Sie haben den Namen *&INVTXT* bereits in einer vorhergehenden Anweisung generiert. Die Namen in einer Generierungsdatei müssen eindeutig sein.

```
ocpic_gen 31 Der aus APT und AEQ gebildete AE Title ist bereits vergeben (Fehler in Zeile
           &LINENO)
```

Bedeutung
Sie haben in einer PARTAPPL- oder LOCAPPL-Anweisung einen AE Title (Kombination aus APT und AEQ) generiert, der bereits in einer vorhergehenden PARTAPPL- oder LOCAPPL-Anweisung vergeben wurde. Die AE Title dienen zur Identifizierung der Anwendungen im Netz und müssen daher eindeutig sein.

```
ocpic_gen 32 Interner Fehler: &ERRTXT.
```

Bedeutung
Im Programm *ocpic_gen* ist ein interner Fehler aufgetreten und es konnte keine Konfigurationsdatei erstellt werden.

Maßnahme
Sichern Sie zu Diagnosezwecken die Generierungsdatei und verständigen Sie Ihren Service.

```
ocpic_gen 33 Pflichtparameter APT fehlt (Fehler in Zeile &LINENO)
ocpic_gen 34 Pflichtparameter AEQ fehlt (Fehler in Zeile &LINENO)
```

Bedeutung
In einer LOCAPPL- oder PARTAPPL-Anweisung fehlt der angegebene Pflichtparameter. Die Parameter APT (Application Process Title) und AEQ (Application Entity Qualifier) müssen immer bei der Generierung der lokalen bzw. einer Partneranwendung angegeben werden. Sie bilden zusammen den AET (Application Entity Title) und werden in OSI-TP zur Adressierung einer Anwendung benötigt.

```
ocpic_gen 35 Parameter PARTNER-APRO passt nicht zu PARTNER-TYPE (Fehler in Zeile &LINENO)
```

Bedeutung
Sie haben in einer SYMDEST-Anweisung mit PARTNER-APRO = *partnerap_name* einen Namen generiert, der unzulässige Zeichen enthält. Die Menge der zulässigen Zeichen richtet sich nach dem mit PARTNER-TYPE generierten Wert.

```
ocpic_gen 36 context_name &INVTXT in keiner vorhergehenden APPLICATION-CONTEXT-Anweisung
           definiert (Fehler in Zeile &LINENO)
```

Bedeutung
Sie haben in einer PARTAPPL-Anweisung einen *context_name* angegeben, der in keiner APPLICATION-CONTEXT-Anweisung generiert ist. Beachten Sie, dass die APPLICATION-CONTEXT-Anweisung vor der PARTAPPL-Anweisung erscheinen muss.

```
ocpic_gen 37 Parameter CONTWIN: contwin-max ist kleiner als contwin-min (Fehler in Zeile
           &LINENO)
```

Bedeutung
Sie haben in einer PARTAPPL-Anweisung unzulässige Werte für den Parameter CONTWIN angegeben. Der Wert *contwin_max* muss größer sein als der Wert *contwin_min* (siehe Seite 37).

```
ocpic_gen 38 Name &INVTXT ist reserviert (Fehler in Zeile &LINENO)
```

Bedeutung
Sie haben ein Schlüsselwort oder einen reserviertes Wort als Namen vergeben. Bitte wählen Sie einen anderen Namen.

```
ocpic_gen 39 Name &INVTXT ist unzulaessig (Fehler in Zeile &LINENO)
```

Bedeutung
Sie haben in einer LOCAPPL- oder PARTAPPL-Anweisung einen unzulässigen Namen angegeben. Entweder ist einer der Namensteile NP1-NP5 zu lang oder der Name *&INVTXT* enthält zu viele Namensteile (d. h. zu viele Trennzeichen '.').

```
ocpic_gen 40 Pflichtparameter PARTNER-APPL fehlt (Fehler in Anweisung vor Zeile &LINENO)
ocpic_gen 41 Pflichtparameter PARTNER-APRO fehlt (Fehler in Anweisung vor Zeile &LINENO)
```

Bedeutung
In einer SYMDEST-Anweisung fehlt der Parameter PARTNER-APPL bzw. PARTNER-APRO.

```
ocpic_gen 42 Parameter SEC-UID fehlt (SEC-TYPE = PROGRAM) (Fehler in Anweisung vor Zeile
           &LINENO)
ocpic_gen 43 Parameter SEC-PASS fehlt (SEC-TYPE = PROGRAM) (Fehler in Anweisung vor Zeile
           &LINENO)
```

Bedeutung
Sie haben in einer SYMDEST-Anweisung SEC-TYPE=PROGRAM generiert. In diesem Fall sind die Parameter SEC-TYPE und SEC-PASS Pflicht.

```
ocpic_gen 44 Warnung: Parameter SEC-UID bzw. SEC-PASS in Anweisung vor Zeile &LINENO ist
           ohne Wirkung
```

Bedeutung
Sie haben in einer SYMDEST-Anweisung SEC-TYPE=NONE generiert. In diesem Fall sind die Parameter SEC-TYPE und SEC-PASS ohne Wirkung. Die Konfiguartionsdatei wird trotzdem erstellt.

```
ocpic_gen 50 Aufruf:
           Generierung: ocpic_gen [gen_file] [-f conf_file]
           Reverse       : ocpic_gen -r [conf_file]
```

Bedeutung
Sie haben *ocpic_gen* mit falschen Parametern aufgerufen.

Maßnahme
Starten Sie *ocpic_gen* erneut unter Angabe der korrekten Parameter.

```
ocpic_gen 51 Datei &FILENAME kann nicht geöffnet werden |
          errno = &ERRNO &ERRNAME
```

Bedeutung
Der Versuch, die Datei *&FILENAME* mit dem Systemaufruf *open()* zu öffnen, schlug fehl. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Prüfen und ändern Sie ggf. die Zugriffsrechte.

```
ocpic_gen 52 Die Version von ocpic_gen stimmt nicht mit der Datei &FILENAME ueberein
          (&INVVER statt &VERSION),
          oder die angegebene Datei wurde nicht mit ocpic_gen erzeugt.
```

Bedeutung
Diese Meldung tritt nur beim Schalter *-r* auf. Die Eingabedatei *&FILENAME* enthält eine falsche (*&INVVER*) oder keine Versionsnummer. Sie können mit *ocpic_gen -r* nur Konfigurationsdateien lesen, die mit der aktuellen Version von *ocpic_gen* erzeugt wurden.

```
ocpic_gen 53 LOCAPPL-Anweisung fehlt
```

Bedeutung
Die Generierungsdatei enthält keine LOCAPPL-Anweisung. Eine Generierungsdatei muss genau eine LOCAPPL-Anweisung enthalten.

```
ocpic_gen 54 PARTAPPL-Anweisung fehlt
```

Bedeutung
Die Generierungsdatei enthält keine PARTAPPL-Anweisung. Eine Generierungsdatei muss mindestens eine PARTAPPL-Anweisung enthalten.

```
ocpic_gen 55 Die Generierungsdatei enthaelt keine Generierungsanweisungen
```

Bedeutung
Die Generierungsdatei ist leer oder enthält nur Kommentarzeilen.

```
ocpic_gen 56 STOP (Fehler)
```

Bedeutung
ocpic_gen hat aufgrund eines schwerwiegenden Fehlers abgebrochen. Es konnte keine Konfigurationsdatei erstellt werden. Über die Fehlerursache gibt die vorhergehende Meldung Auskunft.

Maßnahme
Korrigieren Sie den Fehler und starten Sie *ocpic_gen* erneut.

`ocpic_gen 57 Nicht genug Speicherplatz vorhanden`

Bedeutung
ocpic_gen hat mit der Systemfunktion *malloc()* Speicherplatz vom Betriebssystem angefordert, diesen jedoch nicht erhalten.

Maßnahme
Reduzieren Sie den Speicherverbrauch anderer Programme oder vergrößern Sie den Hauptspeicher Ihres Systems. Starten Sie *ocpic_gen* dann erneut.

`ocpic_gen 58 Fehler beim Schreiben in die Datei &FILENAME`

Bedeutung
Beim Schreiben mit *fwrite()* in die Datei *&FILENAME* ist ein Fehler aufgetreten.

Maßnahme
Prüfen Sie die Zugriffsrechte der angegebenen Datei.

`ocpic_gen 59 Fehler beim Lesen aus der Datei &FILENAME`

Bedeutung
Beim Lesen mit *fread()* aus der Datei *&FILENAME* ist ein Fehler aufgetreten.

Maßnahme
Prüfen Sie die Zugriffsrechte der angegebenen Datei.

`ocpic_gen 60 Generierung in die Datei &FILENAME`

Bedeutung
Diese Meldung wird nach erfolgreichem Generierungslauf ausgegeben. *ocpic_gen* erstellt die Konfigurationsdatei *&FILENAME*.

9.3 Meldungen des Administrationsprogramms ocpic_adm

```
ocpic_adm 00 Message catalog ocpic3.cat not opened
```

Bedeutung
Der NLS-Meldungskatalog *ocpic3.cat* konnte nicht geöffnet werden. Er wird abhängig vom Inhalt der Umgebungsvariablen *$LANG* im Verzeichnis */opt/lib/nls/msg/De* bzw. */opt/lib/nls/msg/En* gesucht.

Maßnahme
Stellen Sie den Meldungskatalog wieder her, evtl. durch Neuinstallation des Produktes, oder ändern Sie den Inhalt von *$LANG*.

```
ocpic_adm 01 Aufruf:
           ocpic_adm -t ton
           ocpic_adm -t mon
           ocpic_adm -t bon
           ocpic_adm -t tof
           ocpic_adm -t mof
           ocpic_adm -t bof
           ocpic_adm -t tfl
           ocpic_adm -t mfl
           ocpic_adm -t bfl
           ocpic_adm -d <partappl-name>
           ocpic_adm -a <partappl_name>
           ocpic_adm -e
           ocpic_adm -l <trace-file>
```

Bedeutung
Sie haben *ocpic_adm* mit falschen Parametern aufgerufen.

```
ocpic_adm 02 Nicht genug Speicherplatz vorhanden
```

Bedeutung
ocpic_adm hat mit der Systemfunktion *malloc()* Speicherplatz vom Betriebssystem angefordert, diesen jedoch nicht erhalten.

Maßnahme
Reduzieren Sie den Speicherverbrauch anderer Programme oder vergrößern Sie den Hauptspeicher Ihres Systems. Starten Sie *ocpic_adm* dann erneut.

```
ocpic_adm 03 Interner Fehler: &ERRCODE.
```

Bedeutung
Im Programm *ocpic_adm* ist ein interner Fehler aufgetreten.

Maßnahme
Verständigen Sie Ihren Service.

`ocpic_adm 04 Fehler beim Oeffnen der Pipe &PIPENAME : errno &ERRNO (&ERRNAME)`

Bedeutung
Die Pipe *&PIPENAME* konnte mit dem Systemaufruf *open()* nicht geöffnet werden. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Verständigen Sie ggf. Ihren Systemverwalter.

`ocpic_adm 05 Fehler beim Erzeugen der Pipe &PIPENAME: errno &ERRNO (&ERRNAME)`

Bedeutung
Die Pipe *&PIPENAME* konnte mit dem Systemaufruf *creat()* nicht erzeugt werden. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Verständigen Sie ggf. Ihren Systemverwalter.

`ocpic_adm 06 OpenCPIC-Manager ist nicht gestartet`

Bedeutung
Der OpenCPIC-Manager ist noch nicht gestartet und der von Ihnen abgesetzte Administrationsbefehl ist somit wirkungslos.

`ocpic_adm 07 Fehler beim Schreiben in die Pipe &PIPENAME: errno &ERRNO (&ERRNAME )`

Bedeutung
In die Pipe *&PIPENAME* konnte mit dem Systemaufruf *write()* nicht geschrieben werden. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Verständigen Sie ggf. Ihren Systemverwalter.

`ocpic_adm 08 Fehler beim Lesen der Pipe &PIPENAME: errno &ERRNO (&ERRNAME)`

Bedeutung
Aus der Pipe *&PIPENAME* konnte mit dem Systemaufruf *read()* nicht gelesen werden. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Verständigen Sie ggf. Ihren Systemverwalter.

`ocpic_adm 09 Der partappl_name ist nicht generiert`

Bedeutung
Sie haben beim Aufruf von *ocpic_adm -d* bzw. *ocpic_adm -a* einen ungültigen *partappl_name* angegeben. Der *partappl_name* muss in der Generierungsdatei mit einer PARTAPPL-Anweisung generiert worden sein.

Maßnahme
Überprüfen Sie den *partappl_name* und die aktuelle Konfigurierung. Mit dem Kommando *ocpic_gen -r* können Sie sich den Inhalt der momentan benutzten Konfigurationsdatei anzeigen lassen

```
ocpic_adm 11 Der XAP-TP-Trace ist bereits eingeschaltet
```

Bedeutung
Sie haben *ocpic_adm -t ton* bzw. *ocpic_adm -t bon* aufgerufen, obwohl der XAP-TP-Trace bereits eingeschaltet war. Der Aufruf ist ohne Wirkung auf den XAP-TP-Trace.

```
ocpic_adm 12  Der Manager-Trace ist bereits eingeschaltet
```

Bedeutung
Sie haben *ocpic_adm -t mon* bzw. *ocpic_adm -t bon* aufgerufen, obwohl der Manager-Trace bereits eingeschaltet war. Der Aufruf ist ohne Wirkung auf den Manager-Trace.

```
ocpic_adm 13 Der XAP-TP-Trace war nicht eingeschaltet
```

Bedeutung
Sie haben *ocpic_adm -t tof* bzw. *ocpic_adm -t bof* aufgerufen, obwohl der XAP-TP-Trace nicht eingeschaltet war. Der Aufruf ist ohne Wirkung auf den XAP-TP-Trace.

```
ocpic_adm 14 Der Manager-Trace war nicht eingeschaltet
```

Bedeutung
Sie haben *ocpic_adm -t mof* bzw. *ocpic_adm -t bof* aufgerufen, obwohl der Manager-Trace nicht eingeschaltet war. Der Aufruf ist ohne Wirkung auf den Manager-Trace.

```
ocpic_adm 15 Es gibt derzeit keine freien Associations
```

Bedeutung
Alle Associations zwischen Ihrer lokalen Anwendung und der Partneranwendung *partappl_name* sind momentan belegt. Der Aufruf von *ocpic_adm -d* ist daher ohne Wirkung.

Maßnahme
Mit dem Kommando *ocpic_sta* können Sie nähere Informationen über den Zustand der Associations erhalten.

```
ocpic_adm 16 Es gibt derzeit keine Associations
```

Bedeutung
Zwischen Ihrer lokalen Anwendung und der Partneranwendung *partappl_name* bestehen momentan keine Associations. Der Aufruf von *ocpic_adm -a* ist daher ohne Wirkung.

```
ocpic_adm 17 Systemfehler beim OpenCPIC-Manager
```

Bedeutung
Es ist im Zusammenhang mit der Administration ein Systemfehler beim OpenCPIC-Manager aufgetreten.

Maßnahme
Ein erste Diagnosehilfe bietet u. U. die Datei *$OCPICDIR/PROT/prot.mgr*. Sichern Sie den Inhalt des Verzeichnisses *$OCPICDIR/PROT* und verständigen Sie Ihren Service.

```
ocpic_adm 18 Fehler beim Oeffnen der Datei &FILENAME: errno &ERRNO (&ERRNAME)
```

Bedeutung
Die Datei *&FILENAME* konnte mit dem Systemaufruf *open()* nicht geöffnet werden. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Prüfen und ändern Sie ggf. die Zugriffsrechte.

```
ocpic_adm 20 Fehler beim Lesen der PID aus der Datei &FILENAME: errno &ERRNO (&ERRNAME)
```

Bedeutung
Beim Lesen aus der Datei *&FILENAME* mit dem Systemaufruf *read()* ist ein Fehler aufgetreten. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Prüfen und ändern Sie ggf. die Zugriffsrechte.

```
ocpic_adm 21 Fehler beim Senden eines Signals: errno &ERRNO (&ERRNAME )
```

Bedeutung
Beim Senden eines Signals durch *ocpic_adm* ist ein Fehler aufgetreten. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Verständigen Sie ggf. Ihren Systemverwalter.

```
ocpic_adm 23 Fehler beim Erzeugen der Datei &FILENAME: errno &ERRNO (&ERRNAME)
```

Bedeutung
Beim Erzeugen der Datei *&FILENAME* mit dem Systemaufruf *creat()* ist ein Fehler aufgetreten. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Prüfen und ändern Sie ggf. die Zugriffsrechte.

```
ocpic_adm 24 Fehler beim Loeschen der Datei &FILENAME: errno &ERRNO (&ERRNAME)
```

Bedeutung
Beim Löschen der Datei *&FILENAME* mit dem Systemaufruf *unlink()* ist ein Fehler aufgetreten. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Prüfen und ändern Sie ggf. die Zugriffsrechte.

```
ocpic_adm 27 Kein Flush moeglich, da der XAP-TP-Trace nicht eingeschaltet ist
```

Bedeutung
Sie haben *ocpic_adm -t tfl* bzw. *ocpic_adm -t bfl* aufgerufen, obwohl der XAP-TP-Trace nicht eingeschaltet war. Der Aufruf ist ohne Wirkung auf den XAP-TP-Trace.

```
ocpic_adm 28 Kein Flush moeglich, da der Manager-Trace nicht eingeschaltet ist
```

Bedeutung
Sie haben *ocpic_adm -t mfl* bzw. *ocpic_adm -t bfl* aufgerufen, obwohl der Manager-Trace nicht eingeschaltet war. Der Aufruf ist ohne Wirkung auf den Manager-Trace.

```
ocpic_adm 29 OCPICDIR = &PATHNAME ist zu lang
```

Bedeutung
Der Inhalt *&PATHNAME* der Umgebungsvariablen *OCPICDIR* hat die zulässige Maximallänge von 128 Zeichen überschritten. *ocpic_adm* bricht deshalb ab.

```
ocpic_adm 30 partappl_name = &NAME ist zu lang
```

Bedeutung
Sie haben einen fehlerhaften *partappl_name* (*&NAME*) als Argument übergeben. Der *partappl_name* darf nicht länger als 78 Zeichen lang sein.

```
ocpic_adm 34 OpenCPIC-Manager befindet sich noch in der Initialisierungsphase
```

Bedeutung
Solange sich der OpenCPIC-Manager noch in der Initialisierungsphase befindet, ist keine Administration möglich.

Maßnahme
Rufen Sie das Administrationskommando zu einem späteren Zeitpunkt noch einmal auf.

9.4 Meldungen des Programms ocpic_sta

```
ocpic_sta 00 Message catalog ocpic3.cat not opened
```

Bedeutung
Der NLS-Meldungskatalog *ocpic3.cat* konnte nicht geöffnet werden. Er wird abhängig vom Inhalt der Umgebungsvariablen *$LANG* im Verzeichnis */opt/lib/nls/msg/De* bzw. */opt/lib/nls/msg/En* gesucht.

Maßnahme
Stellen Sie den Meldungskatalog wieder her, evtl. durch Neuinstallation des Produktes, oder ändern Sie den Inhalt von *$LANG*.

```
ocpic_sta 01 Aufruf: ocpic_sta [-l]
```

Bedeutung
Sie haben *ocpic_sta* mit falschen Parametern aufgerufen.

```
ocpic_sta 02 Nicht genug Speicherplatz vorhanden
```

Bedeutung
ocpic_sta hat mit der Systemfunktion *malloc()* Speicherplatz vom Betriebssystem angefordert, diesen jedoch nicht erhalten.

Maßnahme
Reduzieren Sie den Speicherverbrauch anderer Programme oder vergrößern Sie den Hauptspeicher Ihres Systems. Starten Sie *ocpic_sta* dann erneut.

```
ocpic_sta 03 Interner Fehler: &ERRCODE.
```

Bedeutung
Im Programm *ocpic_sta* ist ein interner Fehler aufgetreten.

Maßnahme
Verständigen Sie Ihren Service.

```
ocpic_sta 04 Fehler beim Oeffnen der Pipe &PIPENAME: errno &ERRNO (&ERRNAME)
```

Bedeutung
Die angegebene Pipe konnte mit dem Systemaufruf *open()* nicht geöffnet werden. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Verständigen Sie ggf. Ihren Systemverwalter.

```
ocpic_sta 05 Fehler beim Erzeugen der Pipe &PIPENAME errno &ERRNO (&ERRNAME)
```

Bedeutung
Die Pipe *&PIPENAME* konnte mit dem Systemaufruf *creat()* nicht erzeugt werden. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Verständigen Sie ggf. Ihren Systemverwalter.

```
ocpic_sta 06 Fehler beim Schreiben in die Pipe &PIPENAME: errno &ERRNO (&ERRNAME)
```

Bedeutung
In die Pipe *&PIPENAME* konnte mit dem Systemaufruf *write()* nicht geschrieben werden. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Verständigen Sie ggf. Ihren Systemverwalter.

```
ocpic_sta 07 Fehler beim Lesen der Pipe &PIPENAME : errno &ERRNO (&ERRNAME )
```

Bedeutung
Aus der Pipe *&PIPENAME* konnte mit dem Systemaufruf *read()* nicht gelesen werden. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Verständigen Sie ggf. Ihren Systemverwalter.

```
ocpic_sta 08 Systemfehler beim OpenCPIC-Manager
```

Bedeutung
Es ist im Zusammenhang mit der Administration ein Systemfehler beim OpenCPIC-Manager aufgetreten.

Maßnahme
Ein erste Diagnosehilfe bietet u. U. die Datei *$OCPICDIR/PROT/prot.mgr*. Sichern Sie den Inhalt des Verzeichnisses *$OCPICDIR/PROT* und verständigen Sie Ihren Service.

```
ocpic_sta 09 Fehler beim Oeffnen der Datei &FILEMAE: errno &ERRNO (&ERRNAME)
```

Bedeutung
Die Datei *&FILENAME* konnte mit dem Systemaufruf *open()* nicht geöffnet werden. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Prüfen und ändern Sie ggf. die Zugriffsrechte.

```
ocpic_sta 10 Fehler beim Lesen der Datei &FILENAME: errno &ERRNO (&ERRNAME)
```

Bedeutung
Beim Lesen aus der Datei *&FILENAME* mit dem Systemaufruf *read()* ist ein Fehler aufgetreten. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*.

`ocpic_sta 11 OCPICDIR = &PATHNAME ist zu lang`

Bedeutung
Der Inhalt *&PATHNAME* der Umgebungsvariablen *OCPICDIR* hat die zulässige Länge von 128 Zeichen überschritten. *ocpic_sta* bricht deshalb ab.

`ocpic_sta 12 Fehler beim Schliessen der Datei &FILENAME: errno &ERRNO (&ERRNAME)`

Bedeutung
Beim Schließen der Datei *&FILENAME* mit dem Systemaufruf *close()* ist ein Fehler aufgetreten. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*.

`ocpic_sta 13 Fehler beim Loeschen der Datei &FILENAME: errno &ERRNO (&ERRNAME )`

Bedeutung
Beim Löschen der Datei *&FILENAME* mit dem Systemaufruf *unlink()* ist ein Fehler aufgetreten. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*.

`ocpic_sta 14 Fehler beim Lesen der PID aus der Datei &FILENAME: errno &ERRNO (&ERRNAME)`

Bedeutung
Beim Lesen aus der Datei *&FILENAME* mit dem Systemaufruf *read()* ist ein Fehler aufgetreten. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*.

`ocpic_sta 15 Fehler beim Senden eines Signals an den OpenCPIC-Manager: errno &ERRNO (&ERRNAME)`

Bedeutung
Beim Senden eines Signals durch *ocpic_adm* ist ein Fehler aufgetreten. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*.

9.5 Meldungen des Programms ocpic_logdump

Die folgenden Meldungen werden vom Programm *ocpic_logdump* ausgegeben. *ocpic_logdump* ist ein internes Programm zur Auswertung von Traceinformationen und wird z. B. von *ocpic_adm -l* und *ocpic_tredit* aufgerufen.

```
ocpic_logdump 11 Fehler beim Oeffnen der Datei &FILENAME, errno = &ERRNO
```

Bedeutung
Die Datei *&FILENAME* konnte mit dem Systemaufruf *open()* nicht geöffnet werden. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Prüfen und ändern Sie ggf. die Zugriffsrechte.

```
ocpic_logdump 12 Fehler beim Lesen der Datei &FILENAME, errno = &ERRNO
```

Bedeutung
Beim Lesen aus der Datei *&FILENAME* mit dem Systemaufruf *open()* ist ein Fehler aufgetreten. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*. Prüfen und ändern Sie ggf. die Zugriffsrechte.

```
ocpic_logdump 13 Nicht genug Speicherplatz vorhanden
```

Bedeutung
ocpic_logdump hat mit der Systemfunktion *malloc()* Speicherplatz vom Betriebssystem angefordert, diesen jedoch nicht erhalten.

Maßnahme
Reduzieren Sie den Speicherverbrauch anderer Programme oder vergrößern Sie den Hauptspeicher Ihres Systems. Rufen Sie das Kommando dann nochmals auf.

```
ocpic_logdump 14 Fehler beim Schliessen der Datei &FILENAME, errno = &ERRNO
```

Bedeutung
Beim Schließen der Datei *&FILENAME* mit dem Systemaufruf *close()* ist ein Fehler aufgetreten. Die genauere Ursache wird durch den Inhalt der Systemvariablen *errno = &ERRNO* angegeben.

Maßnahme
Abhängig von *&ERRNO*.

9.6 Meldungen des Programms xatmigen

Die Meldungen von *xatmigen* haben die Form `XGnn meldungstext...` und werden nach *stdout* ausgegeben.

`XG01` `Generierung des Local Configuration Files: &LCF / &DEF / &CODE`

Bedeutung
Startmeldung des Programms .
&LCF Name des erzeugten Local Configuration Files
&DEF Name des erzeugten Generierungsfragments
&CODE Stringcode für Character-Arrays

`XG02` `Generierung erfolgreich beendet`

Bedeutung
Das LCF wurde erzeugt, die Generierung wurde erfolgreich beendet.

`XG03` `Generierung erfolgreich mit Warnungen beendet`

Bedeutung
Das LCF wurde erzeugt. Es wird jedoch mindestens eine Warnung ausgegeben (z. B. weil nicht benötigte Dateien angegeben wurden). Diese Warnung hat auf die Generierung keinen Einfluss.

`XG04` `Generierung wegen Fehler beendet`
`Keine Datei erzeugt.`

Bedeutung
Das LCF wurde nicht erzeugt, die Generierung konnte nicht durchgeführt werden. Die Ursache geht aus den vorhergehenden Meldungen hervor.

`XG05` `&FTYPE Datei '&FNAME'`

Bedeutung
Diese Meldung gibt die gerade bearbeitete Datei an in folgender Form:

&FTYPE	„Description“-File	enthält Datenstrukturen
	„Definition“-File	enthält den LCF-Input
	„LC“-File	enthält die Local Configuration
&FNAME	Filename	

`XG10` `Aufruf: &PARAM`

Bedeutung
Syntaxfehler beim Aufruf von *xatmigen*:
&PARAM Aufrufparameter und Schalter

XG11 `[Error] &FTYPE Datei 'FNAME' kann nicht erzeugt werden: &REASON`

Bedeutung
Die Datei *&FNAME* des Typs *&FTYPE* kann nicht erzeugt werden.
&REASON enthält eine nähere Begründung.
&FTYPE GEN = Generation Fragment File
LC = Local Configuration File

XG12 `[Warning] Datei nicht gefunden.`

Bedeutung
Das Definition File oder eine Description File wurde nicht gefunden; möglicherweise existiert es nicht.

XG13 `[Warning] Zu viele &OBJECTS, Maximum: &MAXNUM`

Bedeutung
Meldung über zu viele gefundene Objekte
&OBJECTS Subtypen
&MAXNUM Maximale Anzahl

XG14 `[Error] Zeile &LINE: Syntaxfehler, &HELPTEXT`

Bedeutung
Syntaxfehler in Zeile *&LINE* in der LC-Definitionsdatei
&HELPTEXT Hilfetext

XG15 `[Error] Zeile &LINE: Keine Record-Definition gefunden für Puffer &BUFF`

Bedeutung
Für den Puffer *&BUFF* in Zeile *&LINE* konnte keine zugehörige Record-Definition gefunden werden.

XG16 `[Error] Zeile &LINE: Basistyp-Fehler in Puffer &BUFF`

Bedeutung
Die Syntaxbeschreibung des Puffers *&BUFF* in Zeile *&LINE* des LCF enthält einen falschen Basistype (int, short usw.) oder es wurde bei COBOL ein unzulässiger Stringcode angegeben (z. B. "P").

XG17 `[Error] &FTYPE Datei '&FNAME' kann nicht geöffnet werden. &REASON`

Bedeutung
Die Datei *&FNAME* des Typs *&FTYPE* kann nicht geöffnet werden.
&REASON enthält eine nähere Begründung.
&FTYPE DEF = LC-Definitionsdatei

XG18 `[Error] &REASON`

Bedeutung
Allgemeine Fehlermeldung. *&REASON* enthält eine nähere Beschreibung des Fehlers.

XG19 `[Message] Neuen Puffer erzeugt: &BUFF`

Bedeutung
Es wurde der neue Puffer *&BUFF* erzeugt.

XG20 `[Message] Servicename ‘&SVC‘ auf 16 Zeichen gekuerzt!`

Bedeutung
&SVC: Servicename

XG21 `[Message] Zeile &LINE: unbekannte Anweisungszeile ‘&HELPTEXT‘`

Bedeutung
Meldung für die Zeile *&LINE* in der LC-Definitionsdatei
&HELPTEXT Hilfetext (ein Teil der LC-Zeile)

XG22 `[Message] Zeile &LINE: Standardwert gesetzt MODE=‘&TEXT‘`

Bedeutung
Meldung für die Zeile *&LINE* in der LC-Definitionsdatei
&TEXT gesetzter Default Servicemode

10 Anhang

10.1 Zeichensätze

In OSI-TP werden die ASN.1-Character-String-Typen *PrintableString* und *T61String* verwendet (siehe [29]). CPI-C verwendet die Zeichensätze 01134 und 00640.

Der Zeichensatz *T61String* enthält alle in der nachfolgenden Code-Tabelle definierten Zeichen. Die vollständige CCITT-Dokumentation finden Sie in [34].

Code-Tabelle des Zeichensatzes T61

	0	1	2	3	4	5	6	7	8	9	...	F
0			SP	**0**	**@**	**P**		**p**				
1			**!**	**1**	**A**	**Q**	**a**	**q**				
2			**“**	**2**	**B**	**R**	**b**	**r**				
3			**#**	**3**	**C**	**S**	**c**	**s**				
4				**4**	**D**	**T**	**d**	**t**				
5			**%**	**5**	**E**	**U**	**e**	**u**				
6			**&**	**6**	**F**	**V**	**f**	**v**				
7			**‘**	**7**	**G**	**W**	**g**	**w**				
8	BS		**(**	**8**	**H**	**X**	**h**	**x**				
9		SS2	**)**	**9**	**I**	**Y**	**i**	**y**				
A	LF	SUB	*****	**:**	**J**	**Z**	**j**	**z**				
B		ESC	**+**	**;**	**K**	**[**	**k**		PLD	CSI		
C	FF		**,**	**<**	**L**		**l**	**|**	PLU			
D	CR	SS3	**-**	**=**	**M**	**]**	**m**					
E	LS1		**.**	**>**	**N**		**n**					
F	LS0		**/**	**?**	**O**	_	**o**					

Bedeutung der Abkürzungen:

BS	BACKSPACE	LS1	LOCKING SHIFT ONE
CR	CARRIAGE RETURN	PLD	PARTIAL LINE DOWN
CSI	CONTROL SEQUENCE INTRODUCER	PLU	PARTIAL LINE UP
ESC	ESCAPE	SP	SPACE
FF	FORM FEED	SS2	SINGLE SHIFT TWO
LF	LINE FEED	SS3	SINGLE SHIFT THREE
LS0	LOCKING SHIFT ZERO	SUB	SUBSTITUTE CHARACTER

Die Zeichensätze *PrintableString*, 01134 und 00640 sind Untermengen des Zeichensatzes *T61String*. Die nachfolgende Tabelle gibt an, welche Zeichen im jeweiligen Zeichensatz zulässig sind.

Tabelle der Zeichensätze PrintableString, 01134 und 00640

Zeichen	PrintableString	01134	00640
SP	x		x
.	x		x
<			x
(	x		x
+	x		x
&			x
*			x
)	x		x
;			x
-	x		x
/	x		x
,	x		x
%			x
_			x
>			x
?	x		x
:	x		x
‘	x		x
=	x		x
“			x
a-z	x		x
A-Z	x	x	x
0-9	x	x	x

10.2 Code-Konvertierungstabellen

Die folgenden Konvertierungstabellen werden bei der Konvertierung von ASCII nach EBCDIC durch den CPI-C-Aufruf *Convert_Outgoing (CMCNVO)* bzw. von EBCDIC nach ASCII durch den CPI-C-Aufruf *Convert_Incoming (CMCNVI)* verwendet.

ASCII nach EBCDIC Konvertierung

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	00	01	02	03	37	2D	2E	2F	16	05	25	0B	0C	0D	0E	0F
1	10	11	12	13	3C	3D	32	26	18	19	3F	27	1C	1D	1E	1F
2	40	4F	7F	7B	5B	6C	50	7D	4D	5D	5C	4E	6B	60	4B	61
3	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	7A	5E	4C	7E	6E	6F
4	7C	C1	C2	C3	C4	C5	C6	C7	C8	C9	D1	D2	D3	D4	D5	D6
5	D7	D8	D9	E2	E3	E4	E5	E6	E7	E8	E9	4A	E0	5A	5F	6D
6	79	81	82	83	84	85	86	87	88	89	91	92	93	94	95	96
7	97	98	99	A2	A3	A4	A5	A6	A7	A8	A9	C0	6A	D0	A1	07
8	20	21	22	23	24	15	06	17	28	29	2A	2B	2C	09	0A	1B
9	30	31	1A	33	34	35	36	08	38	39	3A	3B	04	14	3E	E1
A	41	42	43	44	45	46	47	48	49	51	52	53	54	55	56	57
B	58	59	62	63	64	65	66	67	68	69	70	71	72	73	74	75
C	76	77	78	80	8A	8B	8C	8D	8E	8F	90	9A	9B	9C	9D	9E
D	9F	A0	AA	AB	AC	AD	AE	AF	B0	B1	B2	B3	B4	B5	B6	B7
E	B8	B9	BA	BB	BC	BD	BE	BF	CA	CB	CC	CD	CE	CF	DA	DB
F	DC	DD	DE	DF	EA	EB	EC	ED	EE	EF	FA	FB	FC	FD	FE	FF

EBCDIC nach ASCII Konvertierung

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
0	00	01	02	03	9C	09	86	7F	97	8D	8E	0B	0C	0D	0E	0F
1	10	11	12	13	9D	85	08	87	18	19	92	8F	1C	1D	1E	1F
2	80	81	82	83	84	0A	17	1B	88	89	8A	8B	8C	05	06	07
3	90	91	16	93	94	95	96	04	98	99	9A	9B	14	15	9E	1A
4	20	A0	A1	A2	A3	A4	A5	A6	A7	A8	5B	2E	3C	28	2B	21
5	26	A9	AA	AB	AC	AD	AE	AF	B0	B1	5D	24	2A	29	3B	5E
6	2D	2F	B2	B3	B4	B5	B6	B7	B8	B9	7C	2C	25	5F	3E	3F
7	BA	BB	BC	BD	BE	BF	C0	C1	C2	60	3A	23	40	27	3D	22
8	C3	61	62	63	64	65	66	67	68	69	C4	C5	C6	C7	C8	C9
9	CA	6A	6B	6C	6D	6E	6F	70	71	72	CB	CC	CD	CE	CF	D0
A	D1	7E	73	74	75	76	77	78	79	7A	D2	D3	D4	D5	D6	D7
B	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7
C	7B	41	42	43	44	45	46	47	48	49	E8	E9	EA	EB	EC	ED
D	7D	4A	4B	4C	4D	4E	4F	50	51	52	EE	EF	F0	F1	F2	F3
E	5C	9F	53	54	55	56	57	58	59	5A	F4	F5	F6	F7	F8	F9
F	30	31	32	33	34	35	36	37	38	39	FA	FB	FC	FD	FE	FF

Abkürzungen

ACSE	Association Control Service Element
AEQ	Application Entity Qualifier
AES	Advanced Encryption Standard
AET	Application Entity Title
AP	Anwendungsprogramm
APT	Application Process Title
ASCII	American Standard Code for Information Interchange
ASE	Application Service Element
Axis	Apache eXtensible Interaction System
BCAM	Basic Communication Access Method
BER	Basic Encoding Rules
BLS	Binder-Lader-Starter (BS2000-Systeme)
CCP	Communication Control Program
CCR	Commitment, Concurrency and Recovery
CCS	Codierter Zeichensatz (Coded Character Set)
CCSN	Name des codierten Zeichensatzes (Coded Character Set Name)
CICS	Customer Information Control System (IBM)
CID	Control Identification
CMX	Communication Manager in Unix-, Linux- und Windows-Systemen
COM	Component Object Model
CPI-C	Common Programming Interface for Communication
CRM	Communication Resource Manager
CRTE	Common Runtime Environment (BS2000-Systeme)
DB	Datenbank
DC	Data Communication

DCAM	Data Communication Access Method
DES	Data Encryption Standard
DLM	Distributed Lock Manager (BS2000-Systeme)
DMS	Data Management System
DNS	Domain Name Service
DSS	Datensichtstation (=Terminal)
DTD	Document Type Definition
DTP	Distributed Transaction Processing
DVS	Datenverwaltungssystem
EBCDIC	Extended Binary-Coded Decimal Interchange Code
EJB	Enterprise JavaBeans™
FGG	File Generation Group
FHS	Format Handling System
FT	File Transfer
GSSB	Globaler Sekundärer Speicherbereich
HIPLEX®	Highly Integrated System Complex (BS2000-Systeme)
HLL	High-Level Language
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IFG	Interaktiver Format-Generator
ILCS	Inter Language Communication Services (BS2000-Systeme)
IMS	Information Management System (IBM)
IPC	Inter-Process-Communication
IRV	Internationale Referenzversion
ISO	International Organization for Standardization
Java EE	Java Platform, Enterprise Edition
JCA	Java EE Connector Architecture
JDK	Java Development Kit
KAA	KDCS Application Area
KB	Kommunikationsbereich
KBPROG	KB-Programmbereich
KDCADMI	KDC Administration Interface

KDCS	Kompatible Datenkommunikationsschnittstelle
KTA	KDCS Task Area
LAN	Local Area Network
LCF	Local Configuration File
LLM	Link and Load Module (BS2000-Systeme)
LSSB	Lokaler Sekundärer Speicherbereich
LU	Logical Unit
MQ	Message Queuing
MSCF	Multiple System Control Facility (BS2000-Systeme)
NB	Nachrichtenbereich
NEA	Netzwerkarchitektur bei BS2000-Systemen
NFS	Network File System/Service
NLS	Unterstützung der Landessprache (Native Language Support)
OLTP	Online Transaction Processing
OML	Object Modul Library
OSI	Open System Interconnection
OSI TP	Open System Interconnection Transaction Processing
OSS	OSI Session Service
PCMX	Portable Communication Manager
PID	Prozess-Identifikation
PIN	Persönliche Identifikationsnummer
PLU	Primary Logical Unit
PTC	Prepare to commit
RAV	Rechenzentrums-Abrechnungs-Verfahren
RDF	Resource Definition File
RM	Resource Manager
RSA	Encryption-Algorithmus nach Rivest, Shamir, Adleman
RSO	Remote SPOOL Output (BS2000-Systeme)
RTS	Runtime System (Laufzeitsystem)
SAT	Security Audit Trail (BS2000-Systeme)
SECOS	Security Control System
SEM	SE Manager

SGML	Standard Generalized Markup Language
SLU	Secondary Logical Unit
SM2	Software Monitor 2
SNA	Systems Network Architecture
SOA	Service-oriented Architecture
SOAP	Simple Object Access Protocol
SPAB	Standard Primärer Arbeitsbereich
SQL	Structured Query Language
SSB	Sekundärer Speicherbereich
SSO	Single-Sign-On
TAC	Transaktionscode
TCEP	Transport Connection End Point
TCP/IP	Transport Control Protocol / Internet Protocol
TIAM	Terminal Interactive Access Method
TLS	Terminal-spezifischer Langzeitspeicher
TM	Transaction Manager
TNS	Transport Name Service
TP	Transaction Processing (Transaktions-Betrieb)
TPR	Task privileged (privilegierter Funktionszustand des BS2000-Systems)
TPSU	Transaction Protocol Service User
TSAP	Transport Service Access Point
TSN	Task Sequence Number
TU	Task user (nicht privilegierter Funktionszustand des BS2000-Systems)
TX	Transaction Demarcation (X/Open)
UDDI	Universal Description, Discovery and Integration
UDS	Universelles Datenbanksystem
UDT	Unstructured Data Transfer
ULS	User-spezifischer Langzeitspeicher
UPIC	Universal Programming Interface for Communication
USP	UTM-Socket-Protokoll
UTM	Universeller Transaktionsmonitor
UTM-D	UTM-Funktionen für verteilte Verarbeitung („Distributed“)
UTM-F	Schnelle UTM-Variante („Fast“)

UTM-S	UTM-Sicherheitsvariante
UTM-XML	XML-Schnittstelle von openUTM
VGID	Vorgangs-Identifikation
VTSU	Virtual Terminal Support
VTV	Verteilte Transaktionsverarbeitung
VV	Verteilte Verarbeitung
WAN	Wide Area Network
WS4UTM	WebServices for openUTM
WSDD	Web Service Deployment Descriptor
WSDL	Web Services Description Language
XA	X/Open Access Interface (Schnittstelle von X/Open zum Zugriff auf Resource Manager)
XAP	X/OPEN ACSE/Presentation programming interface
XAP-TP	X/OPEN ACSE/Presentation programming interface Transaction Processing extension
XATMI	X/Open Application Transaction Manager Interface
XCS	Cross Coupled System
XHCS	eXtended Host Code Support
XML	eXtensible Markup Language

Fachwörter

Fachwörter, die an anderer Stelle erklärt werden, sind mit *kursiver* Schrift ausgezeichnet.
UTM-spezifische Fachwörter finden Sie in den Fachwortverzeichnissen der UTM-Manuale.

Abstrakte Syntax (OSI)
Eine abstrakte Syntax ist die Menge der formal beschriebenen Datentypen, die zwischen Anwendungen über *OSI TP* ausgetauscht werden sollen. Eine abstrakte Syntax ist unabhängig von der eingesetzten Hardware und der jeweiligen Programmiersprache.

Access Point (OSI)
siehe *Dienstzugriffspunkt*.

ACID-Eigenschaften
Abkürzende Bezeichnung für die grundlegenden Eigenschaften von *Transaktionen*: Atomicity, Consistency, Isolation und Durability.

Akzeptor (CPI-C)
Die Kommunikationspartner einer *Conversation* werden *Initiator* und Akzeptor genannt. Der Akzeptor nimmt die vom Initiator eingeleitete Conversation mit Accept_Conversation entgegen.

Application Context (OSI)
Der Application Context ist die Menge der Regeln, die für die Kommunikation zwischen zwei Anwendungen gelten sollen. Dazu gehören z.B. die *abstrakten Syntaxen* und die zugeordneten *Transfer-Syntaxen*.

Application Entity (OSI)
Eine Application Entity (AE) repräsentiert alle für die Kommunikation relevanten Aspekte einer realen Anwendung. Eine Application Entity wird durch einen global (d.h. weltweit) eindeutigen Namen identifiziert, den *Application Entity Title* (AET). Jede Application Entity repräsentiert genau einen *Application Process*. Ein Application Process kann mehrere Application Entities umfassen.

Application Entity Qualifier (OSI)

Bestandteil des *Application Entity Titles*. Der Application Entity Qualifier identifiziert einen *Dienstzugriffspunkt* innerhalb der Anwendung. Ein Application Entity Qualifier kann unterschiedlich aufgebaut sein. openUTM unterstützt den Typ "Zahl".

Application Entity Title (OSI)

Ein Application Entity Title ist ein global (d.h. weltweit) eindeutiger Name für eine *Application Entity*. Er setzt sich zusammen aus dem *Application Process Title* des jeweiligen *Application Process* und dem *Application Entity Qualifier*.

Application Process (OSI)

Der Application Process repräsentiert im *OSI-Referenzmodell* eine Anwendung. Er wird durch den *Application Process Title* global (d.h. weltweit) eindeutig identifiziert.

Application Process Title (OSI)

Gemäß der OSI-Norm dient der Application Process Title (APT) zur global (d.h. weltweit) eindeutigen Identifizierung von Anwendungen. Er kann unterschiedlich aufgebaut sein. openUTM unterstützt den Typ *Object Identifier*.

Application Service Element (OSI)

Ein Application Service Element (ASE) repräsentiert eine Funktionsgruppe der Anwendungsschicht (Schicht 7) des *OSI-Referenzmodells*.

Association (OSI)

Eine Association ist eine Kommunikationsbeziehung zwischen zwei *Application Entities*. Dem Begriff Association entspricht der *LU6.1*-Begriff *Session*.

Asynchron-Conversation

CPI-C-Conversation, bei der nur der *Initiator* senden darf. Für den *Akzeptor* muss in der *UTM-Anwendung* ein asynchroner Transaktionscode generiert sein.

CCR (Commitment, Concurrency and Recovery)

CCR ist ein von OSI definiertes Application Service Element (ASE) für die OSI-TP-Kommunkation, welches die Protokollelemente (Services) zum Beginn und Abschluss (Commit oder Rollback) einer *Transaktion* enthält. CCR unterstützt das Zwei-Phasen-Commitment.

CCS-Name (BS2000-Systeme)

siehe *Coded-Character-Set-Name*.

Client

Clients einer *UTM-Anwendung* können sein:

– Terminals
– UPIC-Client-Programme
– Transportsystem-Anwendungen (z.B. DCAM-, PDN-, CMX-, Socket-Anwendungen oder UTM-Anwendungen, die als *Transportsystem-Anwendung* generiert sind)

Clients werden über LTERM-Partner an die UTM-Anwendung angeschlossen. openUTM-Clients mit Trägersystem OpenCPIC werden wie *OSI TP-Partner* behandelt.

Client-Seite einer Conversation

Begriff ersetzt durch *Initiator*.

Coded-Character-Set-Name (BS2000-Systeme)

Bei Verwendung des Produkts *XHCS* (e**X**tended **H**ost **C**ode **S**upport) wird jeder verwendete Zeichensatz durch einen Coded-Character-Set-Namen (abgekürzt: "CCS-Name" oder "CCSN") eindeutig identifiziert.

Communication Resource Manager

Communication Resource Manager (CRMs) kontrollieren in verteilten Systemen die Kommunikation zwischen den Anwendungsprogrammen. openUTM stellt CRMs für den internationalen Standard OSI TP, für den Industrie-Standard *LU6.1* und für das openUTM-eigene Protokoll UPIC zur Verfügung.

Contention Loser

Jede Verbindung zwischen zwei Partnern wird von einem der Partner verwaltet. Der Partner, der die Verbindung verwaltet, heißt *Contention Winner*. Der andere Partner ist der Contention Loser.

Contention Winner

Der Contention Winner einer Verbindung übernimmt die Verwaltung der Verbindung. Aufträge können sowohl vom Contention Winner als auch vom *Contention Loser* gestartet werden. Im Konfliktfall, wenn beide Kommunikationspartner gleichzeitig einen Auftrag starten wollen, wird die Verbindung vom Auftrag des Contention Winner belegt.

Conversation

Bei CPI-C nennt man die Kommunikation zwischen zwei CPI-C-Anwendungsprogrammen Conversation. Die Kommunikationspartner einer Conversation werden *Initiator* und *Akzeptor* genannt.

Conversation-ID

Jeder *Conversation* wird von CPI-C lokal eine Conversation-ID zugeordnet, d.h. *Initiator* und *Akzeptor* haben jeweils eine eigene Conversation-ID. Mit der Conversation-ID wird jeder CPI-C-Aufruf innerhalb eines Programms eindeutig einer Conversation zugeordnet.

CPI-C

CPI-C (Common Programming Interface for Communication) ist eine von X/Open und dem CIW (**C**PI-C **I**mplementor's **W**orkshop) normierte Programmschnittstelle für die Programm-Programm-Kommunikation in offenen Netzen. Das in openUTM implementierte CPI-C genügt der CPI-C V2.0 CAE Specification von X/Open. Die Schnittstelle steht in COBOL und C zur Verfügung. CPI-C in openUTM kann über die Protokolle OSI TP, LU6.1, UPIC und mit openUTM-LU6.2 kommunizieren.

Dialog-Conversation

CPI-C-Conversation, bei der sowohl der *Initiator* als auch der *Akzeptor* senden darf. Für den *Akzeptor* muss in der *UTM-Anwendung* ein Dialog-Transaktionscode generiert sein.

Dienstzugriffspunkt

Im *OSI-Referenzmodell* stehen einer Schicht am Dienstzugriffspunkt die Leistungen der darunterliegenden Schicht zur Verfügung. Der Dienstzugriffspunkt wird im lokalen System durch einen *Selektor* identifiziert. Bei der Kommunikation bindet sich die *UTM-Anwendung* an einen Dienstzugriffspunkt. Eine Verbindung wird zwischen zwei Dienstzugriffspunkten aufgebaut.

Distributed Transaction Processing

X/Open-Architekturmodell für die transaktionsorientierte *verteilte Verarbeitung*.

Funktionseinheit, Functional Unit (FU)

Teilmenge des *OSI-TP*-Protokolls, die eine bestimmte Funktionalität beinhaltet. Das OSI-TP-Protokoll ist in folgende Funktionseinheiten aufgeteilt:

- Dialogue
- Shared Control
- Polarized Control
- Handshake
- Commit
- Chained Transactions
- Unchained Transactions
- Recovery

Ein Hersteller, der OSI-TP implementiert, muss nicht alle Funktionseinheiten realisieren, sondern kann sich auf eine Teilmenge beschränken. Eine Kommunikation zwischen Anwendungen zweier unterschiedlicher OSI-TP-Implementierungen ist nur dann möglich, wenn die realisierten Funktionseinheiten zueinander passen.

Inbound-Conversation (CPI-C)

siehe *Incoming-Conversation.*

Incoming-Conversation (CPI-C)

Eine *Conversation*, bei der das lokale CPI-C-Programm *Akzeptor* ist, heißt Incoming-Conversation. In der X/Open-Specification wird für Incoming-Conversation auch das Synonym Inbound-Conversation verwendet.

Initiator (CPI-C)

Die Kommunikationspartner einer *Conversation* werden Initiator und *Akzeptor* genannt. Der Initiator baut die Conversation mit den CPI-C-Aufrufen Initialize_Conversation und Allocate auf.

Insert

Feld in einem Meldungstext, in das openUTM aktuelle Werte einträgt.

KDCDEF

UTM-Tool für die Generierung von UTM-Anwendungen. KDCDEF erstellt anhand der Konfigurationsinformationen in den KDCDEF-Steueranweisungen die UTM-Objekte *KDCFILE* und die ROOT-Tabellen-Source für die Main Routine *KDCROOT.*
In UTM-Cluster-Anwendungen erstellt KDCDEF zusätzlich die Cluster-Konfigurationsdatei, die Cluster-User-Datei, den Cluster-Pagepool, die Cluster-GSSB-Datei und die Cluster-ULS-Datei.

KDCFILE

Eine oder mehrere Dateien, die für den Ablauf einer *UTM-Anwendung* notwendige Daten enthalten. Die KDCFILE wird mit dem UTM-Generierungstool *KDCDEF* erstellt. Die KDCFILE enthält unter anderem die *Konfiguration* der Anwendung.

KDCROOT

Main Routine eines *Anwendungsprogramms*, die das Bindeglied zwischen *Teilprogrammen* und UTM-Systemcode bildet. KDCROOT wird zusammen mit den *Teilprogrammen* zum *Anwendungsprogramm* gebunden.

KDCS-Programmschnittstelle

Universelle UTM-Programmschnittstelle, die den nationalen Standard DIN 66 265 erfüllt und Erweiterungen enthält. Mit KDCS (Kompatible Datenkommunikationsschnittstelle) lassen sich z.B. Dialog-Services erstellen und *Message Queuing* Funktionen nutzen. Außerdem stellt KDCS Aufrufe zur *verteilten Verarbeitung* zur Verfügung.

Logische Verbindung

Zuordnung zweier Kommunikationspartner.

LPAP-Bündel

LPAP-Bündel ermöglichen die Verteilung von Nachrichten an LPAP-Partner auf mehrere Partner-Anwendungen. Soll eine UTM-Anwendung sehr viele Nachrichten mit einer Partner-Anwendung austauschen, kann es für die Lastverteilung sinnvoll sein, mehrere Instanzen der Partner-Anwendung zu starten und die Nachrichten auf die einzelnen Instanzen zu verteilen. In einem LPAP-Bündel übernimmt *open*UTM die Verteilung der Nachrichten an die Instanzen der Partner-Anwendung. Ein LPAP-Bündel besteht aus einem Master-LPAP und mehreren Slave-LPAPs. Die Slave-LPAPs werden dem Master-LPAP bei der UTM-Generierung zugeordnet. LPAP-Bündel gibt es sowohl für das OSI TP-Protokoll als auch für das LU6.1-Protokoll.

Netzwerk-Selektor

Der Netzwerk-Selektor identifiziert im lokalen System einen *Dienstzugriffspunkt* zur Vermittlungsschicht des *OSI-Referenzmodells*.

Object Identifier

Ein Object Identifier ist ein weltweit eindeutiger Bezeichner für Objekte im OSI-Umfeld. Ein Object Identifier besteht aus einer Folge von ganzen Zahlen, die einen Pfad in einer Baumstruktur repräsentiert.

OpenCPIC

Trägersystem für UTM-Clients, die das *OSI TP* Protokoll verwenden.

OpenCPIC-Client

OSI TP Partner-Anwendungen mit Trägersystem *OpenCPIC*.

openUTM-Anwendung

siehe *UTM-Anwendung*.

openUTM-Cluster

aus der Sicht von UPIC-Clients, **nicht** aus Server-Sicht: Zusammenfassung mehrerer Knoten-Anwendungen einer UTM-Cluster-Anwendung zu einer logischen Anwendung, die über einen gemeinsamen Symbolic Destination Name adressiert wird.

openUTM-D

openUTM-D (openUTM-Distributed) ist eine openUTM-Komponente, die *verteilte Verarbeitung* ermöglicht. openUTM-D ist integraler Bestandteil von openUTM.

OSI-LPAP-Bündel

LPAP-Bündel für *OSI TP*-Partner-Anwendungen.

OSI-LPAP-Partner

OSI-LPAP-Partner sind die bei openUTM generierten Adressen der *OSI TP-Partner*. Für die *verteilte Verarbeitung* über das Protokoll *OSI TP* muss in der lokalen Anwendung für jede Partner-Anwendung ein OSI-LPAP-Partner konfiguriert werden. Der OSI-LPAP-Partner spiegelt in der lokalen Anwendung die Partner-Anwendung wider. Bei der Kommunikation wird die Partner-Anwendung nicht über ihren Anwendungsnamen oder ihre Adresse, sondern über den Namen des zugeordneten OSI-LPAP-Partners angesprochen.

OSI-Referenzmodell

Das OSI-Referenzmodell stellt einen Rahmen für die Standardisierung der Kommunikation von offenen Systemen dar. ISO, die Internationale Organisation für Standardisierung, hat dieses Modell im internationalen Standard ISO IS7498 beschrieben. Das OSI-Referenzmodell unterteilt die für die Kommunikation von Systemen notwendigen Funktionen in sieben logische Schichten. Diese Schichten haben jeweils klar definierte Schnittstellen zu den benachbarten Schichten.

OSI TP

Von der ISO definiertes Kommunikationsprotokoll für die verteilte Transaktionsverarbeitung. OSI TP steht für Open System Interconnection Transaction Processing.

OSI TP-Partner

Partner der UTM-Anwendung, der mit der UTM-Anwendung über das OSI TP-Protokoll kommuniziert.
Beispiele für solche Partner sind:

– eine UTM-Anwendung, die über OSI TP kommuniziert
– eine Anwendung im IBM-Umfeld (z.B. CICS), die über openUTM-LU62 angeschlossen ist
– eine Anwendung des Trägersystems OpenCPIC des openUTM-Client
– Anwendungen anderer TP-Monitore, die OSI TP unterstützen

Outbound-Conversation (CPI-C)

siehe *Outgoing-Conversation*.

Outgoing-Conversation (CPI-C)

Eine Conversation, bei der das lokale CPI-C-Programm der *Initiator* ist, heißt Outgoing-Conversation. In der X/Open-Specification wird für Outgoing-Conversation auch das Synonym Outbound-Conversation verwendet.

Presentation-Selektor

Der Presentation-Selektor identifiziert im lokalen System einen *Dienstzugriffspunkt* zur Darstellungsschicht des *OSI-Referenzmodells*.

Resource Manager

Resource Manager (RMs) verwalten Datenressourcen. Ein Beispiel für RMs sind Datenbank-Systeme. openUTM stellt aber auch selbst Resource Manager zur Verfügung, z.B. für den Zugriff auf *Message Queues*, lokale Speicherbereiche und Logging-Dateien. Anwendungsprogramme greifen auf RMs über RM-spezifische Schnittstellen zu. Für Datenbank-Systeme ist dies meist SQL, für die openUTM-RMs die Schnittstelle KDCS.

RFC1006

Von IETF (Internet Engineering Task Force) definiertes Protokoll der TCP/IP-Familie zur Realisierung der ISO-Transportdienste (Transportklasse 0) auf TCP/IP-Basis.

Selektor

Ein Selektor identifiziert im lokalen System einen *Zugriffspunkt* auf die Dienste einer Schicht des *OSI-Referenzmodells*. Jeder Selektor ist Bestandteil der Adresse des Zugriffspunktes.

Semaphor (Unix-, Linux- und Windows-Systeme)

Betriebsmittel auf Unix-, Linux- und Windows-Systemen, das zur Steuerung und Synchronisation von Prozessen dient.

Server

Ein Server ist eine *Anwendung*, die *Services* zur Verfügung stellt. Oft bezeichnet man auch den Rechner, auf dem Anwendungen laufen, als Server.

Server-Seite einer Conversation (CPI-C)

Begriff ersetzt durch *Akzeptor*.

Server-Server-Kommunikation

siehe *verteilte Verarbeitung*.

Service Access Point

siehe *Dienstzugriffspunkt.*

Service

Services bearbeiten die *Aufträge*, die an eine Server-Anwendung geschickt werden. Ein Service in einer UTM-Anwendung wird auch *Vorgang* genannt und setzt sich aus einer oder mehreren *Transaktionen* zusammen. Ein Service wird über den *Vorgangs-TAC* aufgerufen. Services können von *Clients* oder anderen Services angefordert werden.

Service Routine

siehe *Teilprogramm*.

Session-Selektor

Der Session-Selektor identifiziert im lokalen System einen *Zugriffspunkt* zu den Diensten der Kommunikationssteuerschicht (Session-Layer) des *OSI-Referenzmodells*.

Sicherungspunkt

Ende einer *Transaktion*. Zu diesem Zeitpunkt werden alle in der Transaktion vorgenommenen Änderungen der *Anwendungsinformation* gegen Systemausfall gesichert und für andere sichtbar gemacht. Während der Transaktion gesetzte Sperren werden wieder aufgehoben.

TAC

siehe *Transaktionscode*.

Teilprogramm

UTM-*Services* werden durch ein oder mehrere Teilprogramme realisiert. Die Teilprogramme sind Bestandteile des *Anwendungsprogramms*. Abhängig vom verwendeten API müssen sie KDCS-, XATMI- oder CPIC-Aufrufe enthalten. Sie sind über *Transaktionscodes* ansprechbar. Einem Teilprogramm können mehrere Transaktionscodes zugeordnet werden.

TNS (Unix-, Linux- und Windows-Systeme)

Abkürzung für den Transport Name Service, der einem Anwendungsnamen einen Transport-Selektor und das Transportsystem zuordnet, über das die Anwendung erreichbar ist.

Transaktion

Verarbeitungsabschnitt innerhalb eines *Services*, für den die Einhaltung der *ACID-Eigenschaften* garantiert wird. Von den in einer Transaktion beabsichtigten Änderungen der *Anwendungsinformation* werden entweder alle konsistent durchgeführt oder es wird keine durchgeführt (Alles-oder-Nichts Regel). Das Transaktionsende bildet einen *Sicherungspunkt*.

Transaktionscode/TAC

Name, über den ein *Teilprogramm* aufgerufen werden kann. Der Transaktionscode wird dem Teilprogramm bei der *statischen* oder *dynamischen Konfiguration* zugeordnet. Einem Teilprogramm können auch mehrere Transaktionscodes zugeordnet werden.

Transaktionsrate

Anzahl der erfolgreich beendeten *Transaktionen* pro Zeiteinheit.

Transfer-Syntax

Bei *OSI TP* werden die Daten zur Übertragung zwischen zwei Rechnersystemen von der lokalen Darstellung in die Transfer-Syntax umgewandelt. Die Transfer-Syntax beschreibt die Daten in einem neutralen Format, das von allen beteiligten Partnern verstanden wird. Jeder Transfer-Syntax muss ein *Object Identifier* zugeordnet sein.

Transport-Selektor

Der Transport-Selektor identifiziert im lokalen System einen *Dienstzugriffspunkt* zur Transportschicht des *OSI-Referenzmodells*.

Transportverbindung

Im *OSI-Referenzmodell* eine Verbindung zwischen zwei Instanzen der Schicht 4 (Transportschicht).

Typisierter Puffer (XATMI)

Puffer für den Austausch von typisierten und strukturierten Daten zwischen Kommunikationspartnern. Durch diese typisierten Puffer ist die Struktur der ausgetauschten Daten den Partnern implizit bekannt.

UPIC

Trägersystem für UTM-Clients. UPIC steht für Universal Programming Interface for Communication.

UPIC-Client
Bezeichnung für UTM-Clients mit Trägersystem UPIC.

UTM-Anwendung
Eine UTM-Anwendung stellt *Services* zur Verfügung, die Aufträge von *Clients* oder anderen Anwendungen bearbeiten. openUTM übernimmt dabei u.a. die Transaktionssicherung und das Management der Kommunikations- und Systemressourcen. Technisch gesehen ist eine UTM-Anwendung eine Prozessgruppe, die zur Laufzeit eine logische Server-Einheit bildet.

UTM-Generierung
Statische Konfiguration einer *UTM-Anwendung* mit dem UTM-Tool *KDCDEF* und Erzeugen des Anwendungsprogramms.

Verteilte Transaktion
Transaktion, die sich über mehr als eine Anwendung erstreckt und in mehreren (Teil)-Transaktionen in verteilten Systemen ausgeführt wird.

Verteilte Transaktionsverarbeitung
Verteilte Verarbeitung mit *verteilten Transaktionen*.

Verteilte Verarbeitung
Bearbeitung von *Dialog-Aufträgen* durch mehrere Anwendungen oder Übermittlung von *Hintergrundaufträgen* an eine andere Anwendung. Für die verteilte Verarbeitung werden die höheren Kommunikationsprotokolle *LU6.1* und *OSI TP* verwendet. Über openUTM-LU62 ist verteilte Verarbeitung auch mit LU6.2 Partnern möglich. Man unterscheidet verteilte Verarbeitung mit *verteilten Transaktionen* (Anwendungs-übergreifende Transaktionssicherung) und verteilte Verarbeitung ohne verteilte Transaktionen (nur lokale Transaktionssicherung). Die verteilte Verarbeitung wird auch Server-Server-Kommunikation genannt.

Vorgang (KDCS)
Ein Vorgang dient zur Bearbeitung eines *Auftrags* in einer *UTM-Anwendung*. Er setzt sich aus einer oder mehreren *Transaktionen* zusammen. Die erste Transaktion wird über den *Vorgangs-TAC* aufgerufen. Es gibt *Dialog-Vorgänge* und *Asynchron-Vorgänge*. openUTM stellt den Teilprogrammen eines Vorgangs gemeinsame Datenbereiche zur Verfügung. Anstelle des Begriffs Vorgang wird häufig auch der allgemeinere Begriff *Service* gebraucht.

Vorgangs-TAC (KDCS)
Transaktionscode, mit dem ein *Vorgang* gestartet wird.

Vorgangs-Wiederanlauf (KDCS)

Wird ein Vorgang unterbrochen, z.B. infolge Abmeldens des Terminal-Benutzers oder Beendigung der *UTM-Anwendung*, führt openUTM einen Vorgangs-Wiederanlauf durch. Ein *Asynchron-Vorgang* wird neu gestartet oder beim zuletzt erreichten *Sicherungspunkt* fortgesetzt, ein *Dialog-Vorgang* wird beim zuletzt erreichten Sicherungspunkt fortgesetzt. Für den Terminal-Benutzer wird der Vorgangs-Wiederanlauf eines Dialog-Vorgangs als *Bildschirm-Wiederanlauf* sichtbar, sofern am letzten Sicherungspunkt eine Dialog-Nachricht an den Terminal-Benutzer gesendet wurde.

Wiederanlauf

siehe *Vorgangs-Wiederanlauf.*

XATMI

XATMI (X/Open Application Transaction Manager Interface) ist eine von X/Open standardisierte Programmschnittstelle für die Programm-Programm-Kommunikation in offenen Netzen.
Das in openUTM implementierte XATMI genügt der XATMI CAE Specification von X/Open. Die Schnittstelle steht in COBOL und C zur Verfügung. XATMI in openUTM kann über die Protokolle *OSI TP*, *LU6.1* und *UPIC* kommunizieren.

Zugriffspunkt

siehe *Dienstzugriffspunkt.*

Literatur

Dokumentation zu openUTM

[1] **openUTM**
Konzepte und Funktionen
Benutzerhandbuch

[2] **openUTM**
Anwendungen programmieren mit KDCS für COBOL, C und C++
Basishandbuch

[3] **openUTM**
Anwendungen generieren
Benutzerhandbuch

[4] **openUTM**
Einsatz von openUTM-Anwendungen auf BS2000-Systemen
Benutzerhandbuch

[5] **openUTM**
Einsatz von openUTM-Anwendungen auf Unix-, Linux- und Windows-Systemen
Benutzerhandbuch

[6] **openUTM**
Anwendungen administrieren
Benutzerhandbuch

[7] **openUTM**
Meldungen, Test und Diagnose auf BS2000-Systemen
Benutzerhandbuch

[8] **openUTM**
Meldungen, Test und Diagnose auf Unix-, Linux- und Windows-Systemen
Benutzerhandbuch

[9] **openUTM**
Anwendungen erstellen mit X/Open-Schnittstellen
Benutzerhandbuch

[10] **openUTM**
XML für openUTM

[11] **openUTM-Client** (Unix-Systeme)
für Trägersystem OpenCPIC
Client-Server-Kommunikation mit openUTM
Benutzerhandbuch

[12] **openUTM-Client**
für Trägersystem UPIC
Client-Server-Kommunikation mit openUTM
Benutzerhandbuch

[13] **openUTM WinAdmin**
Grafischer Administrationsarbeitsplatz für openUTM
Beschreibung und Online-Hilfe

[14] **openUTM WebAdmin**
Web-Oberfläche zur Administration von openUTM
Beschreibung und Online-Hilfe

[15] **openUTM**, **openUTM-LU62**
Verteilte Transaktionsverarbeitung
zwischen openUTM und CICS-, IMS- und LU6.2-Anwendungen
Benutzerhandbuch

[16] **openUTM** (BS2000)
Anwendungen programmieren mit KDCS für Assembler
Ergänzung zum Basishandbuch

[17] **openUTM** (BS2000)
Anwendungen programmieren mit KDCS für Fortran
Ergänzung zum Basishandbuch

[18] **openUTM** (BS2000)
Anwendungen programmieren mit KDCS für Pascal-XT
Ergänzung zum Basishandbuch

[19] **openUTM** (BS2000)
Anwendungen programmieren mit KDCS für PL/I
Ergänzung zum Basishandbuch

[20] **WS4UTM** (Unix- und Windows-Systeme**)**
Web-Services für openUTM

[21] **openUTM**
Masterindex

Dokumentation zum Umfeld von Unix-, Linux- und Windows-Systemen

[22] **CMX** V6.0 (Unix-Systeme)
Betrieb und Administration
Benutzerhandbuch

[23] **CMX** V6.0
CMX-Anwendungen programmieren
Programmierhandbuch

X/Open Veröffentlichungen

[24] X/Open CAE Specification
Distributed Transaction Processing:
The CPI-C Specification, Version 2
ISBN: 1-85912-135-7

[25] X/Open CAE Specification
ACSE/Presentation:
Transaction Processing API (XAP-TP)
ISBN: 1-85912-091-1

[26] X/Open CAE Specification:
Distributed Transaction Processing:
The TX (Transaction Demarcation) Specification
ISBN:1-85912-094-6

[27] X/Open CAE Specification
Distributed Transaction Processing:
The XA Specification
ISBN: 1-872630-24-3

[28] X/Open CAE Specification
Distributed Transaction Processing
The XATMI Specification
ISBN: 1-85912-130-6

ISO Veröffentlichungen

[29] ISO 8824
Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)

[30] ISO 8825
Information Processing Systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)

[31] ISO 9804
Information Technology - Open Systems Interconnection - Service Definition for the Commitment, Concurrency and Recovery service element

[32] ISO/IEC 10026
Information Technology - Open Systems Interconnection - Distributed Transaction Processing
Part 1: 1992, OSI TP Model
Part 2: 1992, OSI TP Service
Part 3: 1992, Protocol Specification
Part 4: 1995, Protocol Implementation Conformance Statement (PICS) proforma
Part 5: DIS 1993, Application context proforma and guidelines when using OSI TP
Part 6: 1994, Unstructured Data Transfer (UDT)

Weitere Literatur

[33] **NetView/PC Application Programming, Version 1.2**
IBM, SC31-6004

[34] **T.61**
CCITT Recommendation T.61: 1988
Character Repertoire and Coded Character Sets for the International Teletx Service

Stichwörter

Z