openUTM V5.3, openUTM-LU62 V5.1

Distributed Transaction Processing between openUTM and CICS, IMS and LU6.2 Applications

Edition July 2008

Comments… Suggestions… Corrections…

The User Documentation Department would like to know your opinion on this manual. Your feedback helps us to optimize our documentation to suit your individual needs.

Feel free to send us your comments by e-mail to:
manuals@ts.fujtsu.com

Certified documentation according to DIN EN ISO 9001:2000

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2000.

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

Copyright and Trademarks

Contents

1 Preface 11

1.1 Summary of contents and target group 13

1.2 Summary of contents of the openUTM manuals 14
1.2.1 openUTM documentation 14
1.2.2 Documentation for the openSEAS product environment 18
1.2.3 README files 19

1.3 Changes to the previous manual 20

1.4 Notational conventions 21

2 Interconnection to IBM systems 23

2.1 Direct LU6.2 interconnection via TRANSIT on Solaris 25

2.2 Direct LU6.2 interconnection via SNAP-IX on Solaris 26

2.3 Direct LU6.2 interconnection via IBM Communications Server on Linux or AIX systems 27

2.4 Direct LU6.2 interconnection via IBM Communications Server on Windows systems 28

2.5 LU6.2 interconnection via gateway computer with TRANSIT on Solaris 29

2.6 LU6.2 interconnection via gateway computer with SNAP-IX on Solaris 30

2.7 LU6.2 interconnection via gateway computer with IBM Communications Server on Linux or AIX systems 31

2.8 LU6.2 interconnection via gateway computer with IBM Communications Server on Windows 32

2.9 Direct LU6.1 interconnection with openUTM via TRANSIT on Solaris 33

2.10 LU6.1 interconnection via gateway computer with TRANSIT on Solaris 34

3 LU6.2 interconnections with openUTM-LU62 . . . 35

3.1 openUTM-LU62 concepts and functions . . . 35
3.1.1 Substitute concept . . . 35
3.1.2 The openUTM-LU62 architecture . . . 37
3.1.3 openUTM-LU62 components . . . 39
3.1.4 Recovery functions . . . 40
3.1.5 Limitations of the protocol mapping . . . 41

3.2 Generating openUTM-LU62 . . . 42
3.2.1 Generation file format . . . 42
3.2.2 INSTANCE statement . . . 43
3.2.3 Starting the generation program . . . 52
3.2.4 Restoring generation files . . . 53
3.2.5 Displaying the name of the generation file used . . . 53

3.3 Administering openUTM-LU62 . . . 54
3.3.1 Administration under UNIX systems and Windows systems . . . 54
3.3.2 Starting openUTM-LU62 . . . 55
3.3.3 Terminating openUTM-LU62 . . . 58
3.3.4 Displaying status information . . . 59
3.3.5 Establishing connections . . . 62
3.3.6 Clearing connections . . . 63
3.3.7 Activating and deactivating traces . . . 64
3.3.8 Analyzing traces . . . 66
3.3.9 Creating a dump . . . 67
3.3.10 Protocol trace . . . 67

4 openUTM-CICS interconnection via LU6.2 . . . 77

4.1 Generating an openUTM-CICS interconnection . . . 77
4.1.1 Definitions in CICS . . . 77
DEFINE CONNECTION . . . 78
DEFINE SESSIONS . . . 80
4.1.2 VTAM generation . . . 82
4.1.3 TRANSIT generation for openUTM-CICS sessions . . . 82
4.1.4 SNAP-IX generation for openUTM-CICS sessions . . . 82
4.1.5 Generating the IBM Communications Server for openUTM-CICS sessions . . . 82
4.1.6 openUTM-LU62 generation . . . 83
4.1.7 TNSX generation . . . 83
4.1.8 openUTM generation . . . 83
4.1.9 Defining CICS transactions . . . 84

4.1.10 Using user IDs . . . 85
4.1.11 A complete sample generation . . . 86

4.2 Programming an openUTM-CICS interconnection . . . 110
4.2.1 CICS commands for CICS job-submitter services . . . 111
ALLOCATE . . . 111
CONNECT PROCESS . . . 112
SEND . . . 113
RECEIVE . . . 114
CONVERSE . . . 114
ISSUE ABEND . . . 115
ISSUE CONFIRMATION . . . 115
ISSUE ERROR . . . 115
SYNCPOINT . . . 116
SYNCPOINT ROLLBACK . . . 116
ISSUE PREPARE . . . 116
FREE . . . 117
WAIT CONVID . . . 117
ISSUE SIGNAL . . . 117
4.2.2 CICS commands for CICS job-receiving services . . . 118
EXTRACT PROCESS . . . 118
RECEIVE . . . 118
SEND . . . 118
CONVERSE . . . 119
ISSUE ABEND . . . 119
ISSUE CONFIRMATION . . . 119
ISSUE ERROR . . . 119
SYNCPOINT . . . 120
SYNCPOINT ROLLBACK . . . 120
FREE . . . 120
WAIT CONVID . . . 120
ISSUE SIGNAL . . . 120
ISSUE PREPARE . . . 120
RETURN . . . 121
4.2.3 CICS programming hints . . . 122
4.2.4 Comparison with KDCS calls . . . 123
4.2.5 Examples of openUTM-CICS communication . . . 125
Starting an openUTM dialog service from within a CICS application program . . . 125
Starting an openUTM asynchronous service from a CICS application program . . . 133
Starting a CICS dialog service from a UTM application program . . . 135
Starting a CICS asynchronous service from a UTM application program . . . 146
4.2.6 Distributed Program Link . . . 148
4.2.7 openUTM programming hints . . . 152

4.3 Using the CPI-C program interface . . . 153
4.3.1 Comparison to KDCS calls . . . 153
4.3.2 Examples of openUTM-CPIC communication . . . 156
Starting an openUTM dialog service from a CPIC application program . . . 156
Starting an openUTM asynchronous service from a CPIC application program . . . 163
Starting a CPIC dialog service from a UTM application program . . . 164
Starting a CPIC asynchronous service from a UTM application program . . . 178

5 openUTM-IMS interconnection via LU6.2 . . . 181

5.1 Generating an openUTM-IMS interconnection . . . 181
5.1.1 IMS startup parameters . . . 181
5.1.2 Defining the LU name of IMS . . . 182
5.1.3 Defining IMS transactions . . . 183
5.1.4 Defining partner LUs and openUTM transactions . . . 184
5.1.5 VTAM generation . . . 186
5.1.6 LU6.2 security . . . 187
5.1.7 Full generation example . . . 188

5.2 Programming an openUTM-IMS interconnection . . . 198
5.2.1 DL/I program interface . . . 198
5.2.2 CPI-C program interface . . . 201
5.2.3 LU6.2 Edit Exit routine . . . 201
5.2.4 Use of format names . . . 201
5.2.5 Examples of execution sequences with DL/I programs . . . 202
5.2.6 Examples of execution sequences with CPI-C programs . . . 208
5.2.7 Examples of execution sequences with standard IMS transactions . . . 209
5.2.8 Examples of execution sequences with IMS as the job submitter . . . 211

5.3 IMS administration . . . 213

6 openUTM-CICS interconnection via LU6.1 . . . 215

6.1 CICS definitions for openUTM-CICS sessions . . . 215
DEFINE CONNECTION . . . 216
DEFINE SESSIONS . . . 217
Example of dependencies in CICS and openUTM generation . . . 219

6.2 TRANSIT generation for openUTM-CICS sessions . . . 220

6.3 Defining CICS transactions . . . 222
6.3.1 Local transactions . . . 222
6.3.2 Remote transactions . . . 223

6.4 CICS programming during interconnection with openUTM . . . 224
6.4.1 Rules and restrictions for CICS programming . . . 224
6.5 CICS commands for CICS job-submitting conversations . . . 226
ALLOCATE . . . 226
BUILD ATTACH . . . 227
SEND . . . 228
RECEIVE . . . 229
CONVERSE . . . 230
EXTRACT ATTACH . . . 231
SYNCPOINT command . . . 232
RETURN command . . . 232
6.6 CICS commands for CICS job-receiving conversations . . . 233
RECEIVE . . . 233
EXTRACT ATTACH . . . 233
BUILD ATTACH . . . 234
SEND . . . 234
CONVERSE . . . 235
SYNCPOINT . . . 235
RETURN . . . 235
6.7 Comparison with KDCS calls . . . 236
6.8 Programming examples of CICS-openUTM communication . . . 237
6.9 CICS commands for queued jobs . . . 238
START command . . . 238
RETRIEVE . . . 239
Examples of the exchange of queued jobs . . . 239
6.10 Notes regarding openUTM-CICS programming . . . 240

7 openUTM-IMS interconnection via LU6.1 . . . 243

7.1 IMS generation for openUTM-IMS sessions . . . 243
COMM macro . . . 244
TYPE macro . . . 244
TERMINAL macro . . . 245
VTAMPOOL macro . . . 246
SUBPOOL macro . . . 246
NAME macro . . . 247
7.1.1 Examples of dependencies in openUTM/IMS generation . . . 248
7.2 TRANSIT generation for openUTM-IMS sessions . . . 249

7.3 Generating IMS transactions . . . 249
TRANSACT macro . . . 249

7.4 IMS programming for links to openUTM . . . 250
7.4.1 IMS interconnection options . . . 250
7.4.2 IMS calls . . . 251
7.4.3 Comparison to KDCS calls . . . 251
7.4.4 Examples of openUTM-IMS communication . . . 252
7.4.5 Examples of the exchange of queued jobs . . . 253
7.4.6 Notes on openUTM-IMS programming . . . 254

8 LU6.1 dummy dialogs between asynchronous services . . . 255

8.1 Program interfaces for openUTM . . . 257
INFO GN . . . 258
APRO IN . . . 259

8.2 Passing FMH6 parameters in openUTM, IMS and CICS . . . 261
8.2.1 Transaction code handling in openUTM . . . 261
8.2.2 Transaction code handling in IMS . . . 262
8.2.3 Transaction code handling in CICS . . . 262

9 Error diagnosis . . . 263

9.1 Diagnostic aids . . . 263

9.2 LU6.1 sense data . . . 265

10 openUTM-LU62 messages . . . 269

10.1 Messages from the u62_tp program . . . 270

10.2 Messages from the XAP-TP provider . . . 319
10.2.1 General inserts for the XAP-TP messages . . . 337

10.3 Messages from the utilities . . . 346
10.3.1 Messages from u62_start . . . 346
10.3.2 Messages from u62_sta . . . 352
10.3.3 Messages from u62_adm . . . 354
10.3.4 Messages from u62_gen . . . 358

Glossary 361

Abbreviations 373

Related publications 377

Index 395

1 Preface

Modern enterprise-wide IT environments are subjected to many challenges of an increasingly explosive nature. This is the result of:

- heterogeneous system landscapes
- different hardware platforms
- different networks and different types of network access (TCP/IP, SNA, HTTP)
- the applications used by companies

Consequently, problems arise – whether as a result of mergers, joint ventures or labor-saving measures. Companies are demanding flexible, scalable applications, as well as transaction processing capability for processes and data, while business processes are becoming more and more complex. The growth of globalization means, of course, that applications are expected to run 24 hours a day, seven days a week, and must offer high availability in order to enable Internet access to existing applications across time zones.

openUTM, which is a classical application server, offers a runtime environment that meets all these requirements of modern, business-critical applications, because openUTM combines all the standards and advantages of transaction monitor middleware and message queuing systems:

- consistency of data and processing
- high availability of the applications (not just the hardware)
- high throughput even when there are large numbers of users (i.e. highly scalable)
- flexibility as regards changes to and adaptation of the IT system

openUTM is part of the comprehensive **openSEAS** (Open Suite for Enterprise Application Servers) offering. The mature technology of openUTM is used by innovative **openSEAS** products:

- BeanConnect is a family of adapters that conforms to the Java Connector Architecture (JCA) of Sun and supports standardized connection of openUTM applications to J2EE application servers, in particular to Oracle AS 10g. Oracle AS 10g is the application server that, as a part of openSEAS, is available as a development and runtime platform to all business applications, portals and Web pages in grid computing scale.
- In conjunction with, for example, openUTM, Web*Transactions* supports the development of modern e-business applications. Existing openUTM applications can be transferred unchanged to the World Wide Web using Web*Transactions* and can be integrated into portals. With the business object builder Biz*Transactions*, Web*Transactions* contains the openSEAS tool for generating re-usable business objects. The business objects are generated from existing applications (e.g. BS2000/OSD and OS/390 applications, transaction applications with openUTM and SAP R/3) and, depending on requirements, can be made available as JavaBeans, .Net components or Web services.

1.1 Summary of contents and target group

This manual, “Distributed Transaction Processing between openUTM and CICS, IMS and LU6.2 Applications“, describes how to link UTM applications in BS2000/OSD, UNIX systems and Windows systems to CICS, IMS and other LU6.2 applications in IBM systems when using distributed transaction processing. The manual is intended for system administrators, network administrators and application programmers.

Chapter 2 contains general information on interconnection between openUTM and IBM systems.

Chapters 3 to 5 describe LU6.2 interconnection between openUTM and IBM systems. Chapter 3 describes the openUTM-LU62 gateway, including the administration of openUTM-LU62. Chapters 4 and 5 examine the link to CICS and IMS.

Chapters 6 through 8 describe LU6.1 interconnection between openUTM and CICS or IMS. Chapter 6 describes the link to CICS, and chapter 7 describes LU6.1 interconnection to IMS. Special enhancements to the openUTM program interface used for openUTM - IMS interconnection are described in chapter 8.

Chapter 9 contains information for diagnosing errors, and chapter 10 contains a list of all openUTM-LU62 messages.

The detailed appendices at the end of the manual, i.e. the sections “Glossary”, “Abbreviations”, “Related publications” and “Index”, should make working with this manual easier.

1.2 Summary of contents of the openUTM manuals

This section provides an overview of the manuals in the openUTM suite and of the various related products.

1.2.1 openUTM documentation

The openUTM documentation consists of manuals, an online help system for openUTM WinAdmin, which is the graphical administration workstation, and a release note for each platform on which openUTM is released.

Some manuals are valid for all platforms, and others apply specifically to BS2000/OSD, UNIX systems or Windows systems.

All the manuals are available as PDF files on the internet at
`http://manuals.ts.fujitsu.com`

Most of them can also be ordered in book form. In addition, the PDF files are included on the openUTM product DVD.

The following sections provide a task-oriented overview of the openUTM V5.3 documentation. You will find a complete list of documentation for openUTM in the chapter on related publications at the back of the manual on page 377.

Introduction and overview

The **Concepts and Functions** manual gives a coherent overview of the essential functions, features and areas of application of openUTM. It contains all the information required to plan a UTM operation and to design an openUTM application. The manual explains what openUTM is, how it is used, and how it is integrated in the BS2000/OSD, UNIX based and Windows based platforms.

Programming

- You will require the **Programming Applications with KDCS for COBOL, C and C++** manual to create server applications via the KDCS interface. This manual describes the KDCS interface as used for COBOL, C and C++. This interface provides the basic functions of the universal transaction monitor, as well as the calls for distributed processing. The manual also describes interaction with databases.
- You will require the **Creating Applications with X/Open Interfaces** manual if you want to use the X/Open interfaces. This manual contains descriptions of the UTM-specific extensions to the X/Open program interfaces TX, CPI-C and XATMI as well as notes on configuring and operating UTM applications which use X/Open interfaces. In addition, you will require the X/Open-CAE specification for the corresponding X/Open interface.
- If you want to interchange data on the basis of XML, you will need the document entitled **Data Marshalling with XML for openUTM**. This describes the C and COBOL calls required to work with XML documents. This description is only available as a PDF document (online and on CD-ROM).
- For BS2000/OSD there is supplementary documentation on the programming languages Assembler, Fortran, Pascal-XT and PL/1. This is only available in the form of PDF files (online and on the WinAdmin CD-ROM).

Configuration

The **Generating Applications** manual is available to you for defining configurations. This describes how to use the UTM tool KDCDEF to define the configuration and create the KDCFILE for a UTM application. In addition, it also shows you how to transfer important administration and user data to a new KDCFILE using the KDCUPD tool. You do this, for example, when moving to a new openUTM version or after changes have been made to the configuration.

Linking, starting and using UTM applications

In order to be able to use UTM applications, you will need the **Using openUTM Applications** manual for the relevant operating system (BS2000/OSD or UNIX systems/Windows systems). This describes how to link and start a UTM application program, how to sign on and off to and from a UTM application and how to replace application programs dynamically and in a structured manner. It also contains the UTM commands that are available to the terminal user.

Administering applications and changing configurations dynamically

- The **Administering Applications** manual describes the program interface for administration and the UTM administration commands. It provides information on how to create your own administration programs and on the facilities for administering several different applications centrally. It also describes how to administer message queues and printers using the KDCS calls DADM and PADM.
- If you are using **openUTM WinAdmin**, the graphical administration workstation, the following documentation is available to you:
  - A **description of WinAdmin**, which provides a comprehensive overview of the functional scope and handling of WinAdmin. This document is shipped with the software and is also available online as a PDF file.
  - The **online help system**, which provides context-sensitive help information on all dialog boxes and associated parameters offered by the graphical user interface. In addition, it also tells you how to configure WinAdmin in order to administer and generate openUTM applications.

Testing and diagnosing errors

You will also require the **Messages, Debugging and Diagnostics** manuals (there are separate manuals for UNIX systems / Windows systems and for BS2000/OSD) to carry out the tasks mentioned above. These manuals describe how to debug a UTM application, the contents and evaluation of a UTM dump, the behavior in the event of an error, and the openUTM message system, and also lists all messages and return codes output by openUTM.

Creating openUTM clients

The following manuals are available to you if you want to create client applications for communication with UTM applications:

- The **openUTM-Client for the UPIC Carrier System** describes the creation and operation of client applications based on UPIC. In addition to the description of the CPI-C and XATMI interfaces, you will find information on how you can use the C++ classes or ActiveX to create programs quickly and easily.
- The **openUTM-Client for the OpenCPIC Carrier System** manual describes how to install and configure OpenCPIC and configure an OpenCPIC application. It describes how to install OpenCPIC and how to configure an OpenCPIC application. It indicates what needs to be taken into account when programming a CPI-C application and what restrictions apply compared with the X/Open CPI-C interface.

- The documentation for the **JUpic-Java classes** shipped with **BeanConnect** is supplied with the software. This documentation consists of Word and PDF files that describe its introduction and installation and of Java documentation with a description of the Java classes.
- If you want to provide UTM services on the Web quickly and easily then you need the manual **Web-Services for openUTM**. The manual describes how to use the software product WS4UTM (WebServices for openUTM) to make the services of UTM applications available as Web services. This description is only available as a PDF document (online and on the openUTM product DVD). The use of the graphical user interface is described in the corresponding **online help system**.

Communicating with the IBM world

The present manual **Distributed Transaction Processing between openUTM and CICS, IMS and LU6.2 Applications** describes how to communicate with IBM transaction systems from your UTM application. This describes the CICS commands, IMS macros and UTM calls that are required to link UTM applications to CICS and IMS applications. The link capabilities are described using detailed configuration and generation examples. The manual also describes communication via openUTM-LU62 as well as its installation, generation and administration.

1.2.2 Documentation for the openSEAS product environment

The connection between openUTM and the openSEAS product environment is outlined in the openUTM manual **Concepts and Functions**. The following sections show which openSEAS documentation is relevant to openUTM.

Integrating J2EE application servers and UTM applications

The BeanConnect adapter family forms part of the openSEAS product suite (open Suite for Enterprise Application Servers). The BeanConnect adapters implement the connection between conventional transaction monitors and modern application servers and thus permit the efficient integration of legacy applications in modern Java applications.

- The manual **BeanConnect for openUTM** describes the product BeanConnect for openUTM. BeanConnect for openUTM provides a JCA 1.5-compliant adapter which connects UTM applications with applications based on J2EE, e.g. the Oracle application server Oracle AS 10g.
  The manuals for Oracle AS 10g can be obtained from Oracle.
- The manual **BeanConnect for CICS** describes the product BeanConnect for CICS. BeanConnect for CICS provides a JCA 1.5-compliant adapter which connects CICS applications with applications based on J2EE, e.g. the Oracle application server Oracle.

Connecting to the web and application integration

You will require the following new and existing manuals to connect UTM applications to the web using the Web*Transactions* product:

- The introductory manual **Web*Transactions* - Concepts and Functions** provides an overview of the capabilities and possible applications of Web*Transactions,* describing its features and how it works. The manual explains the object concept and the dynamic operation of a Web*Transactions* session.
- All language resources in the WTML template language are explained in the manual **Web*Transactions* - Template Language**. This manual contains numerous examples that illustrate the language resources and make it easier for you to use these resources.
- The manuals **Web*Transactions* - Web Access to openUTM Applications via UPIC** and **Web*Transactions* - Web Access to OSD Applications** describe what you have to do to connect UTM dialog applications to the web via the UPIC or terminal interface. The steps you have to take are illustrated by means of a concrete example.

- If you want to integrate openUTM applications and services in Microsoft applications, you can use the Biz*Transactions* component shipped with Web*Transactions* together with the associated manual, **Biz*Transactions* - Application Integration with Business Objects**. In addition to installation and configuration, this also describes the concepts and components of Biz*Transactions*. Administration and the development of integrated client applications is also explained.

  Biz*Transactions* also includes an online help system.

The manuals listed above will also be supplemented by JavaDocs.

1.2.3 README files

Information on any functional changes and additions to the current product version described in this manual can be found in the product-specific README files.

- BS2000/OSD:

  On a BS2000 computer, you will find information in the Release Note (file name SYSFGM.UTM.053.*language* and possibly in a README file as well (file name SYSRME.UTM.053.*language*). Please ask your systems support for the user ID on which the README file is located. You can view the README file with the /SHOW-FILE command or in an editor or you can print it to a standard printer with the following command:

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

- UNIX systems:

  The README file and any other files, such as a manual supplement file, can be found in the *utmpath* under `/docs/`*language*.

- Windows systems:

  The README file and any other files, such as a manual supplement file, can be found in the *utmpath* under *\Docs\language*.

1.3 Changes to the previous manual

The most important changes compared to the previous manual are:

- Extension of the openUTM-LU62 generation parameters: the parameters, which up to now were only available in TNSX, have been included. Thus, it is no longer necessary to use TNSX.
- New starting option, which provides a middle course between cold start and warm start: `u62_start -k`
- New option in displaying status information. With this option you can display the status of the Enterprise Extender connections: `u62_sta -c`

1.4 Notational conventions

This manual uses the following notational conventions:

UPPERCASE LETTERS
: The names of files, calls, statements, commands and operands appear in uppercase.

lowercase letters
: Placeholders for operand values appear in lowercase.

[ ]
: Square brackets are used to enclose optional specifications that may be omitted.

{ }
: Braces are used to enclose alternative specifications, from which you must select a value.

<u>underscore</u>
: The underscore is used to identify the default value.

`typewriter font`
: Typewriter font is used to indicate input which must be entered precisely in this form or output which will appear precisely in this form.

italic
: italic font is used to indicate variables.

2 Interconnection to IBM systems

Distributed transaction processing is usually implemented in IBM systems using the LU6.2 SNA protocol.

The following IBM transaction monitors and communication products support this protocol:

- CICS (or CICS Transaction Server)
- IMS (as of version 6)
- TXSeries on System p5 (formerly RS/6000)
- CPI-C application programs on IBM System i5 (formerly AS/400).

Older IBM transaction monitors, such as IMS up to version 5, can only be reached via LU6.1 on a transaction-oriented basis.

openUTM can use the LU6.1, OSI-TP and LU6.2 protocols for distributed transaction processing.

Only LU6.2 should be used when new interconnections are created to the IBM world, provided the partner application supports it.

For an interconnection between openUTM and an LU6.2 partner, the product openUTM-LU62 and suitable basic SNA software for the operating system are required. openUTM-LU62 is available on Solaris, Windows, Linux and AIX operating systems. The following software configurations are available:

- openUTM-LU62 on Solaris with the products TRANSIT-SERVER and TRANSIT-CPIC
- openUTM-LU62 on Solaris with SNAP-IX
- openUTM-LU62 on Windows with IBM Communications Server for Windows
- openUTM-LU62 on Linux with IBM Communications Server for Linux
- openUTM-LU62 on AIX with IBM Communications Server for AIX

In this manual the names of IBM products such as IBM COmmunications Server for Linux are abbreviated to IBM Communications Server.

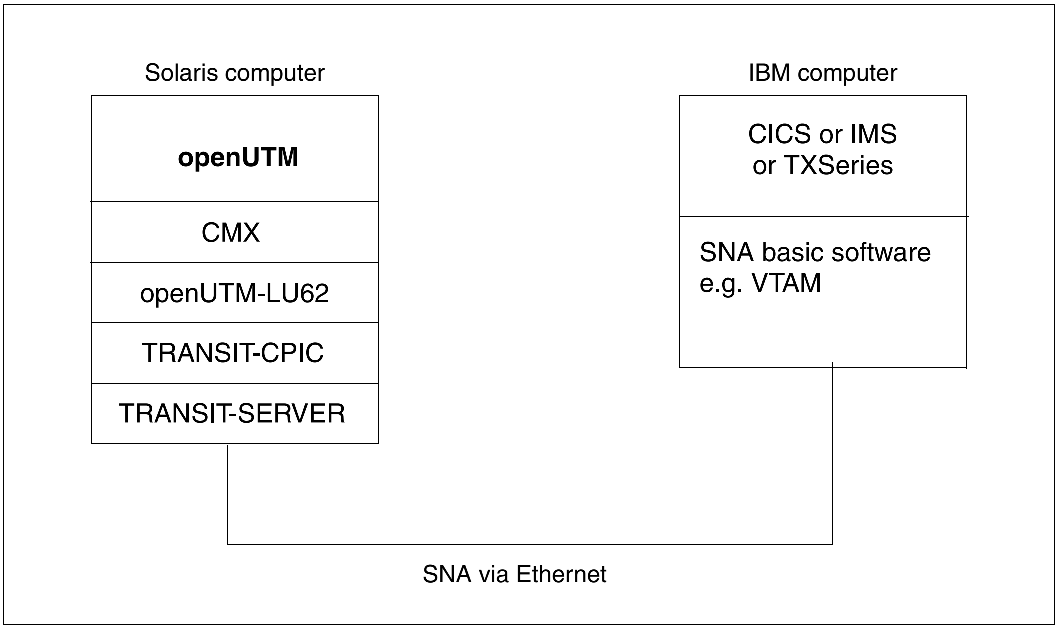
Typical configurations for interconnection between openUTM and an IBM product are listed in the following. The products of different vendors are mentioned.

– IBM Communications Server is an IBM product that provides the functions of the basic SNA software.
– SNAP-IX is a Data Connection Limited product that provides the functions of the basic SNA software.
– TRANSIT-SERVER, TRANSIT-CLIENT and TRANSIT-CPIC are Fujitsu Siemens Computers products that provide the functions of the basic SNA software.
– CMX, PCMX and BCAM are Fujitsu Siemens Computers products.
– CICS, IMS, TxSeries and VTAM are IBM products.

As a general rule, the names of the IBM products may differ slightly in some versions.

2.1 Direct LU6.2 interconnection via TRANSIT on Solaris

The following interconnection option can be used when the UTM application is running on a Solaris computer connected directly to the SNA network.



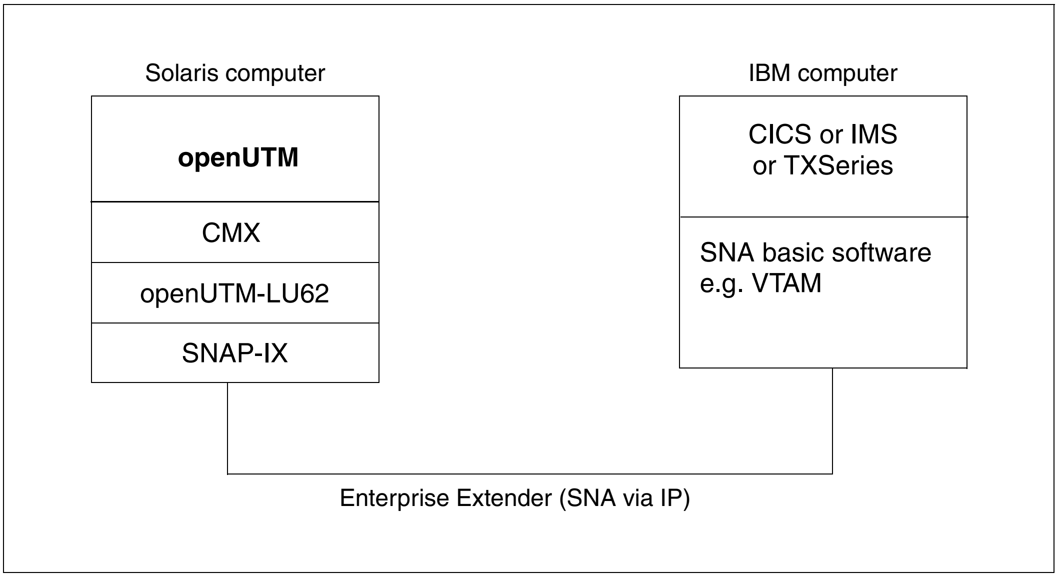
Direct LU6.2 interconnection with openUTM on Solaris and use of TRANSIT

The Solaris computer must be generated as a type 2.1 PU in the SNA network.

The diagram shows SNA interconnection via Ethernet. See the TRANSIT-SERVER manual for other methods of interconnection.

2.2 Direct LU6.2 interconnection via SNAP-IX on Solaris

The following interconnection option can be used when the UTM application is running on a Solaris computer connected directly to the SNA network.

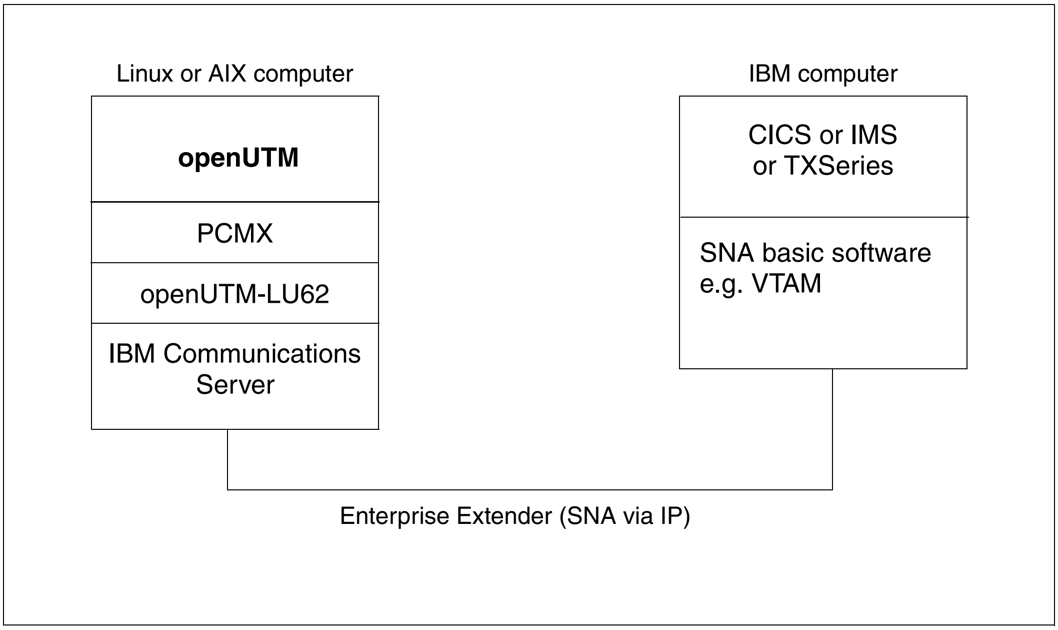


Direct LU6.2 interconnection with openUTM on Solaris and use of SNAP-IX

The figure above shows an SNA interconnection via Enterprise Extender (SNA via IP). See the SNAP-IX documentation for information on other possible interconnections.

2.3 Direct LU6.2 interconnection via IBM Communications Server on Linux or AIX systems

The following configuration option can be used when the UTM application is running on a Linux or AIX computer connected directly to the SNA network.

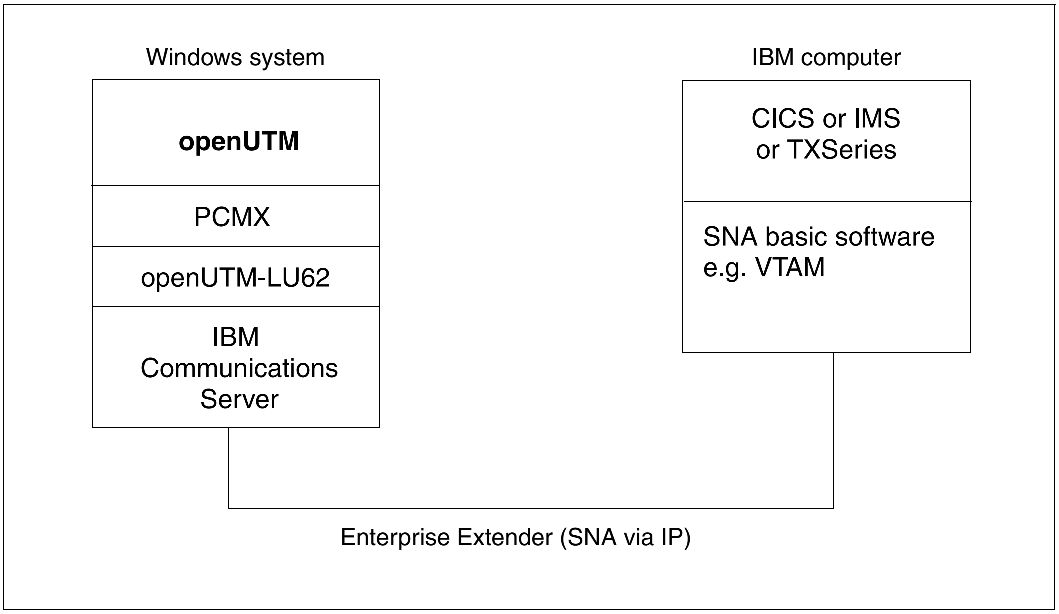

Linux or AIX computer
openUTM
PCMX
openUTM-LU62
IBM Communications Server
IBM computer
CICS or IMS or TXSeries
SNA basic software e.g. VTAM
Enterprise Extender (SNA via IP)

Direct LU6.2 interconnection with openUTM on Linux or AIX systems and use of IBM Communications Server

The figure above shows an SNA interconnection via Enterprise Extender (SNA via IP). See the IBM Communications Server documentation for information on other possible interconnections.

2.4 Direct LU6.2 interconnection via IBM Communications Server on Windows systems

The following interconnection option can be used if the UTM application is running on a Windows system computer connected directly to the SNA network.

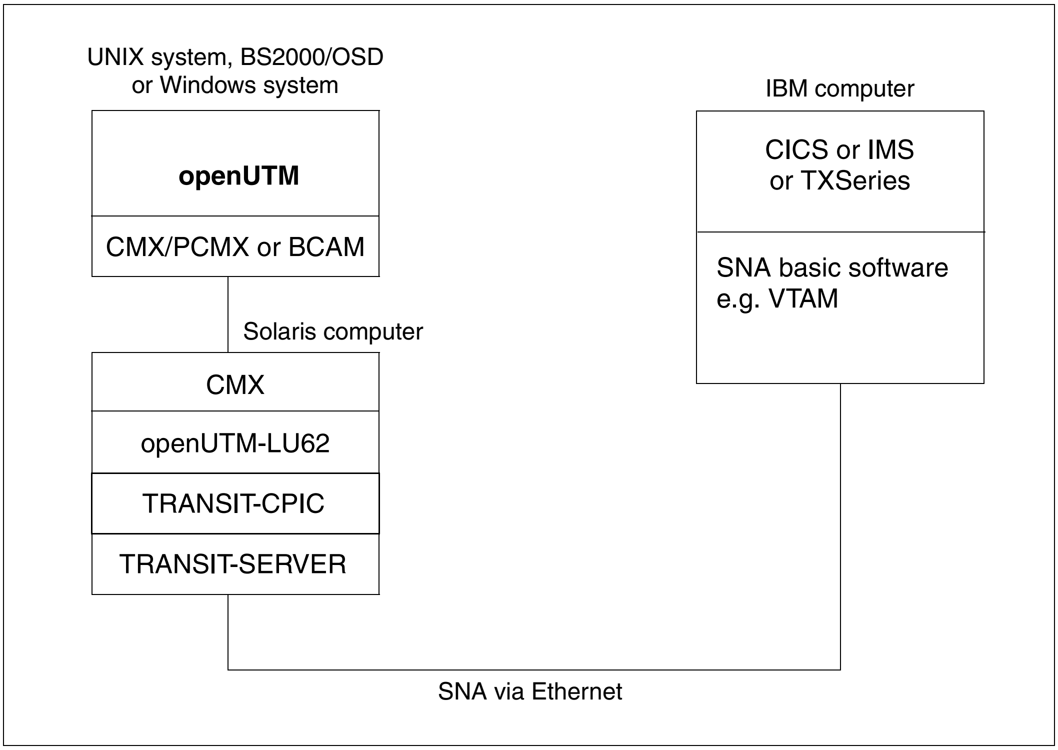

Windows system
openUTM
PCMX
openUTM-LU62
IBM Communications Server
IBM computer
CICS or IMS or TXSeries
SNA basic software e.g. VTAM
Enterprise Extender (SNA via IP)

Direct LU6.2 interconnection with openUTM on Windows systems and use of IBM Communications Server

The diagram shows SNA interconnection via Enterprise Extender (SNA via IP). See the documentation on the IBM Communications Server for other methods of interconnection.

2.5 LU6.2 interconnection via gateway computer with TRANSIT on Solaris

The following interconnections via LU6.2 can always be utilized when the computer on which the UTM application runs is not or should not be directly connected to the SNA network. Any operating system can be used on this computer, for example BS2000/OSD, a Solaris or Linux system, a Windows system or a different UNIX system. A Solaris system is required as a gateway computer.



LU6.2 via gateway computer with TRANSIT on Solaris

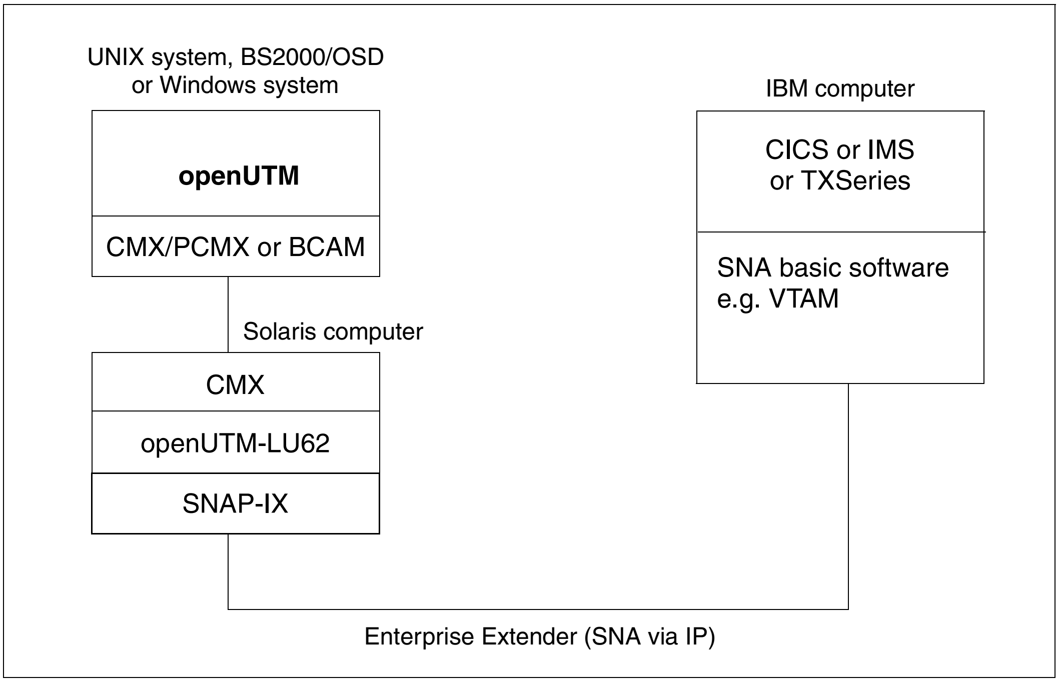
In the case of openUTM under BS2000/OSD, the product OSS(BS2000/OSD) is also required.

The connection between the two computers on the left can be established via TCP/IP with RFC1006.
The Solaris computer at the bottom must be generated as a type 2.1 PU in the SNA network.

The diagram shows SNA interconnection via Ethernet. See the TRANSIT-SERVER manual for other methods of interconnection.

2.6 LU6.2 interconnection via gateway computer with SNAP-IX on Solaris

The following interconnection option via LU6.2 can be used whenever the computer on which the UTM application is running is not to be or cannot be connected directly to the SNA network. It does not matter which operating system is running on this computer. It can be a BS2000/OSD system, a Solaris or Linux system, a Windows system or a different UNIX system. A Solaris system is required as the gateway computer.


UNIX system, BS2000/OSD or Windows system
openUTM
CMX/PCMX or BCAM
Solaris computer
CMX
openUTM-LU62
SNAP-IX
IBM computer
CICS or IMS or TXSeries
SNA basic software e.g. VTAM
Enterprise Extender (SNA via IP)

LU6.2 interconnection via Gateway computer with SNAP-IX on Solaris

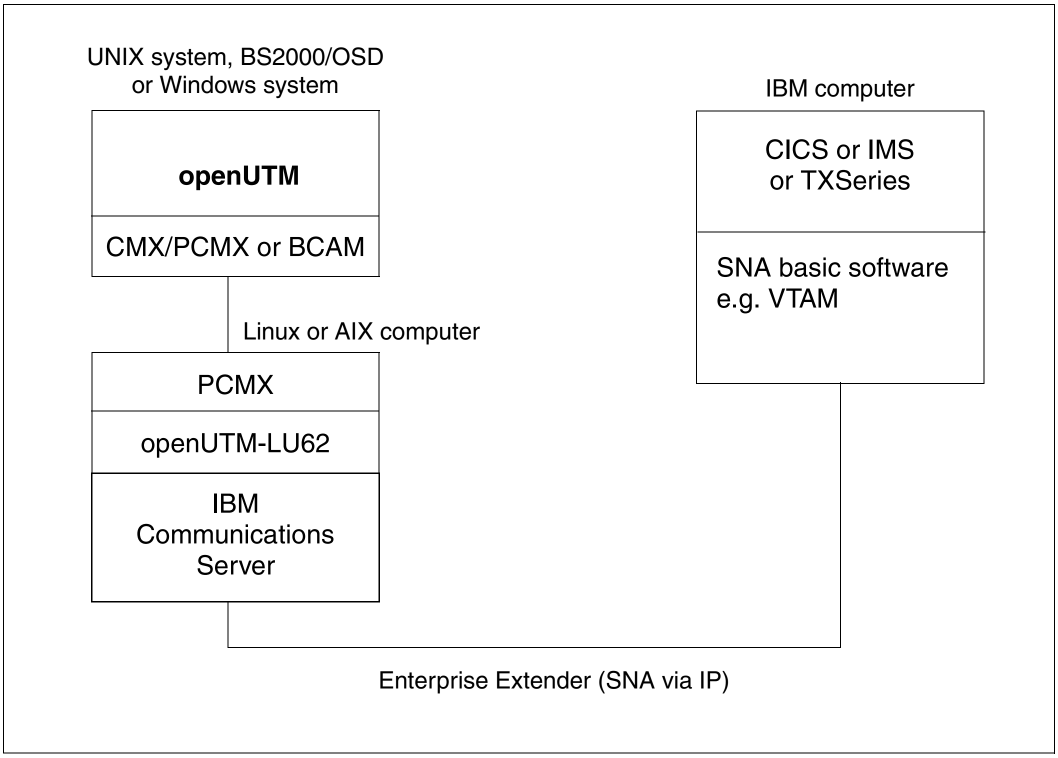
In the case of openUTM under BS2000/OSD, the product OSS(BS2000/OSD) is also required.

The connection between the two computers on the left is implemented via TCP/IP with RFC1006.

The figure shows an SNA interconnection via Enterprise Extender (SNA via IP). See the SNAP-IX documentation for information on other possible interconnections.

2.7 LU6.2 interconnection via gateway computer with IBM Communications Server on Linux or AIX systems

The following interconnection option via LU6.2 can be used whenever the computer on which the UTM application is running is not to be or cannot be connected directly to the SNA network. It does not matter which operating system is running on this computer. It can be a BS2000/OSD system, a Solaris or Linux system, a Windows system or a different UNIX system. A Linux or AIX system is required as the gateway computer.


UNIX system, BS2000/OSD or Windows system
openUTM
CMX/PCMX or BCAM
Linux or AIX computer
PCMX
openUTM-LU62
IBM Communications Server
IBM computer
CICS or IMS or TXSeries
SNA basic software e.g. VTAM
Enterprise Extender (SNA via IP)

LU6.2 interconnection via gateway computer with IBM Communications Server on Linux or AIX systems

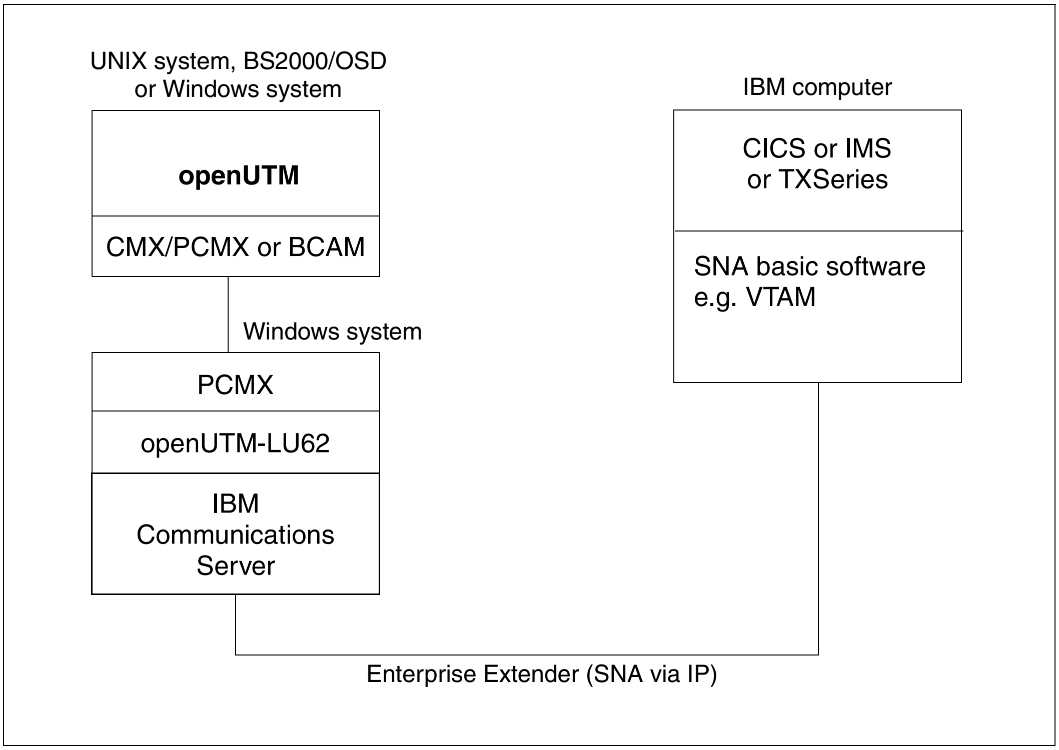
In the case of openUTM under BS2000/OSD, the product OSS(BS2000/OSD) is also required.

The connection between the two computers on the left is implemented via TCP/IP with RFC1006.

The figure shows an SNA interconnection via Enterprise Extender (SNA via IP). See the IBM Communications Server documentation for information on other possible interconnections.

2.8 LU6.2 interconnection via gateway computer with IBM Communications Server on Windows

The following LU6.2 interconnection option can be used if the computer running the UTM application cannot or should not be connected directly to the SNA network. The operating system on the computer is immaterial, e.g. it can be a BS2000/OSD system, a UNIX system or a WIndows system. A Windows system is required as the gateway computer.


UNIX system, BS2000/OSD or Windows system
openUTM
CMX/PCMX or BCAM
Windows system
PCMX
openUTM-LU62
IBM Communications Server
IBM computer
CICS or IMS or TXSeries
SNA basic software e.g. VTAM
Enterprise Extender (SNA via IP)

LU6.2 interconnection via a gateway computer with IBM Communications Server on Windows systems

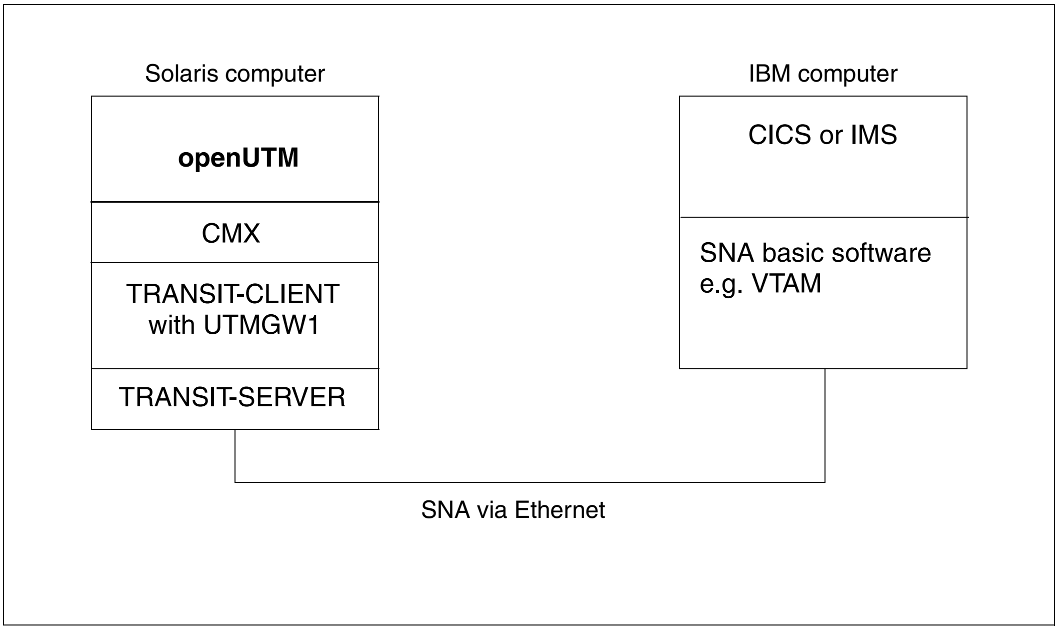
In the case of openUTM under BS2000/OSD, the product OSS(BS2000/OSD) is also required.

The connection between the two computers on the left is implemented via TCP/IP with RFC1006.

The diagram shows SNA interconnection via Enterprise Extender (SNA via IP). See the documentation on the IBM Communications Server for other methods of interconnection.

2.9 Direct LU6.1 interconnection with openUTM via TRANSIT on Solaris

The following interconnection option can be used if the UTM is running on a Solaris computer connected directly to an SNA network.

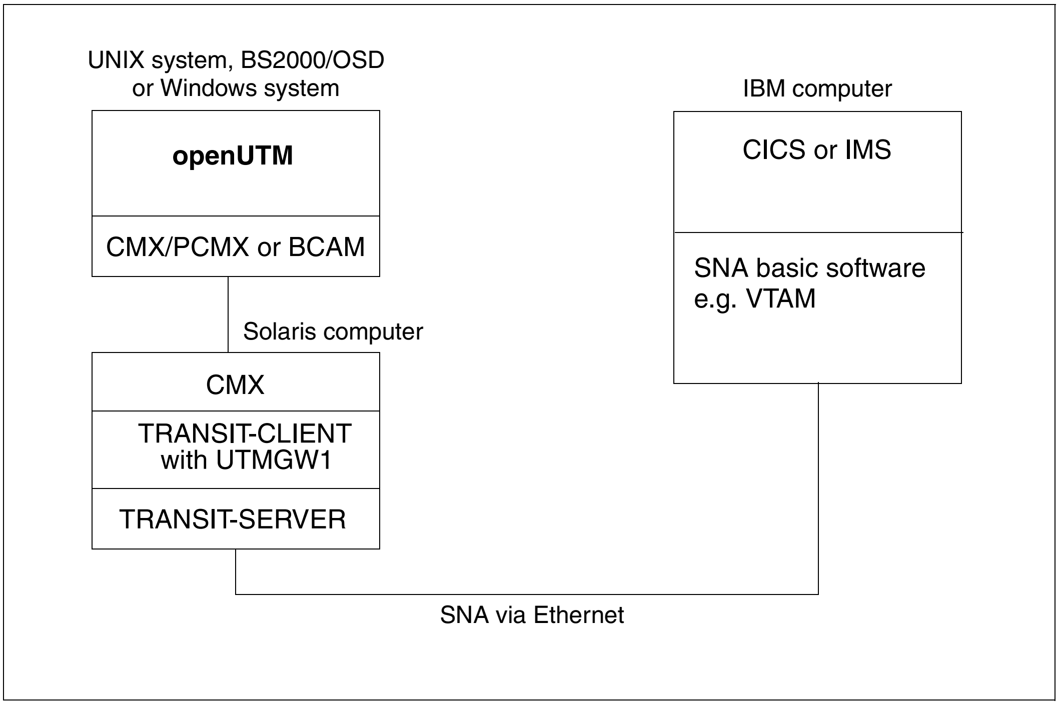


LU6.1 interconnection via TRANSIT with openUTM on Solaris

The figure shows an SNA interconnection via Ethernet. See the TRANSIT-SERVER manual for information on other possible interconnections.

2.10 LU6.1 interconnection via gateway computer with TRANSIT on Solaris

The following possible interconnections via LU6.1 can always be used when the computer on which the UTM application runs is not or should not be directly connected to the SNA network. Any operating system can be used on this computer, for example a BS2000/OSD system, a Solaris or Linux system, a Windows system or another UNIX system. A Solaris system is required as the gateway computer.


UNIX system, BS2000/OSD or Windows system
openUTM
CMX/PCMX or BCAM
Solaris computer
CMX
TRANSIT-CLIENT with UTMGW1
TRANSIT-SERVER
IBM computer
CICS or IMS
SNA basic software e.g. VTAM
SNA via Ethernet

LU6.1 interconnection via a gateway computer with TRANSIT on Solaris

The connection between the two computers on the left can be established using TCP/IP using the RFC1006.

The diagram shows SNA interconnection via Ethernet. See the TRANSIT-SERVER manual for other methods of interconnection.

3 LU6.2 interconnections with openUTM-LU62

The openUTM-LU62 product allows communication between UTM applications and LU6.2 applications, i.e. applications that use the LU6.2 protocol. Examples of such LU6.2 applications in the IBM environment are CICS, IMS starting with version 5, TXSeries and CPIC applications. The LU6.2 protocol allows communication with or without transaction management.

First, the general openUTM-LU62 concept and the use of this concept will be described in the following. Then the details to consider when utilizing openUTM-CICS interconnection via LU6.2 will be presented in the chapter “openUTM-CICS interconnection via LU6.2” on page 77.

3.1 openUTM-LU62 concepts and functions

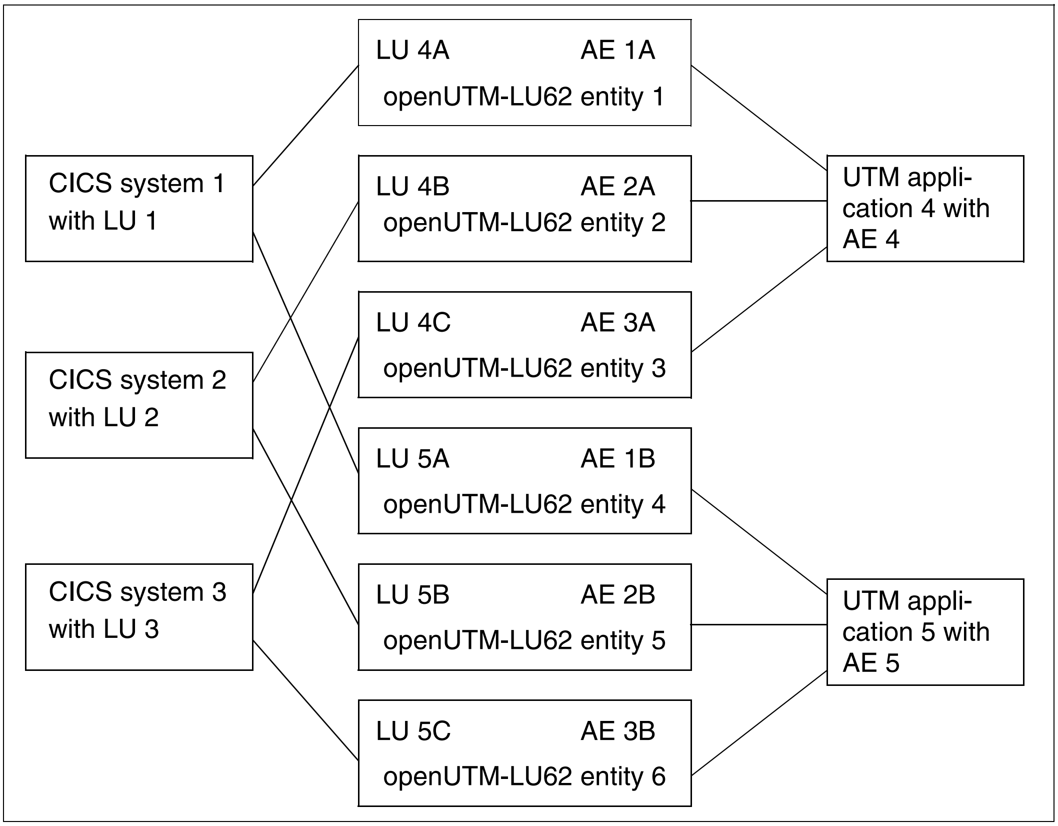
3.1.1 Substitute concept

openUTM-LU62 responds to a UTM application just like it does an application linked via OSI-TP. openUTM-LU62 implements a substitute OSI-TP application for every LU6.2 application that can be reached from a UTM application, meaning for every CICS, for example.

From the point of view of the LU6.2 application, openUTM-LU62 is a remote LU6.2 partner. openUTM-LU62 implements a substitute LU (Logical Unit) for every UTM application that can be reached from a LU6.2 application.

One openUTM-LU62 entity is required for each OSI-CON statement in a UTM application. You will find more information on the OSI-CON statement in the openUTM manual “Generating Applications”.

If should to be able to reach 3 different CICS systems from 2 UTM applications, then 6 openUTM-LU62 entities are required (see the section “INSTANCE statement” on page 43). The entire system is illustrated in the following diagram:



Interconnection of several applications to several openUTM-LU62 entities

Each UTM application uses an application entity (AE) as an access point for OSI-TP communication. This point is also designated as an access point in a UTM generation. The 3 CICS systems use one logical unit (LU) each as the access point for the LU6.2 communication.

openUTM-LU62 uses application entities as access points for OSI-TP communication and LUs as access points for LU6.2 communication. Exactly one application entity and exactly one LU is implemented in an openUTM-LU62 entity.

Entity 1 of openUTM-LU62 implements the LU 1 - AE 4 connection. Entity 2 of openUTM-LU62 implements the LU 2 - AE 4 connection.

If, for example, UTM application 4 wants to establish a connection to CICS system 1, then it must establish a connection to AE 1A instead. If CICS system 3 wants to establish a connection to UTM application 5, then it must address LU 5C instead.

In this manner, LUs 4A, 4B, 4C are the substitutes for UTM application 4, i.e. for AE 4, in the SNA network. AEs 1A and 1B are the substitutes for CICS system 1, i.e. for LU 1, in the ISO network.

3.1.2 The openUTM-LU62 architecture

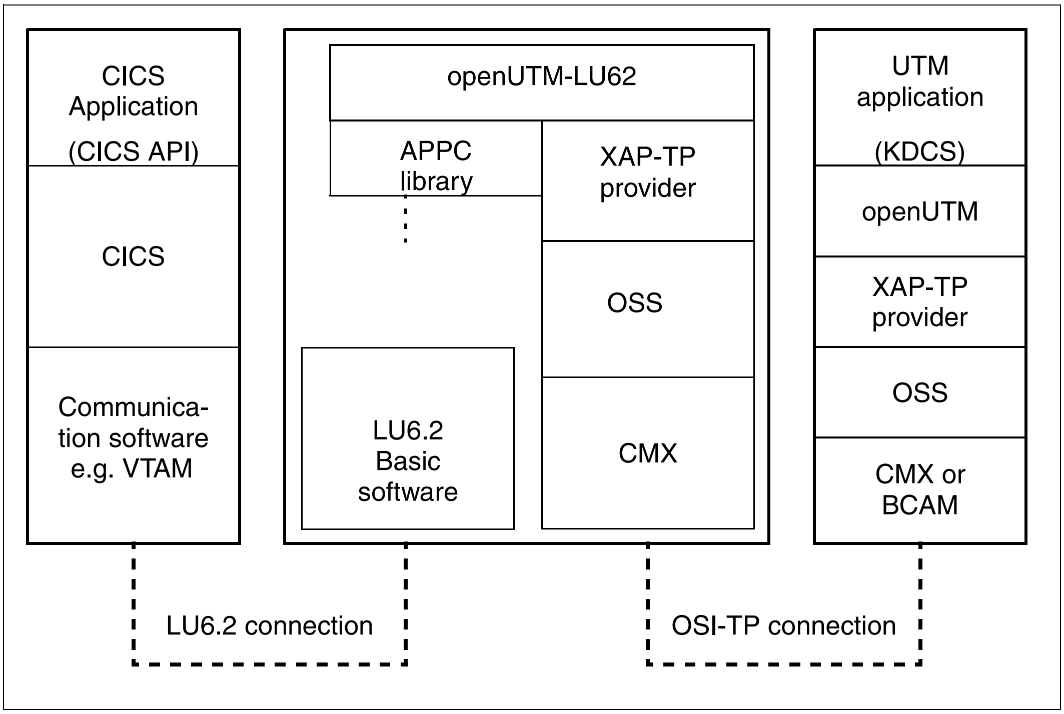
openUTM-LU62 uses the XAP-TP system program interface standardized by X/Open for OSI-TP communication. This program interface is also used in the openUTM and openUTM-Client (OpenCPIC carrier system) products. The XAP-TP provider is based on the OSS component. OSS is in turn based on CMX on some operation systems, and on PCMX on other systems. The XAP-TP provider and OSS are included in the product openUTM-LU62

openUTM-LU62 uses the APPC program interface for LU6.2 communication. This program interface is implemented by means of the basic LU6.2 software. The following can be used as the basic LU6.2 software:

- TRANSIT-SERVER and TRANSIT-CPIC on Solaris
- SNAP-IX on Solaris
- IBM Communications Server on Windows, Linux or AIX.

openUTM-LU62 not only converts LU6.2 protocols to OSI-TP protocols and vice-versa, but also represents its own node in the transaction diagram, and therefore also writes log records and independently executes a transaction recovery on the OSI-TP side and on the LU6.2 side when a connection or a computer breaks down.

This architecture is shown in the following figure:



openUTM-LU62 architecture

The boxes outlined in bold represent the borders of the computers. The lines within a box designate the borders between the various software components, and the name of the program interface is sometimes specified in parentheses on the bottom line.

As mentioned in section “Direct LU6.2 interconnection via SNAP-IX on Solaris” on page 26, the computers in the middle and on the right may be identical for openUTM under UNIX or Windows systems. In this case, the local loop from CMX via the IP address 127.0.0.1 is used for communication between openUTM and openUTM-LU62.

3.1.3 openUTM-LU62 components

openUTM-LU62 consists of the following components:

- The `u62_start` program starts the openUTM-LU62 entities and the write log processes. It monitors the openUTM-LU62 processes after that.
- The `u62_tp` program to map the protocol between OSI-TP and LU6.2. It is started by `u62_start` once for each openUTM-LU62 entity.
- The `u62_wlog` write log program writes the log records to a file. It is started once for each openUTM-LU62 entity as a write log daemon when transaction logging is used.
- There is a generation utility (`u62_gen`), and there are also administration utilities (`u62_adm`, `u62_sta`).
- On Windows systems, the `u62_svc` program is also available for automatically starting openUTM-LU62 as a service on system startup.

All programs except `u62_sta` can be found in the directory `/opt/lib/utmlu62` on UNIX systems. By default, they can be called by any user. It is possible to limit administration rights to specific users of the UNIX system (see the section “Administration under UNIX systems and Windows systems” on page 54).

The program `u62_sta` can be found in `/opt/bin` on UNIX systems and can be called by all UNIX users.

On Windows systems, the administration programs are located in the directory `Programs\utmlu62`, where `Programs\utmlu62` is the default installation directory of openUTM-LU62.

On UNIX systems, interprocess communication between `u62_tp` and `u62_wlog`, `u62_adm` and `u62_sta` is implemented using named pipes and shared memory. On Windows systems, `u62_tp` and `u62_wlog` run as threads in the same process (`u62_tp.exe`). Interprocess communication between this process and `u62_adm` or `u62_sta` is implemented using sockets and shared memory.

3.1.4 Recovery functions

When the UTM and LU6.2 application programs communication is secured through transaction logging, then openUTM-LU62 ensures a transaction recovery that guarantees the consistency of the corresponding databases for update transactions in case a connection is lost or a computer crashes.

If the connection between openUTM-LU62 and the LU6.2 application or the UTM application is lost while the transaction is terminating, then openUTM-LU62 ensures the proper termination of the transaction as soon as the partner application can be reached again.

openUTM-LU62 writes log records onto the hard drive in critical situations while the transaction is ending. If the transaction has completely terminated, then these log records are deleted. If the computer crashes or openUTM-LU62 terminates abnormally while the transaction is terminating, then the log records are not deleted. The log records are read during the next start of openUTM-LU62 and the open transactions are executed to completion with the participation of the two applications.

On UNIX systems, the log records are stored in the synclog file `/opt/lib/utmlu62/sync.log.`*loc_lu_alias* for each openUTM-LU62 entity. The file suffix *loc_lu_alias* is replaced by the current value of the corresponding openUTM-LU62 entity. This file corresponds for the most part to the KDCFILE of a UTM application. On Windows systems, the file `sync.log.`*loc_lu_alias* is located in the directory `Programs\utmlu62`, where *loc_lu_alias* is the value of the parameter of the IINSTANCE statement (see the section “INSTANCE statement” on page 43.

openUTM-LU62 also offers cold start capabilities. The existing log records are deleted during a cold start. In general, a cold start should only be used in special cases. A warm start should be carried out in any case after a computer crash or the abnormal termination of openUTM-LU62.

If, however, the LU6.2 application was restarted with a cold start between the time of the computer crash and the openUTM-LU62 warm start, then the corresponding openUTM-LU62 entity must also be cold started. In this case the open transactions are not executed to completion.

The command “CEMT SET CONNECTION NOTPENDING” can be entered for CICS for a connection while the system is running. This command deletes the log records stored in CICS for the corresponding connection. If this command is sent to connect to the UTM application, then it has the same effect as a cold start.

If only CICS or only the openUTM-LU62 entity is cold started and the other system is warm started due to a user error, then this is noted when the connection between openUTM-LU62 and CICS is established. In this case, openUTM-LU62 outputs a message and cancels the establishment of the connection to CICS. You can query this status on the CICS side using “CEMT INQUIRE CONNECTION” (“Xno” display).

If an openUTM-LU62 entity has been warm started and recovery information is available, but CICS does not have any recovery information anymore due to a cold start or “CEMT SET CONNECTION NOTPENDING“, then you must also start the corresponding openUTM-LU62 entity. If the openUTM-LU62 entity has been cold started and CICS rejects the establishment of transaction-oriented connections due to an exchange log name error, then you must execute the command “CEMT SET CONNECTION NOTPENDING” in CICS.

The cold/warm start coordination between openUTM-LU62 and the UTM application cannot be checked by openUTM-LU62 because there is no method in the OSI-TP protocol to do this. However, the cold and warm starts between openUTM-LU62 and the UTM application must be coordinated just as carefully.

3.1.5 Limitations of the protocol mapping

openUTM-LU62 maps the LU6.2 and OSI-TP protocols to each other. Since the two protocols are not exactly identical with respect to the scope of their functions, openUTM-LU62 can only map those elements common to both protocols. Details are described in the sections “openUTM programming hints” and “CICS programming hints” in the following chapter.

3.2 Generating openUTM-LU62

openUTM-LU62 requires certain configuration information that the system administrator must provide in the form of a configuration file to start. The configuration file is created in 2 steps:

1. Creation of a generation file with a text editor. The format of the generation file is described in the following section.
2. Creation of the binary configuration files from the generation file. To do this the `u62_gen` program must be started. See below for more detailed information.

3.2.1 Generation file format

All substitute applications and their assignments to the real partners must be defined in the generation file. The INSTANCE statement is used for this purpose. One INSTANCE statement is required per openUTM-LU62 entity. A generation file can contain any number of INSTANCE statements.

INSTANCE statement format

The word INSTANCE must stand alone in a line.

All INSTANCE statement parameters consist of a keyword, the '=' sign and a value.

Parameters of the INSTANCE statement must be separated by a comma, and a comma may never be placed at the beginning of a line. The last parameter must not be followed by a comma. Optional parameters are shown in square brackets. Default values are assumed for any optional parameters not specified. The parameters can be placed in any order within the INSTANCE statement. A parameter must fit into one line and must not span across two lines.

Lines beginning with an asterisk (*) or the number sign (#) are treated as comments and not read by `u62_gen`.

The symbols allowed to be used in names include all printable ASCII characters (letters, numbers, special symbols) with the exception of the following special symbols:

– asterisk (*)
– comma (,)
– semicolon (;)
– space (0x20)
– tabulator (0x09)

3.2.2 INSTANCE statement

You specify the substitute and the parameters for the connection between a UTM access point and an LU6.2 application, i.e. for a openUTM-LU62 entity, using the INSTANCE statement.

As of openUTM-LU62 Version 5.1 you can choose whether the TCP/IP adress information is to be generated in TNSX or in the INSTANCE statement. There is no need to use TNSX when the information is generated in the INSTANCE statement.

Substitute LU

The substitute LU is specified with the LOC-LU-ALIAS parameter.

Substitute AE

When the TCP/IP adress information is generated in the INSTANCE statement, the substitute AE has to be generated with the LOC-APT, LOC-AEQ, LOC_TSEL and LOC-LISTENER-PORT parameters.

With TNSX the LOC-APT, LOC-AEQ and LOC-AE parameters have to be used.

Connection between openUTM-LU62 and LU6.2 application

The connection between openUTM-LU62 and the openUTM LU6.2 application has to be specified with the REM-LU-ALIAS, MODENAME, ALLOC-TIME and LU62-CODE parameters.

Connection between openUTM-LU62 and UTM application

When the TCP/IP adress information is generated in the INSTANCE statement, at least the REM-APT, REM-AEQ, REM-NSEL, REM-TSEL, REM-LISTENER-PORT and CONTWIN parameters are required for the connection between openUTM-LU62 and the UTM application.

With TNSX the REM-APT, REM-AEQ, REM-AE and CONTWIN parameters have to be used.

The remaining parameters (APPL-CONTEXT, ASSOCIATIONS, CONNECT, OSI-TP-CODE, UTM-ASYNC) can be used to specify additional parameters for the connection with the UTM application.

Syntax of the INSTANCE statement:

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

If there is more than one INSTANCE statement in the generation file, the triple REM-APT, REM-AEQ, REM-LU-ALIAS can only occur in one of them.

The parameters have the following meanings:

ALLOC-TIME = alloc_time

The maximum time in seconds that openUTM may wait for the establishment of a connection to the LU6.2 application during the initiation of an LU6.2 job (e.g. of a CICS transaction code) or that the LU6.2 application may wait for the establishment of a connection to openUTM during the initiation of a UTM transaction code.

Default value: 30 seconds.
Possible values: 0 - 300 seconds.
The value 0 means that the time will not be monitored.

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

Name of the application context that is used for the connection to the UTM application. The value must correspond to the value of the APPLICATION-CONTEXT operand from the associated OSI-LPAP statement in the UTM application.

Possible values are UDTCCR, UDTSEC, UDTAC and UDTDISAC. These values have the same meanings as for openUTM.

The default value is UDTSEC.

If UDTSEC is not used, then it is impossible to use user IDs and passwords when opening a conversation.

If UDTAC or UDTDISAC is specified, openUTM-LU62 works without transaction management.

ASSOCIATIONS = ass_number

The maximum number of parallel connections between openUTM-LU62 and a UTM application. This value should correspond to the value of the ASSOCIATIONS parameter from the associated OSI-LPAP statement in the UTM generation. If TRANSIT-SERVER is used, it should also match the value of SESS-MAX in the XMODE statement of TRANSIT-SERVER. If SNAP-IX or the IBM Communications Server is used, it should match the session limit in the mode definition.

Minimum value: 1
Default value: 1
Maximum value: 254

CONNECT = conn_number

The number of connections to the UTM application that are to be automatically established during the initiation of an openUTM-LU62 entity. Permissible values are whole numbers from 0 to ass_number.

The default value is 0.

If openUTM-LU62 cannot establish the desired number of connections during startup, then a new attempt is made every 5 minutes.

The CONNECT parameter is effective throughout the openUTM-LU62 process, unless a connection is cleared by means of an administration command. If the partner UTM application establishes the number of connections specified for CONNECT, openUTM-LU62 does not establish any further connections.

CONTWIN = cwin_number

The number of connections to the UTM application in which openUTM-LU62 is the contention winner. The sum of *cwin_number* and the value of the CONTWIN parameter from the associated OSI-LPAP statement of the UTM generation should be equal to *ass_number*.

LOC-AE = loc_tns_name

Only when using TNSX.
Global name (up to 18 characters long) from the TNSX (Transport Name Service in UNIX systems) that describes the substitute AE. *loc_tns_name* consists of 5 parts, each separated by periods:

NP5.NP4.NP3.NP2.NP1

If some of the parts of the name are empty, then some of the periods can be dropped from the end of the name.

The *loc_tns_name* must be entered in the TNSX as a global name with the “local name” property. Also, a P-selector and an S-selector must be entered for layer 4 in addition to a T-selector. These selectors must be suited for use with the corresponding parameters in an OSI-CON statement from the UTM generation. It is recommended to select empty entries for the S-selectors and P-selectors.

LOC-AEQ = loc_ae_qualifier

Specifies the application entity qualifier of the substitute AE. *loc_ae_qualifier* must be a whole number between 1 and 67108863. It must correspond to the value of the AEQ operand in the associated OSI-LPAP statement of the UTM generation.

LOC-APT = loc_ap_title

Specifies the application process title of the substitute AE in the form of an object name, i.e. in the form

(n1, n2, n3,...)

n1 can take on the values 0, 1 or 2, and *n2* can take on the values 0 to 39. The application process title must correspond to the values of the APT operands from the associated OSI-LPAP statement of the UTM generation.
on.

LOC-LISTENER-PORT = loc_port

Only without use of TNSX, but then the parameter is mandatory.
TCP port number of the substitute AE with the scope 102, 1025 - 65535.

LOC-LU-ALIAS = loc_lu_alias_name

LU alias name (up to 8 characters long) of the substitute LU. The name must correspond to an LU name in the LU6.2 basic software configuration (TRANSIT-SERVER, SNAP-IX or IBM Communications Server. This name only needs to be coordinated between openUTM-LU62 and LU6.2 basic software.

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

Only without use of TNSX, but then the parameter is mandatory.
Transport selector of the substitute AE. The first character specifies the format:

T	TRANSDATA Only capital letters, numbers and the special characters @, # and $ are allowed, where the first character may not be a number.
A	ASCII
E	EBCDIC

The maximum length of the content (`loc_tsel`) for all formats is 8 characters.

LU62-CODE =

Description of the character set in which the user data of the LU6.2 application programs is encoded.

ASCII	The LU6.2 application programs use the 8 bit ASCII character set ISO 8859-1. This is typical for application programs under UNIX systems, Windows systems or OS/2 systems.
EBCDIC-500	The LU6.2 application programs use IBM-EBCDIC with code page 500. This is the international code page usually used by IBM on mainframes or AS/400 systems and is also the national code page for Belgium and Switzerland.
EBCDIC-273	The LU6.2 application programs use IBM-EBCDIC with code page 273. This is the national code page usually used by IBM on mainframes or AS/400 systems for Germany and Austria.
EBCDIC-037	The LU6.2 application programs use IBM-EBCDIC with code page 037. This is the national code page usually used by IBM on mainframes or AS/400 systems for the USA, Canada, the Netherlands and Portugal.
*NO	openUTM-LU62 does not perform code conversions on the user data. This value must be specified if the application programs transmit binary data, for example binary length fields (default value).

openUTM-LU62 carries out a code conversion on all user data if different character sets are generated in LU62-CODE and OSITP-CODE. If *NO is specified, then openUTM-LU62 does not perform code conversion. If *NO is specified here, then *NO must also be specified for OSITP-CODE.

MODENAME = mode_name

Mode name (up to 8 characters long) that is to be used in the connections to the LU6.2 application. This name must also be defined in the LU6.2 basic software configuration (TRANSIT-SERVER, SNAP-IX or IBM Communications Server) and for the partner system. The mode name SNASVCMG must not be used.

OSITP-CODE =
Description of the character set used to encode the user data of the UTM application programs.

ASCII	The UTM application programs use the 8-bit ASCII character set ISO 8859-1. This is typical for UTM application programs under UNIX or Windows systems.
EBCDIC-BS2	The UTM application programs use the EBCDIC variant EBDIC.DF.04 Latin 1 typically used in BS2000/OSD.
*NO	openUTM-LU62 does not perform code conversion on the user data. This value must be specified if the UTM application programs transmit binary data, for example binary length fields (default value).

openUTM-LU62 performs code conversion on all user data if different character sets are generated in LU62-CODE and OSITP-CODE. If *NO is specified, then openUTM-LU62 does not perform code conversion. If *NO is specified here, then *NO must also be specified for OSITP-CODE.

REM-AE = rem_tns_name
Only when using TNSX.
Global name (up to 18 characters long) from the TNSX (Transport Name Service in UNIX systems) that describes the application entity (i.e. the access point) of the corresponding UTM application. *rem_tns_name* consists of 5 parts, each separated by periods:

NP5.NP4.NP3.NP2.NP1

If some of the parts of the name are empty, then some of the periods can be dropped from the end of the name.

The *rem_tns_name* must be entered as a global name with the “transport address” and “transport system” properties. Also, a P-selector and an S-selector must be entered for layer 4 in addition to a T-selector. These selectors must be suited for use with the corresponding parameters in an ACCESS-POINT statement from the UTM generation. It is recommended to select empty entries for the S-selectors and P-selectors.

REM-AEQ = rem_ae_qualifier
Specifies the application entity qualifier of the UTM application. *rem_ae_qualifier* must be a whole number between 1 and 67108863. It must correspond to the value of the AEQ operand from the associated ACCESS-POINT statement of the UTM generation.

REM-APT = rem_ap_title

Specifies the application process title of the UTM application in the form of an object name, i.e. in the form

(n1, n2, n3,...)

These must correspond to the values of the APT operands from the associated UTMD statement of the UTM generation.

REM-LISTENER-PORT = rem_port

Only without use of TNSX, but then the parameter is mandatory.
TCP port number of the partner UTM application with the scope 102, 1025 - 65535.

REM-LU-ALIAS = rem_lu_alias_name

LU alias name (up to 8 characters long) for the remote LU under which the LU6.2 application (for example, the CICS region) can be reached. It must correspond to an RLU name in the LU6.2 basic software configuration (TRANSIT-SERVER, SNAP-IX or IBM Communications Server). This name only needs to be coordinated between openUTM-LU62 and LU6.2 basic software.

REM-NSEL = rem_host

Only without use of TNSX.
(DNS-)Name of the host, on which the partner UTM application is running.

The host name may not contain dots (domain name). The maximum length is 16 characters.

If the partner system resides in another domain, e.g. the `search` parameter in the `/etc/resolv.conf` file under UNIX has to be extended with the relevant domain name.

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

Only without use of TNSX, but then the parameter is mandatory.
Transport selector of the partner UTM application. The first character specifies the format:

T	TRANSDATA Only capital letters, numbers and the special characters @, # and $ are allowed, where the first character may not be a number.
A	ASCII
E	EBCDIC

The maximum length of the content (`loc_tsel`) for all formats is 8 characters.

UTM-ASYNC =

This parameter specifies how a service started by a UTM application program is forwarded to the LU6.2 partner without a functional unit commit and without a functional unit handshake (i.e. in the case of APRO with KCOF=B).

NO	openUTM-LU62 starts these services with sync-level 0 (NONE).
YES	openUTM-LU62 starts these services with sync-level 1 (CONFIRM).

The value YES is necessary when a service is to be started without transaction management for the LU6.2 partner from the UTM application program by means of APRO AM.
If a UTM application uses APRO DM with KCOF=B and APRO AM with KCOF=B, a separate instance of openUTM-LU62 should be started for each type of communication. The first instance is generated with UTM-ASYNC=NO and the second instance with UTM-ASYNC=YES.

3.2.3 Starting the generation program

The `u62_gen` generation program is used to create a binary configuration file from a generation file and simultaneously check the generation file for logical errors. On UNIX systems it may only be called by the system administrator.

On UNIX systems, you can call `u62_gen` directly from the shell. On Windows systems, it is recommended that you create a DOS shell by selecting “Start > Programs > openUTM-LU62 > Prompt“. Under this DOS shell, enter the following command without the prefix `/opt/lib/utmlu62/`.

Call syntax:

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

Description:

gen_file	Name of the generation file from which the generation parameters are to be read. The default value is `/opt/lib/utmlu62/gen/genfile` (UNIX systems) or `Programs\utmlu62\gen\genfile` (Windows systems).
-f *conf_file*	Name of the configuration file in which the generation data is to be written in binary form. If *conf_file* is not specified, then the results are output to the file `/opt/lib/utmlu62/conffile` (UNIX systems) or `Programs\utmlu62\conffile` (Windows systems).

The `u62_tp` program reads the configuration data from the file `conffile` during its initialization. This configuration file can also be recreated while `u62_tp` is running using `u62_gen`. The new configuration data will only take effect during the next start of the corresponding openUTM-LU62 entity.

`u62_gen` does not delete any log records required for a recovery. See page 40 for more information on the subject of recovery.

3.2.4 Restoring generation files

If you want to create a readable generation file from a binary configuration file to check the generation run, for example, then you can do so with the following command:

Call syntax:

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

Description:

-r	(restore)
conf_file	Name of the binary configuration file. If conf_file is not specified, then the file `/opt/lib/utmlu62/conffile` is read.

`u62_gen` outputs the data to `stdout`.

The information relating to Windows systems specified in the section “Starting the generation program” on page 52 also applies here.

3.2.5 Displaying the name of the generation file used

If you have forgotten the name of the generation file used, the following command allows you to display the file name. It also indicates the internal version numbers of openUTM-LU62 programs.

Call syntax:

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

The information relating to Windows systems specified in the section “Starting the generation program” on page 52 also applies here.

3.3 Administering openUTM-LU62

3.3.1 Administration under UNIX systems and Windows systems

Administration under UNIX systems

Under UNIX systems, all administration commands can be called directly from the shell exactly as specified below.

openUTM-LU62 can be administered by all users after installation. If you want to limit the group of administrators of openUTM-LU62, you can create a list of the user IDs with administration authorization in the `/opt/lib/utmlu62/u62_users` file. This list has the following syntax:

- User IDs are separated by a comma, blank, tab or the end of a line. The user *root* always has administration authorization and does not need to be entered.
- Comments are preceded by a hash character (#) and extend to the end of the line.

If this file does not contain any entries or does not exist, all users have administration authorization.

Administration under Windows systems

Under Windows systems, it is recommended that you create a DOS shell by selecting “Start > Programs > openUTM-LU62 > Prompt”. Under this DOS shell, enter the commands described here without the prefix `/opt/lib/utmlu62/`.

3.3.2 Starting openUTM-LU62

openUTM-LU62 is started as a background process with `u62_start`.

Call syntax:

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

-l *loc_lu_alias_name*

If the -l switch is not specified, then `u62_start` determines a list of all openUTM-LU62 entities based on the configuration file and starts all these entities. If the -l switch is specified, then only the entity belonging to the LU *loc_lu_alias_name* is started. *loc_lu_alias_name* must correspond to a name from the openUTM-LU62 configuration.

-ton[,*trace_options*]

(trace on) Activates the trace. The amount of detail recorded by the trace can be controlled by the *trace_options*.

trace_options can be used to specify if and with how much detail the entity trace is to record and whether the XAP trace is to be activated. All internal operations of the `u62_tp` program are logged with the entity trace (IN). There are different trace levels for this entity trace, which are represented by numbers. Every level includes all of the traces in the lower levels. All internal operations of the XAP-TP provider and OSS are recorded with the XAP-TP trace (XAP). The following can be specified for the various *trace_options*:

in=1	The entity trace is activated with the lowest level. Only the trace entries required for a protocol trace are recorded. See page 67 for more information on the protocol trace. This trace level is generally sufficient for an initial error diagnosis.
in=2	The entity trace is activated with the detailed level.
in=3	The entity trace is started with an additional timer for all activities. This level is intended for diagnosing performance problems.
in	The entity trace is activated with the detailed level (like in=2).
xap	The XAP-TP trace is activated.

If both traces are to be activated, then in, xap must be specified as the *trace_options*.
If no *trace_options* are specified, then the entity trace is activated with the detailed level and the XAP-TP trace.

{ -c | -k }

Without a specification a warm start is executed by default. (`start type = w`).

-c
(cold) A cold start is executed. The cold start deletes all recovery information. You should only use this parameter in special cases. See page 40 for more information.

-k
(Lukewarm) A warm start is executed, during which all open transactions are discarded.

The log names however remain, which means that the LU6.2 partner application receives no information on the discard of the transactions.

This is necessary, if e.g. the partner system has discarded the open transactions already because of an error or via administrative interference and a cold start is not possible because it would have effects on other tasks in the partner system. In this case the partner system has to execute a cold start.
A cold start in e.g. IMS requires termination and reboot of the complete IMS. Thus the operation would be completely interrupted, what in production systems naturally is undesired. Starting with the `-k` parameter allows you to revise transactions pending on one side of the partners. In this case no further actions are necessary on the partner system side, especially no interruption.

Example: Start with a timer in the entity trace and an XAP-TP trace on UNIX systems:

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

After a successful start, the following message appears on the screen:

```
u62_start 17  * * *   openUTM-LU62 started:   * * *
                  local LU alias name = T02CGEI4,
                  PID = 1771,
                  start type = w
```

The values for the local LU alias name and PID are replaced by their actual values.

Normally, you will want openUTM-LU62 to start automatically on system startup and terminate automatically on system shutdown. On UNIX systems this is achieved by means of the `/etc/rc2.d/S30u62` file or a file of the same name in a different system RC directory.

On Windows systems, the installation process gives you the opportunity to decide whether or not openUTM-LU62 is to be started as a service. If you choose this option, the operating system automatically starts all openUTM-LU62 entities each time it boots up. If parameters are to be transferred during autostart, these must be entered in the U62_SERVICE_ARGS variable (in the directory `HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\openUTM-LU62\CurrentVersion`) using the operating system's Registry editor. If you start an instance by means of the services administration of the Windows system, U62_SERVICE_ARGS is not evaluated. Instead, the start parameters must be entered manually.

If you wish to have openUTM-LU62 started as a service without having to reinstall the product, you can also use the following command:

```
u62_start -r
```

To cancel the autostart, simply issue the following command:

```
u62_start -u
```

3.3.3 Terminating openUTM-LU62

openUTM-LU62 is terminated with `u62_adm -e` (end).

Call syntax:

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

Description:

-l *loc_lu_alias_name*
: If the -l switch is not specified, then all openUTM-LU62 entities are stopped. If the -l switch is specified, then only the entity belonging to the LU *loc_lu_alias_name* is stopped.

On Windows systems, you can also select “Start > Programs > openUTM-LU62 > Exit openUTM-LU62”.

You must exit openUTM-LU62 before terminating TRANSIT.

3.3.4 Displaying status information

You can obtain information on the current status of openUTM-LU62 with `u62_sta`. This command can be used by every UNIX system user. On Windows systems, select “Start > Programs > openUTM-LU62 > Show Status”.

Call syntax:

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

Description:

-l *loc_lu_alias_name*

If the `-l` switch is specified, then only information on the entity belonging to the LU *loc_lu_alias_name* is output.

-c

If the switch `-c` is specified only the number of LU62 session in the local default LU with the CPSVCMG mode is output.

If neither `-l` nor `-c` is specified, then status information is output for all openUTM-LU62 entities.

-b

The data is output in binary form corresponding to a predefined C structure with the -b switch. This switch can be used for programs that are to process the output from `u62_sta`. The C structure is defined in the header file `/opt/lib/utmlu62/status.h` (UNIX systems)
or
`Programme\utmlu62\status.h` (Windows systems).

`u62_sta` outputs two header lines and one line per openUTM-LU62 entity. The output appears similar to the output below:

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

With the `-c` switch `u62_sta` z.B. creates the following output:

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

Meanings of the output fields:

LLU name	Name of the local LU as it was specified in the generation file with LOC-LU-ALIAS.
RLU name	Name of the partner LU as it was specified in the generation file with REM-LU-ALIAS.

local OSI addr

TNSX name of the local application entity as it was specified in the generation file with LOC-AE.

Without use of TNSX:
Local T-selector without format specification and the local port in brackets.
Example: `LOC_TSEL(23766)`

remote OSI addr

TNSX name of the partner application entity as it was specified in the generation file with REM-AE.

Without use of TNSX:
T-selector of the partner application without format specification and the local port in brackets, followed by the host name and the TCP port (if there is enough space).

Example:
```
REM_TSEL(host:102)
REM_TSEL(host)
```

If the host name is too long it will be cut. You will be notified with `...`
```
REM_TSEL(hostn...)
```

XLN

Exchange log name status.

If Sync-level 2 is specified, then an exchange log name command must be executed between openUTM-LU62 and the LU6.2 application after restarting the openUTM-LU62 entity or after an abnormal connection termination. Only after the exchange log name command has been successfully terminated can transactions be started. openUTM-LU62 executes this exchange log name command independently. You may need to intervene manually when an error occurs.

The following is a list of the possible states:

-	No exchange log name required because sync-level 2 is not used.
run	The exchange log name command is currently running.

wai	The exchange log name command is still outstanding. The following is a list of the possible causes: – there is no connection to the LU6.2 application – the previous exchange log name command was cancelled due to a communication error – the openUTM-LU62 and LU6.2 basic software configurations (TRANSIT-SERVER, SNAP-IX or IBM Communications Server) are incompatible, i.e. the LU6.2 basic software configuration was changed while openUTM-LU62 was running.
ok	The exchange log name command was successfully executed.
err	A fatal error occurred during the exchange log name command.
rst	(Restart) This is displayed as the XLN status until the recovery to the UTM application is completed. The log names could not be successfully exchanged with the LU6.2 partner. This status lasts only for a short time after the instance starts up, but it can also last for a longer period if the LU6.2 partner is responsible for the recovery of the transaction.

If `err` or `rst` is displayed, or `wai` is displayed although a connection exists, then additional information can be found in the file

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

or

`Programs\utmlu62\PROT\prot.`*luname*`.txt` (Windows systems).

atot	(Associations total) The number of parallel connections established between the openUTM-LU62 entity and the UTM application.
absy	(Associations busy) The number of busy parallel connections to the UTM application. A parallel connection is busy when program-program communication is currently taking place via this connection.
stot	(Sessions total) The number of LU6.2 sessions opened between the openUTM-LU62 entity and the LU6.2 partner. If communication to the LU6.2 partner is to be possible, then at least 3 sessions must be opened. Two sessions are always required for internal administration.
sbsy	(Sessions busy) The number of busy sessions. A session is busy when a conversation (program-program communication) is currently running via this session.

i You can find additional status information, in particular when an error occurs, in the file

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

or

`Programs\utmlu62\PROT\prot.`*luname*`.txt` (Windows systems).

3.3.5 Establishing connections

You can also establish connections to the UTM application or to the LU6.2 application with the command `u62_adm`.

Call syntax:

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

Description:

-co	(Connect OSI-TP) A parallel connection is established via OSI-TP to the UTM application.
-cs	(Connect SNA) An SNA-LU6.2 session is opened to the LU6.2 application.
-l *loc_lu_alias_name*	If the -l switch is not specified, then a connection is established by every openUTM-LU62 entity. If the -l switch is specified, then a connection is only established to the entity belonging to this LU.

i You need to call `u62_sta` afterwards to check if the connection was established successfully.

3.3.6 Clearing connections

You can also clear connections to the UTM application or to the LU6.2 application using the `u62_adm` command.

Syntax:

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

Description:

-ao	(Abort OSI-TP) All existing parallel connections to the UTM application are cleared, regardless of whether or not they are busy.
-as	(Abort SNA) All existing SNA sessions to the LU6.2 application are closed, regardless of whether or not they are being used for a conversation (program-program communication).
-do	(Disconnect OSI-TP) All existing parallel connections to the UTM application that are not busy are cleared.
-ds	(Disconnect SNA) All existing SNA sessions to the LU6.2 application are closed. Sessions that are currently being used for a conversation are not closed until the conversation has ended. Free sessions are closed immediately.

-l *loc_lu_alias_name*

If the -l switch is not specified, then the connections from each of the openUTM-LU62 entities are cleared. If the -l switch is specified, then only the connection to the entity belonging to this LU is cleared.

i You must check whether or not the connection has been cleared successfully by calling `u62_sta` afterwards.

3.3.7 Activating and deactivating traces

Trace information is useful for error diagnosis in openUTM-LU62. When traces are activated, then the `u62_tp` program continuously writes internal information to special trace files.

Call syntax:

```
/opt/lib/utmlu62/u62_adm -ton[,trace_options ] [ -l loc_lu_alias_name ]

/opt/lib/utmlu62/u62_adm -tof[,trace_options ] [ -l loc_lu_alias_name ]

/opt/lib/utmlu62/u62_adm -tfl[,trace_options ] [ -l loc_lu_alias_name ]
```

Description:

-ton (Trace On) The trace is activated.

-tof (Trace Off) The trace is deactivated.

-tfl (Trace Flush) The trace is flushed, i.e. the current trace file is closed and a new trace file is created.

trace_options Controls the type of trace and the amount of detail recorded. The following *trace_options* are allowed. You can also specify several trace options separated by commas.

in	The entity trace is activated, deactivated or flushed. If this trace option is specified for -ton, then the entity trace is activated with the detailed level (in=2).
xap	The XAP-TP trace is activated, deactivated or flushed.
in=1	(Only for -ton) The entity trace is activated with the lowest level. Only the trace entries required for a protocol trace are recorded. See page 67 for more information on the protocol trace. This trace level is generally sufficient for an initial error diagnosis.
in=2	(Only for -ton) The entity trace is activated with the detailed level.
in=3	(Only for -ton) The entity trace is started with an additional timer for all activities. This level is intended for diagnosing performance problems.

If no *trace_options* are specified, then the `in,xap` options are assumed.

-l *loc_lu_alias_name*
If the -l switch is not specified, then the command affects all openUTM-LU62 entities. If the -l switch is specified, then the command only affects entity belonging to this LU.

Example: Activating the detailed trace with a timer on UNIX systems:

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

i If you want to run the trace right from the start, then you can accomplish this by specifying the -t option for the `u62_start` command.

The traces are stored in the following files:

– entity trace file: `/opt/lib/utmlu62/PROT/inlog.`*loc_lu_alias.suff*

– XAP-TP trace file: `/opt/lib/utmlu62/PROT/xaplog.`*loc_lu_alias.suff1.suff2*

On Windows systems, the corresponding files are located under `Programs\utmlu62\PROT`.

The suffix *suff* assumes the value 0 initially. When the file has grown to a size of about one-half megabyte or the flush command described above is entered, then the trace file is closed and a new trace file is created with the suffix incremented by 1. The *suff* is reset to 0 once the file with *suff*=9 has been closed.

For an XAP-TP trace, *suff1* is only incremented when a flush command is entered, and *suff2* is only incremented when the maximum file size has been reached.

The value of 10 for the cyclical overwriting of the traces can be changed on UNIX systems by setting the shell variable U62_TRC_FILES before openUTM-LU62 is started. If you want to create 20 trace files, for example, then you must enter the following commands:

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

The shell variable U62_TRC_LEN can be used to restrict the length of user data in the trace.

The shell variable U62_TRC_COMPRESS can be used to compress the trace files on UNIX systems. To achieve this, U62_TRC_COMPRESS must be set to “compress” or “gzip”. Note that the “compress” or “gzip” programs are located using the PATH variable.

On Windows systems, variables with the name same as the shell variables can be set using the operating system’s Registry editor. The variables are located in the directory `HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\openUTM-LU62\CurrentVersion`.

3.3.8 Analyzing traces

The following command is provided for analyzing the entity trace:

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

Description

-f tracefile	(File) The entity trace files with the names *tracefile* (or *tracefile2*, etc.) are converted to a readable form and output.
-o outfile	Name of the output file.
-p	An additional protocol trace is output to the file *outfile*`.flow`. `u62_adm` calls the awk program `u62_flow` for this purpose. This protocol trace is described in more detail in the following section.

Example:

If the trace files `inlog.T02CGEI4.0` and `inlog.T02CGEI4.1` were stored in the UNIX system directory `/opt/lib/utmlu62/PROT`, then you can analyze these files with the following commands:

```
cd /opt/lib/utmlu62/PROT
../u62_adm -fpo outfile inlog.T02CGEI4.0 inlog.T02CGEI4.1
```

The files `outfile` and `outfile.flow` are created in the directory `/opt/lib/utmlu62/PROT` in the process.

The following command is available to analyze the XAP-TP trace:

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

`step` prepares the XAP-TP traces and outputs the traces in a readable form to `stdout`.

On Windows systems, `step` is located under `Program Files\utmlu62`.

3.3.9 Creating a dump

In some error situations, it may be necessary to save openUTM-LU62 status information in the form of a dump to help with diagnostics. You use the following command for this:

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

This creates the following files:

```
/opt/lib/utmlu62/PROT/in.dump.loc_lu_alias.1
/opt/lib/utmlu62/PROT/xap.dump.loc_lu_alias.1.DIAG00
```

On Windows systems the files are located under `Program Files\utmlu62\PROT`.

i With each call of `u62_adm -b`, the counter which is `1` in the example above, is incremented by `1`.

3.3.10 Protocol trace

You will find an abridged representation of the protocol flow from LU6.2 and OSI-TP in the protocol trace. Because of the fact that openUTM-LU62 uses the APPC program interface for LU6.2 communication and the XAP-TP program interface for OSI-TP communication, several properties of these two interfaces can also be viewed in this trace.

On the LU6.2 side, the name of the APPC call, the TP-ID assigned by LU6.2 basic software (TRANSIT-CPIC, SNAP-IX or the IBM Communications Server, the direction of processing and any additional parameters are output for each message in the trace. The direction of processing is marked with an arrow. An arrow pointing to the left marks a message sent by openUTM-LU62, and an arrow pointing to the right marks a message received by openUTM-LU62. First, administration data is exchanged between openUTM-LU62 and the LU6.2 partner when an openUTM-LU62 entity is started and after the loss of a connection. The transaction code X'06F2' is used for this purpose. These protocol flows are marked with a single arrow (--->). Protocol flows that come from application programs are marked with a double arrow (===>).

On the OSI-TP side, the name of the XAP-TP call, the entity descriptor (fd) assigned by the XAP-TP provider, the direction of processing and any additional parameters are output for each message in the trace. The direction of processing is marked with an arrow. An arrow pointing to the right marks a message sent by openUTM-LU62, and an arrow pointing to the left marks a message received by openUTM-LU62. A control entity is used on the XAP-TP interface when an openUTM-LU62 entity is started and during the commit phase. These protocol flows are marked with a single arrow (--->). All other protocol flows are marked with a double arrow (===>).

Each message contains a correlation number (corr) in order to make it easier to associate an LU6.2 conversation and a parallel connection via XAP-TP. Protocol flows that are not associated with a conversation are assigned the correlation number 0. Additionally, every message is output with the time and the number of the corresponding line in the original output file.

The protocol trace does not contain user data. This data can only be found in the original output file.

The following example of a protocol trace contains additional comments. These comments sometimes refer to the terms used in the following chapter, for example the names of CICS commands.

The example has the following sequence:

1. Exchange log name
2. Call of a CICS transaction by a UTM application program
3. Call of a UTM transaction by a CICS application program

Exchange log name

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

openUTM-LU62 was started. The XAP-TP provider sign-on was successful.

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

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

CICS has opened a conversation with the TP name X’06F2’ and has sent an exchange log name command.

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

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

openUTM-LU62 replies to the exchange log name. Transaction-oriented communication between openUTM-LU62 and CICS is only possible after this point in time.

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

Call of a CICS transaction by a UTM application program

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

A UTM application program was started and wants to establish a connection to a CICS application program.

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

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

openUTM-LU62 opens a conversation to the desired CICS application program using the transaction code UPS1. You can see that the connection is a transaction-oriented connection by the line containing sync_level = SYNCPT. No user ID and password pair is sent.

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

The UTM application sends 19 bytes of data. These 19 bytes also contain an OSI-TP header.

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

openUTM-LU62 sends the data on to the CICS application program. You can see here that this data consists of just 2 bytes of user data. The OSI-TP header was therefore 17 bytes long (the length of the OSI-TP header may vary).

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

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

The UTM application program has issued a CTRL PE and a PEND KP after the MPUT NE. The CTRL PE can be regarded as a TP_DEFERRED_END_DIALOGUE_IND and a TP_PREPARE_IND.

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

openUTM-LU62 sends a prepare to the CICS application program, thereby requesting that the end of transaction be initiated.

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

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

The CICS application program has confirmed the end of transaction with the CICS command SYNCPOINT. CICS therefore sends a Request Commit and gives up its send authorization.

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

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

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

openUTM-LU62 transmits the Request Commit as a TP_COMMIT_REQUEST to the UTM application. This ends the first phase of the 2-phase commit protocol. The XAP-TP calls TP_PREPARE_ALL_REQ and TP_READY_ALL_IND issued beforehand are only local in nature.

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

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

The UTM application now starts the second phase of the 2-phase commit protocol. openUTM-LU62 receives two TP_COMMIT_INDs. openUTM could also have rolled back the transaction up to this point in time. This is not possible anymore after TP_COMMIT_IND.

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

openUTM-LU62 sends a Committed command to CICS based on the TP_COMMIT_IND received.

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

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

CICS replies with a Forget. The DEALLOC_NORMAL thereafter shows that the conversation was terminated as desired after the end of transaction in the same manner as for a UTM application program when CTRL PE is sent.

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

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

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

openUTM-LU62 transmits the Forget it has received as a TP_DONE_REQ to the UTM application. After the second TP_COMMIT_COMPLETE_IND has been received, the transaction and the OSI-TP dialog are terminated.

Call of a UTM transaction by a CICS application program

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

A CICS application program was started and wants to establish a connection to a UTM application program with the transaction code UCS1.

```
                                 APM_ALLOCATE_REQ (fd = 1)
 10:47:45, line 5615, corr 2    ========================================>
                                 APM_ALLOCATE_CNF+ (fd = 1)
 10:47:45, line 5687, corr 2    <========================================
                                 TP_BEGIN_DIALOGUE_REQ (fd = 1)
 10:47:45, line 5848, corr 2    ========================================>
                                udata_len = 28
```

openUTM-LU62 establishes a connection to the desired UTM application program.

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

The CICS application program sends 2 bytes of data.

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

openUTM-LU62 sends the data on to the UTM application program. Here you can see that openUTM-LU62 places a 21 byte long OSI-TP header in front of the data.

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

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

The CICS application program has requested the UTM application program to end the transaction with the SYNCPOINT command. CICS sends a Request Commit and gives up its send authorization. CICS does not use a real 2-phase commit protocol in this case, rather a simplified protocol that makes it impossible to roll back the transaction later on. CICS always uses this simplified protocol when the CICS application program only has one job-receiver and has not issued an ISSUE PREPARE before the SYNCPOINT command.

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

That which was not recognizable in the protocol trace when the Request Commit was received becomes clear here: The CICS application program has sent the message with SEND LAST before the SYNCPOINT and therefore initiated the termination of the conversation. openUTM-LU62 therefore sends a TP_DEFERRED_END_DIALOGUE_REQ on to the UTM application.

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

After that, openUTM-LU62 transmits the request to end the transaction to the UTM application.

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

The UTM application program has confirmed the end of transaction by issuing PEND FI. openUTM-LU62 therefore receives a TP_READY_ALL_IND.

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

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

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

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

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

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

All TP_COMMIT_REQ calls up to the TP_COMMIT_COMPLETE_IND call serve to ensure that the transaction is terminated in accordance with the protocol.

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

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

After the transaction has been terminated successfully on the OSI-TP side, openUTM-LU62 sends a Committed to CICS and then ends the conversation with MC_DEALLOCATE. This also terminates the transaction and the conversation on the LU6.2 side.

4 openUTM-CICS interconnection via LU6.2

In this chapter you will find a discussion of the aspects of openUTM-CICS interconnection via LU6.2 as they relate to generation and programming.

4.1 Generating an openUTM-CICS interconnection

4.1.1 Definitions in CICS

The following definitions are required in CICS:

- Definition of APPLID
- A connection and a session definition for each instance of openUTM-LU62

The CICS system administrator has several possibilities to define connections and sessions, for example RDO (Resource Definition Online) using CEDA, DFHCSDUP or macros.

APPLID

The name of the CICS application must be specified in the system initialization table (SIT):

DFHSIT APPLID=cics_netname
: Name (1-8 characters) with which the SNA network name is specified for the CICS application. This name must match the definitions in VTAM (VBUILD TYPE=APPL).

DEFINE CONNECTION

The remote system, in this case the substitute LU in openUTM-LU62, is described with this definition.

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

The operands have the following meaning:

CONNECTION connection_name
: Name (1-4 characters) used to designate the UTM application. This name only exists within CICS. It must be specified in the CICS programs in the parameter SYSID for ALLOCATE if the CICS program is to call a UTM transaction code.

GROUP group_name
: Name (1-8 characters). The CICS system administrator must assign this name. The name only appears in CICS definitions.

NETNAME netname
: Name (1-8 characters) of the substitute LU. This name must match the LU name defined in VTAM and the name defined in the basic LU6.2 software. If TRANSIT is the basic LU6.2 software, this is the second part of the name assigned for XLU NETNAME. Depending on the SNA protocol used, it may not be necessary to define LU names in VTAM.

ACCESSMETHOD VTAM
: VTAM access method.

PROTOCOL APPC
: An LU6.2 connection is defined.

SINGLESESS NO
: Parallel sessions are allowed. openUTM-LU62 requires parallel sessions.

DATASTREAM USER
: The user data will be defined completely by the application program.

RECORDFORMAT U
: The user data will be sent as a chain of RUs.

AUTOCONNECT { NO | YES | ALL }
: The sessions will be opened at the start of CICS when YES or ALL is specified. This is not true for NO.

INSERVICE YES
: The connection is available.

ATTACHSEC { LOCAL | IDENTIFY | VERIFY | MIXIDPE }
: The UTM application may not specify a user ID and password for APRO (KCSECTYP=N) when specifying LOCAL (default value).
The UTM application must specify KCSECTYP=S in APRO when specifying IDENTIFY.
The UTM application must specify KCSECTYP=P in APRO and a user ID and password when specifying VERIFY.
The UTM application can select either KCSECTYP=S or KCSECTYP=P when specifying MIXIDPE.

DEFINE SESSIONS

The properties of the connection to the substitute LU in openUTM-LU62 are described with this definition.

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

The operands have the following meanings:

SESSIONS session_name
: Name of the sessions definition. This name only appears in CICS definitions.

GROUP group_name
: Name (1-8 characters). The CICS system administrator must assign this name. The name only appears in CICS definitions. The same name should be chosen as was chosen in the associated connection definition.

CONNECTION connection_name
: Name of the associated connection definition.

MODENAME mode_name
: Name (1-8 characters) for the connection between CICS and openUTM-LU62. This name must also be defined as the mode name in VTAM and basic LU6.2 software (TRANSIT-SERVER, SNAP-IX or the IBM Communications Server), and openUTM-LU62.

PROTOCOL APPC
: An LU6.2 connection is defined.

MAXIMUM max1, max2

The maximum number of sessions between the two LUs is specified with max1. This value must be between 1 and 254. When using TRANSIT-SERVER, it should match the value of the parameter SESS-MAX in the associated XMODE statement of the TRANSIT configuration. When using SNAP-IX or the IBM Communications Server, it should match the session limit in the mode definition of the IBM Communications Server.

The maximum number of sessions in which the CICS side is the contention winner is specified with max2. When using TRANSIT-SERVER, it should match the value of the parameter SESS-LOS in the associated XMODE statement of the TRANSIT configuration. When using SNAP-IX or the IBM Communications Server, it need not be aligned with the IBM Communications Server configuration.

If the maximum values of CICS and the basic LU6.2 software are not the same, the systems agree on the lower value.

SENDSIZE ru_size RECEIVESIZE ru_size

This specifies the maximum RU (Request Unit) size for LU6.2 communication between CICS and openUTM-LU62. You should use the default value of 4096 if possible. Smaller RUs lead to higher network loads. You should select the same values for SENDSIZE and RECEIVESIZE.

The value specified here should correspond to the value in the MODE macro from VTAM. When using TRANSIT-SERVER, it should also match the value SRU-MAX or RRU-MAX configured there. When using SNAP-IX or the IBM Communications Server, it should match the maximum RU size defined for the mode name. Please note that the RU sizes must be specified in different formats in VTAM and TRANSIT-SERVER. In TRANSIT, the value 4096 is to be written as 89. More details are described in the TRANSIT manual.

AUTOCONNECT { NO | YES | ALL }

The sessions will be opened at the start of CICS when YES or ALL is specified. This is not true for NO.

INSERVICE YES

The connection is available.

BUILDCHAIN YES

The parameter must always be set to the default value YES.

IOAREALEN data_length

The minimum size of the terminal input/output area of CICS is specified with *value1*. This value should be selected so that it is at least as large as the maximum message length for the application programs.

4.1.2 VTAM generation

In cases in which the LUs are defined in VTAM, the substitute LUs used by openUTM-LU62 must be generated as independent LUs, meaning with LOCADDR=0. In addition, the mode name must be generated in VTAM. See the section below containing the complete sample generation.

4.1.3 TRANSIT generation for openUTM-CICS sessions

When using TRANSIT-SERVER, the SNA connection between the UNIX system and the SNA host must be configured in TRANSIT on the one hand. On the other hand, the LU6.2 connection between CICS and openUTM-LU62 must also be configured there. See the section below containing the complete sample generation.

4.1.4 SNAP-IX generation for openUTM-CICS sessions

When SNAP-IX is used, the SNA connection between the UNIX system and the SNA host must be configured in this product. The LU6.2 connection between CICS and openUTM-LU62 must also be configured there. See the complete generation example below.

4.1.5 Generating the IBM Communications Server for openUTM-CICS sessions

When using the IBM Communications Server, the SNA connection between the UNIX/Windows system and the SNA host must be configured in the Communications Server on the one hand. On the other hand, the LU6.2 connection between CICS and openUTM-LU62 must also be configured there. See the section below containing the complete sample generation.

4.1.6 openUTM-LU62 generation

The openUTM-LU62 generation is described in detail in the previous chapter. See the section below containing the complete sample generation to compare the generation parameters with those of the other generations.

4.1.7 TNSX generation

On the computer on which openUTM-LU62 is running, the TCP/IP address information for communication with the UTM application must be entered either in the TNSX (Transport Name Server UNIX Systems) or directly in the openUTM-LU62. The address information consists of IP addresses, port numbers and T-selectors. See also the complete generation example below.

4.1.8 openUTM generation

From openUTM's point of view, an LU6.2 interconnection must be generated in exactly the same manner as a OSI-TP interconnection. However, you should bear in mind that a maximum of 254 connections can be established per access point via openUTM-LU62. This means that the value of the ASSOCIATIONS parameter in the OSI-LPAP statement should not be larger than 254 for an LU6.2 connection.

4.1.9 Defining CICS transactions

Local transactions are defined for CICS using RDO as follows:

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

The operands have the following meanings:

TRANSACTION transaction_code
: Transaction code (1-4 characters) for the CICS transaction. The name must be specified in the KCRN filed for the UTM call APRO.

GROUP group_name
: Name (1-8 characters). The CICS system administrator must assign this name. It only appears in the CICS definitions.

PROGRAM program_name
: Name of the CICS program.

INDOUBT WAIT
: This must be specified to achieve proper synchronization with openUTM after session errors.

Remote transactions do not have to be defined in CICS. The transaction code is specified just like for openUTM in the CICS programs. In contrast to the LU6.1 interface, CICS also offers the possibility to specify transaction codes with more than 4 characters directly in the program for LU6.2. This is important because UTM transaction codes can be up to 8 characters long.

4.1.10 Using user IDs

When using distributed processing, it often makes sense to only allow the human end user to enter his or her user ID and password at one location in the entire system to prove that he or she is authorized to execute specific actions. The other partner in a job-submitter/receiver relationship is informed of the user ID so that the user only has certain authorizations in the other system, too. The password, on the other hand, is usually not transmitted over the network.

In order for such a procedure to function, the user IDs must be unique in all the participating systems. The system administrators of the participating systems must therefore enter the same user IDs to some extent. User IDs are specified as USER in the KDCDEF generation for openUTM, and for CICS the user IDs are usually administered using the RACF product.

If user IDs are to be transmitted together with the jobs, then the application context UDTSEC must be specified for openUTM-LU62 and for openUTM.

Application programs usually do not have to take care of sending the user ID when using the LU6.2 protocol. UTM application programs can send the user ID on to the CICS system by setting the KCSECTYP=S in APRO.

CICS programs always transmit the user ID. As mentioned above, however, it can only be transmitted to the openUTM application if UDTSEC was generated as an application context. If UDTSEC was not generated as an application context and the CICS program sends a user ID, then openUTM-LU62 closes the LU6.2 conversation.

If different user IDs are used in the openUTM environment than in the CICS environment, then the UTM application program can set a user ID and password itself as a job-submitter. The APRO parameter KCSECTYP=P must be set. The user ID must be entered in KCUSERID, and the password must be entered in KCPSWORD. Note that KCUIDTYP and KCPWDTYP are to be set to “P” or “T“. The “octet string” type is not supported by openUTM-LU62. Note that in this case all unused fields in the second parameter area must be deleted for APRO.

However, a CICS program can only send the user ID with which it was called. A CICS program cannot send a password to a UTM application.

If several UTM transactions are called from CICS simultaneously with the same user ID, the value RESTART=NO must be specified in the USER statement in the UTM generation.

In addition, the UTM application must be generated in such a way that users can be signed on with the same user ID more than once at any one time (SIGNON generation statement, MULTI-SIGNON=YES parameter).

4.1.11 A complete sample generation

A complete example of openUTM on UNIX system via CICS/ESA interconnection is shown in the following. The connection between CICS and the UNIX systems is implemented with SNA via Ethernet in this case. Generation parameters that must match are marked on the right border with a number from (1) to (43). These parameters are explained in even more detail at the end of this section.

The example contains communication in both directions: the calling of a CICS program by openUTM and vice versa.

CICS program

Addressing in the job-submitter program:

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

CICS definitions

System initialization:

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

CEDA definitions:

```
CEDA  View Connection( IDI4 )
    Connection     : IDI4                                              (1)
    Group          : KNAS1IND
    DEscription    :
   CONNECTION IDENTIFIERS
    Netname        : TO2CGEI4                                          (4)
    INDsys         :
   REMOTE ATTRIBUTES
    REMOTESYSTem   :
    REMOTEName     :
    REMOTESYSNet   :
```

```
CONNECTION PROPERTIES
 ACcessmethod   : Vtam
 PRotocol       : Appc
 Conntype       :
 SInglesess     : No
 DAtastream     : User
 RECordformat   : U
 Queuelimit     : No
 Maxqtime       : No
OPERATIONAL PROPERTIES
 AUtoconnect    : No
 INService      : Yes
SECURITY
 SEcurityname   :
 ATtachsec      : Local
 BINDPassword   :
 BINDSecurity   : No
 Usedfltuser    : No
RECOVERY
 PSrecovery     : Sysdefault

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

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

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

VTAM generation:

Start options:

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

Major nodes:

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

Application Definitions:

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

Logon mode table:

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

TRANSIT generation

```
 XLINK  L02CGE,                                                    (16)
        TYP=LAN,
        XID=0172000E                                          (11),(12)
 XPU    P02CGE,                                                    (17)
        TYP=PEER,
        LINK=L02CGE,                                               (16)
        DMAC=400037450001,                                         (18)
        MAXDATA=1033                                               (13)
 XLU    T02CGEI4,                                                  (19)
        TYP=6,
        SESS-CTR=IND,
        SESS-LMT=20,                                                (6)
        PUCONNECT=APHSTART,
        NETNAME=DESNI000.T02CGEI4,                              (4),(43)
        SEC-PAIR=NO_CHECK NO_PASS,
        PAIR_EXT=CICST6 MODDIS89 SYNCLEVEL ALREADY_VERIFIED     (5),(20)
 XRLU   CICST6,                                                    (20)
        NETNAME=DESNI000.A9CICST6,                              (3),(43)
        PU=P02CGE                                                  (17)
 XMODE  MODDIS89,                                                   (5)
        SESS-MAX=15,                                                (6)
        SESS-LOS=2,
        SESS-WIN=2,
        SESS-AUTO=2,
        SRU-MAX=89,                                                 (7)
        RRU-MAX=89                                                  (7)
```

openUTM-LU62 generation without use of TNSX

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

openUTM-LU62 generation with use of TNSX

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

TNSX generation

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

openUTM generation

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

UTM application program

Addressing in the job-submitting program:

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

Changes when using openUTM under BS2000/OSD

If you are using openUTM under BS2000/OSD instead of openUTM under UNIX systems, the sample generation changes as follows:

openUTM-LU62-Generierung without use of TNSX

```
INSTANCE
       LOC-LU-ALIAS      = T02CGEI4,                                 (19)
       LOC-TSEL          = T'SMP22804',                              (36)
       LOC-LISTENER-PORT = 22804,                                    (37)
       LOC-APT           = (1,2,3),                                  (22)
       LOC-AEQ           = 1,                                        (23)
       REM-LU-ALIAS      = CICST6,                                   (20)
       MODENAME          = MODDIS89,                                  (5)
       LU62-CODE         = EBCDIC-273,
       REM-TSEL          = T'SMP22800',                              (29)
       REM-LISTENER-PORT = 102,                                      (30)
       REM-NSEL          = utmhost,                                  (33)
       REM-APT           = (1,2,4),                                  (25)
       REM-AEQ           = 1,                                        (26)
       APPL-CONTEXT      = UDTSEC,                                   (27)
       ASSOCIATIONS      = 15,                                        (6)
       CONNECT           = 4,
       CONTWIN           = 5,                                        (28)
       OSITP-CODE        = ASCII
```

TNSX generation

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

BCAM generation

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

openUTM generation

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

The openUTM generation is otherwise identical. All other sections of the sample generation also remain unchanged.

Changes when using openUTM-LU62 under Windows systems:

When using openUTM-LU62 under Windows systems, only the TRANSIT generation is changed. Since this openUTM variant uses the IBM Communications Server for Windows instead of TRANSIT, the generation parameters must be defined in the syntax of the Communications Server. Normally, the generation parameters are entered via menus. In this way, the user can create a generation file (.acg) and select a name and storage location. By default, this file is stored in the following directory:
`<installation directory of IBM Communications Servers for Windows>\PRIVATE`.

Generation of the IBM Communications Server for Windows

```
NODE=(
     ANYNET_SUPPORT=NONE
     COMPRESS_IN_SERIES=0
     CP_ALIAS=P02CGE                                               (17)
     DEFAULT_PREFERENCE=NATIVE
     DISCOVERY_SUPPORT=DISCOVERY_CLIENT
     DLUR_SUPPORT=NORMAL
     FQ_CP_NAME=DESNI000.P02CGE                               (17),(43)
     MAX_LS_EXCEPTION_EVENTS=200
     NODE_ID=0172000E                                         (11),(12)
     NODE_TYPE=END_NODE
     REGISTER_WITH_CDS=1
     REGISTER_WITH_NN=ALL
     TP_SECURITY_BEHAVIOR=VERIFY_EVEN_IF_NOT_DEFINED
)
PORT=(
     PORT_NAME=LAN0_04                                             (16)
     ACTIVATION_DELAY_TIMER=0
     DELAY_APPLICATION_RETRIES=1
     DLC_DATA=00000000000004
     DLC_NAME=LAN
     IMPLICIT_BRANCH_EXTENDER_LINK=0
     IMPLICIT_CP_CP_SESS_SUPPORT=1
     IMPLICIT_DEACT_TIMER=0
     IMPLICIT_DSPU_SERVICES=NONE
     IMPLICIT_HPR_SUPPORT=1
     IMPLICIT_LIMITED_RESOURCE=NO
     IMPLICIT_LINK_LVL_ERROR=0
     LINK_STATION_ROLE=NEGOTIABLE
     MAX_ACTIVATION_ATTEMPTS=0
     MAX_IFRM_RCVD=8
     MAX_RCV_BTU_SIZE=1033                                         (13)
     PORT_TYPE=SATF
     RETRY_LINK_ON_DISCONNECT=1
     RETRY_LINK_ON_FAILED_START=1
```

```
        RETRY_LINK_ON_FAILURE=1
        PORT_LAN_SPECIFIC_DATA=(
             ACK_DELAY=100
             ACK_TIMEOUT=10000
             ADAPTER_NUMBER=0
             BUSY_STATE_TIMEOUT=15
             IDLE_STATE_TIMEOUT=30
             INB_LINK_ACT_LIM=128
             LOCAL_SAP=04
             MAX_RETRY=10
             OUTSTANDING_TRANSMITS=16
             OUT_LINK_ACT_LIM=127
             POLL_TIMEOUT=8000
             POOL_SIZE=32
             REJECT_RESPONSE_TIMEOUT=10
             TEST_RETRY_INTERVAL=8
             TEST_RETRY_LIMIT=5
             TOT_LINK_ACT_LIM=255
             XID_RETRY_INTERVAL=8
             XID_RETRY_LIMIT=5
        )
   )
   LINK_STATION=(
        LS_NAME=P02CGE                                                  (17)
        ACTIVATE_AT_STARTUP=1
        ACTIVATION_DELAY_TIMER=-1
        ADJACENT_BRANCH_EXTENDER_NODE=PROHIBITED
        ADJACENT_NODE_TYPE=SUBAREA_LEN
        AUTO_ACTIVATE_SUPPORT=0
        BRANCH_EXTENDER_LINK=0
        CP_CP_SESS_SUPPORT=0
        DEFAULT_NN_SERVER=0
        DELAY_APPLICATION_RETRIES=1
        DEPENDENT_LU_COMPRESSION=0
        DEPENDENT_LU_ENCRYPTION=OPTIONAL
        DEST_ADDRESS=40003745000104                                      (18)
        DISABLE_REMOTE_ACT=0
        DSPU_SERVICES=NONE
        FQ_ADJACENT_CP_NAME=DESNI000.M88                             (10),(43)
        HPR_LINK_LVL_ERROR=0
        HPR_SUPPORT=0
        INHERIT_PORT_RETRY_PARMS=1
        LIMITED_RESOURCE=NO
        LINK_DEACT_TIMER=0
        LINK_STATION_ROLE=USE_ADAPTER_DEFAULTS
        MAX_ACTIVATION_ATTEMPTS=-1
        MAX_IFRM_RCVD=0
        MAX_SEND_BTU_SIZE=1033                                           (13)
```

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

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

Change when using SNAP-IX, IBM Communications Server for Linux or IBM Communications Server for AIX

When SNAP-IX, IBM Communications Server for Linux or IBM Communications Server for AIX are used instead of TRANSIT, generation parameters must be defined in the syntax of this product.

Generation of SNAP-IX or IBM Communications Server for Linux or IBM Communications Server for AIX

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

[define_node]
cp_alias = P02CGE                                                    (17)
description = SNAP-IX Knoten
fqcp_name = DESNI000.P02CGE                                     (17),(43)
node_type = LEN_NODE
mode_to_cos_map_supp = YES
mds_supported = YES
node_id = <0172000E>                                            (11),(12)
max_locates = 1500
dir_cache_size = 255
max_dir_entries = 0
locate_timeout = 0
reg_with_nn = YES
reg_with_cds = YES
mds_send_alert_q_size = 100
cos_cache_size = 24
tree_cache_size = 40
tree_cache_use_limit = 40
max_tdm_nodes = 0
max_tdm_tgs = 0
max_isr_sessions = 1000
isr_sessions_upper_threshold = 900
isr_sessions_lower_threshold = 800
isr_max_ru_size = 16384
isr_rcv_pac_window = 8
store_endpt_rscvs = NO
store_isr_rscvs = NO
store_dlur_rscvs = NO
cos_table_version = VERSION_0_COS_TABLES
send_term_self = NO
disable_branch_awareness = NO
```

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

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

[define_ethernet_port]
port_name = ETSAP04                                                        (16)
description = ""
dlc_name = ETHER0
port_type = PORT_SATF
port_number = 0
max_rcv_btu_size = 1033                                                    (13)
tot_link_act_lim = 64
inb_link_act_lim = 0
out_link_act_lim = 0
ls_role = LS_NEG
implicit_dspu_services = NONE
implicit_dspu_template = ""
implicit_ls_limit = 0
act_xid_exchange_limit = 9
nonact_xid_exchange_limit = 5
ls_xmit_rcv_cap = LS_TWS
max_ifrm_rcvd = 7
target_pacing_count = 7
max_send_btu_size = 1033                                                   (13)
mac_address = <000000000000>
lsap_address = 0x04
implicit_cp_cp_sess_support = NO
implicit_limited_resource = NO
implicit_deact_timer = 30
implicit_hpr_support = NO
implicit_link_lvl_error = NO
implicit_uplink_to_en = NO
effect_cap = 3993600
connect_cost = 0
byte_cost = 0
security = SEC_NONSECURE
```

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

[define_ethernet_ls]
ls_name = M88
description = IBM-Host Muenchen
port_name = ETSAP04                                                    (16)
adj_cp_name = DESNI000.M88                                        (10),(43)
adj_cp_type = END_NODE
mac_address = <400037450001>                                           (18)
lsap_address = 0x04
auto_act_supp = NO
tg_number = 0
limited_resource = NO
solicit_sscp_sessions = NO
pu_name = <0000000000000000>
disable_remote_act = NO
default_nn_server = NO
dspu_services = NONE
dspu_name = <0000000000000000>
dlus_name = <0000000000000000000000000000000000>
bkup_dlus_name = <0000000000000000000000000000000000>
hpr_supported = NO
hpr_link_lvl_error = NO
link_deact_timer = 30
use_default_tg_chars = YES
ls_attributes = SNA
local_node_id = <00000000>
cp_cp_sess_support = NO
effect_cap = 3993600
connect_cost = 0
byte_cost = 0
security = SEC_NONSECURE
```

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

[define_partner_lu]
plu_alias = ""
description = (Auto defined - default LU)
fqplu_name = DESNI000.M88                                        (10),(43)
plu_un_name = <0000000000000000>
parallel_sess_supp = YES
max_mc_ll_send_size = 0
conv_security_ver = NO

[define_partner_lu]
plu_alias = CICST6                                                    (20)
description = ""
fqplu_name = DESNI000.A9CICST6                                    (3),(43)
plu_un_name = A9CICST6                                                 (3)
parallel_sess_supp = YES
max_mc_ll_send_size = 0
conv_security_ver = NO
```

```
[define_local_lu]
lu_alias = T02CGEI4                                               (19)
list_name = ""
description = ""
lu_name = T02CGEI4                                                 (4)
lu_session_limit = 100                                             (6)
pu_name = <0000000000000000>
nau_address = 0
default_pool = NO
syncpt_support = YES
lu_attributes = NONE
sscp_id = 0
disable = NO
sys_name = ""
timeout = 60
back_level = NO

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

[define_directory_entry]
resource_name = DESNI000.M88                                 (10),(43)
resource_type = ENCP_RESOURCE
description = (Auto defined - remote node)
parent_name = <00000000000000000000000000000000000>
parent_type = ENCP_RESOURCE

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

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

Parameters that must match

The essential parameters in this example are:

Parameter	CICS host	openUTM-LU62 UNIX system	openUTM end system
LU name	DESNI000.A9CICST6	DESNI000.T02CGEI4	
LU alias name	CICST6	T02CGEI4	
CP name	DESNI000.M88	DESNI000.P02CGE	
IDBLK/IDNUM		017 2000E	
MAC address	400037450001	40002222000E	
MODE name	MODDIS89		
IP address		123.45.67.90	123.45.67.89
T selector		SMP22804	SMP22800
Port number		22804	22800
AP title		(1,2,3)	(1,2,4)
AE qualifier		1	1
APPL context		UDTSEC	

Detailed explanation:

(1) If a CICS program wants to address a UTM application, then it must use the connection name defined in CICS in the SYSID parameter of the EXEC CICS ALLOCATE command. Additionally, a session must be defined in CICS that refers to this connection.

(2) Secondly, the CICS program must specify the transaction code of the UTM application program in the EXEC CICS CONNECT PROCESS command in the PROCNAME parameter.

(3) The APPLID name generated for CICS is the LU name of the CICS. It must be specified together with the network ID generated in VTAM in the TRANSIT generation for XRLU or in the definition of the partner LU in SNAP-IX or the IBM Communications Server generation.

(4) The LU name of the substitute LU must be specified in the LU macro in VTAM, in the connection definition in CICS, in the XLU statement in TRANSIT, and in the definition of the local LU in SNAP-IX or the IBM Communications Server. In TRANSIT, the network ID must also be specified here. If you are using SNAP-IX or the IBM Communications Server and you wish to work with transaction management, syncpoint support must be generated in SNAP-IX or the IBM Communications Server in the definition of the local LU.

When Enterprise Extender is used, the definition of the LU name in VTAM is not required.

(5) A mode name must be agreed between VTAM, CICS, and the system on which openUTM-LU62 runs. This mode name must be specified in VTAM, in the connection definition of CICS, in the XMODE and XLU statements in TRANSIT (or in the MODE definition of SNAP-IX or the IBM Communications Server) and in the openUTM-LU62 generation.

(6) The number of LU6.2 sessions between CICS and openUTM-LU62 and the maximum number of parallel connections that may be established between openUTM-LU62 and the UTM application must be specified. These two values should be identical to avoid deadlock situations. The maximum number of sessions must be entered for the session definition in CICS, the XMODE statements in TRANSIT, and the MODE definition of SNAP-IX or the IBM Communications Server. The maximum number of parallel connections must be specified in the openUTM-LU62 generation and the openUTM generation using the parameter ASSOCIATIONS in both cases.

In TRANSIT-SERVER, a session limit must be specified for XLU. This must also be specified in the definition of the local LU in SNAP-IX or the IBM Communications Server. This session limit should be at least 2 sessions greater than the maximum number of sessions specified above.

(7) The maximum RU size must be agreed between CICS, VTAM, and TRANSIT (or SNAP-IX or the IBM Communications Server). It must be specified for CICS in the session definition and for VTAM, TRANSIT, SNAP-IX and the IBM Communications Server in the mode definition. You should the default value 4096 here whenever possible (see also the section “DEFINE SESSIONS” on page 80).

(8) If the UTM application is to initiate a CICS program, then this program must be informed of CICS in a program definition. Additionally, a transaction definition must refer to the program name in CICS.

(9) The CICS transaction code assigned to this CICS program must be specified in a transaction definition by CICS. In order for the UTM application to be able to access the program, an LTAC statement must appear in the openUTM generation with the CICS transaction code used as an RTAC parameter.

(10) In the VTAM start options, the SSCPNAME parameter is used to generate the CP name of the host system. This CP name must also be generated in SNAP-IX or the IBM Communications Server, but need not be specified in TRANSIT.

(11), (12)
If the SNA connection is implemented via ethernet, then the IDBLK and IDNUM parameters must be defined by VTAM and TRANSIT (or SNAP-IX or the IBM Communications Server).

(13) The maximum frame size in communication with SNA via Ethernet must also be agreed between VTAM and TRANSIT (or SNAP-IX or the IBM Communications Server. If the generation parameters do not match, the two partners sometimes agree on the lesser of the two values.

(14) The MAC address of the system running openUTM-LU62 must be specified in VTAM.

(15) The name of a mode table is used in the VTAM logon mode table. This name must be specified for the mode definition and for the LU definition.

(16) A link name must be assigned to the SNA ethernet connection within TRANSIT. This name must be specified in the XLINK and XPU statements. In SNAP-IX or the IBM Communications Server, the port name is used instead.

(17) A PU name must be assigned to the partner system on which the CICS runs within TRANSIT. The name must be specified in the XPU and XRLU statements. You usually select a name that is identical to the PU name in VTAM. In the case of SNAP-IX and IBM Communications Server, the PU name must be entered as the CP name for the node definition.

(18) The MAC address used to reach the SNA host must be specified in TRANSIT or SNAP-IX or the IBM Communications Server.

(19) The substitute LU receives an LU alias name in TRANSIT or SNAP-IX or the IBM Communications Server. This name must not be identical to the LU name defined for (4). It must be specified in the openUTM-LU62 generation.

(20) The CICS system receives an LU alias name in TRANSIT or SNAP-IX or the IBM Communications Server. This name must not be identical to the LU name defined for (3). It must be specified for TRANSIT in the XLU and XRLU statements, for SNAP-IX or the IBM Communications Server in the definition of the partner LU, and for the openUTM-LU62 generation in the REM-LU-ALIAS parameter.

(21) Only when using TNSX.
The substitute AE of openUTM-LU62 must be defined in the openUTM-LU62 generation and in the TNSX.

(22), (23)
An application process title and an application entity qualifier must be specified for the substitute AE. Both are to be specified in the openUTM-LU62 generation and in the openUTM generation.

(24) Only when using TNSX.
The AE used by the UTM application must be defined in the openUTM-LU62 generation and in the TNSX.

(25), (26)
An application process title and an application entity qualifier must be specified for the UTM application. Both are to be specified in the openUTM-LU62 generation and in the openUTM generation. The UTM application and the substitute AE must have different application process titles. The application entity qualifier can be identical in both, however.

(27) An application context must be defined in openUTM-LU62 and in the UTM application. It is recommended to use the default value UDTSEC for this purpose. If UDTAC is used, in the case of TRANSIT the third and fourth parameters for PAIR_EXT must be set to NONE. When UDTAC is used, no distributed transactions are possible. Other values are not permissible.

(28) The parallel connections available between openUTM-LU62 and the UTM application must be separated into contention winner and contention loser connections by the generation. When added together, the CONTWIN values from the openUTM-LU62 generation and openUTM generation should result in the generated number of parallel connections.

(29), (30)
The T-selector and the port number of the UTM application's access point must also be entered in TNSX entry or directly in the openUTM-LU62 generation (24). This establishes the connection between openUTM-LU62 and the UTM application via CMX.

(31), (32)
According to the ISO protocol convention, an S-selector and a P-selector must be assigned for the AE used by the UTM application. The S-selectors and P-selectors must be specified in the TNSX and in the openUTM generation. It is recommended to use empty S-selectors and P-selectors.
If no TNSX is used S- and P-selector are always empty.

(33) The host name of the system running the UTM application must be entered

- In TNSX as IP address for the name used in the openUTM-LU62 generation for REM-AE or
- directly in the openUTM-LU62 generation as host name in DNS (or in /etc/hosts).

If openUTM-LU62 and the UTM application run on the same computer, you can use localhost here.

(34), (35)
According to the ISO protocol convention, an S-selector and a P-selector must be assigned for the substitute AE. The S-selectors and P-selectors must be specified in the TNSX and in the openUTM generation. It is recommended to use empty S-selectors and P-selectors.
If no TNSX is used S- and P-selector are always empty.

(36) The T-selector of the substitute AE must be entered

- in the TNSX or
- directly in the openUTM-LU62 generation and
- in the OSI-CON statement of the UTM generation.

(37) The port number of the substitute AE must be entered

- in the TNSX or
- directly in the openUTM-LU62 generation and
- and in the OSI-CON statement of the UTM generation.

A separate port number must be assigned for each instance of openUTM-LU62.

(38) In the OSI-CON statement of the UTM generation, the name of the computer on which openUTM-LU62 is running must be entered. On UNIX systems, this is generally the DNS host name (e.g. *local* for your own computer), on BS2000/OSD the BCAM name.

(39) The IP address of the computer on which openUTM-LU62 is running is generally entered in DNS. When using openUTM under BS2000/OSD, the IP address is entered using the BCIN command.

(40) The AE used by the UTM application receives a symbolic name in the openUTM generation. This name appears at two different locations in the openUTM generation.

(41) The substitute AE, and therefore the CICS system, receive a symbolic name in openUTM, the LPAP name. This name is generated in the openUTM generation with an OSI-LPAP statement. The OSI-CON statement must refer to this name. If a CICS program is to be addressed by the UTM application, then this name must also be specified in an LTAC statement or in the UTM program in the APRO call.

(42) If a CICS program is to be addressed in the UTM application, then the openUTM generation must contain a corresponding LTAC statement that refers to the CICS system and the transaction code there using the parameters LPAP and RTAC. The UTM application program must use this LTAC name in APRO. In addition, the LPAP name can also be specified with the APRO call.

(43) The network ID of the SNA network is defined in the starting parameters of VTAM. This name must also be specified at several points for TRANSIT or SNAP-IX or IBM Communications Server.

4.2 Programming an openUTM-CICS interconnection

LU6.2 communication between CICS and openUTM can only be programmed for CICS with the help of DTP (Distributed Transaction Processing). Asynchronous processing using START and RETRIEVE is not possible with LU6.2.

CICS programs are usually programmed with CICS commands of the type:

```
EXEC CICS command option
```

The most important CICS commands for LU6.2 communication are described in the following sections. The CPI-C calls can also be used instead of the usual CICS commands. The use of this interface is also described starting on page 153 because CPI-C is also an important program interface for LU6.2 communication in a non-CICS environment.

The following commands are available for CICS for LU6.2 communication:

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

There are several variants for many of these commands. The LU6.2 variant is always designated using APPC in the CICS manuals, for example SEND (APPC).

Similar to CICS, there are several program interfaces to choose from for openUTM. The following description usually assumes that the UTM applications are programmed using KDCS. However, CPI-C is also discussed further below.

4.2.1 CICS commands for CICS job-submitter services

ALLOCATE

An ACICS job-submitting service attempts to allocate a session for a job-receiving service with this command.

The command has the following syntax.

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

SYSID(name)
: Designates the name of the remote UTM application as it has been defined in a connection definition.

PROFILE(name)
: Designates a specific profile for communication with the partner. Profiles are specified using CICS definitions. The PARTNER parameter can also be specified instead of SYSID and PROFILE. This partner must then be created using a CICS definition.

NOQUEUE
: This means that the program should not wait if there is no session available immediately.

CICS returns a CONVID in the EIBRSRCE field after a successful ALLOCATE command. This CONVID must be specified in all the following calls that refer to the session allocated here.

CONNECT PROCESS

The LU6.2 starts a conversation with this command.

The command has the following syntax:

```
EXEC CICS CONNECT PROCESS CONVID(name)
                          PROCNAME(data_area)
                          PROCLENGTH(data_value)
                          SYNCLEVEL(data_value)
```

CONVID(name)
Designates the session allocated by ALLOCATE.

PROCNAME(data_area) and PROCLENGTH(data_value)
Designates the UTM transaction code.

SYNCLEVEL(data_value)
Designates the synchronization level of the LU6.2 protocol. The parameter corresponds to the KCOF field in the UTM call APRO.
0 corresponds to KCOF=B (base functions),
1 corresponds to KCOF=H (handshake functions)
2 corresponds to KCOF=C (commit functions).

It is not possible to use the PIPLIST and PIPLENGTH parameters to send PIP data (program initialization parameter data).

SEND

This command sends a dialog message segment.

The command has the following syntax:

```
EXEC CICS SEND CONVID(name)
               FROM(data_area)
               LENGTH(data_value)
               [ { INVITE | LAST } ]
               [ { CONFIRM | WAIT } ]
```

CONVID(name)
: Designates the session allocated by ALLOCATE.

FROM(data_area) and LENGTH(data_value)
: Designates the address and the length of the message.

INVITE
: Initiates the transmission of the send authorization (change direction) to the partner when the message is sent. The arrival of the send authorization is interpreted by openUTM as the end of the dialog message from the partner. The UTM application program is able to recognize this when it receives the return code 10Z from the next MGET.

LAST
: Initiates the termination of the service after the message has been sent. The UTM application program recognizes this when KCVGST=C.

CONFIRM
: Initiates the request to send a processing acknowledge from the UTM application program after the message has been sent. The UTM application program recognizes this when KCRMGT=H. The parameter may only be specified for sync-levels 1 or 2.

WAIT
: Only allows the CICS command to return after CICS has actually sent the message. If a SYNCPOINT command follows the SEND command, then you should not use this parameter.

RECEIVE

This command is used to receive a message sent by a UTM service.

The command has the following syntax:

```
EXEC CICS RECEIVE CONVID(name)
                  INTO(data_area)
                  LENGTH(data_value)
                  [ MAXLENGTH(data_value) ]
                  [ NOTRUNCATE ]
```

CONVID(name)
: Designates the session allocated by ALLOCATE.

INTO(data_area) and MAXLENGTH(data_value)
: Designates the address and length of the buffer used to input the message.

LENGTH(data_value)
: CICS returns the length of the message received in LENGTH.

NOTRUNCATE
: The rest of the message is not transmitted unless it is shorter than MAXLENGTH. If NOTRUNCATE is not specified, a message with a length of up to MAXLENGTH is delivered. The rest is rejected. If NOTRUNCATE is specified, other parts can be requested with RECEIVE.

CONVERSE

This command causes a message to be sent to the partner and the program to wait for the reply. This command has the same effect as the command list SEND INVITE, RECEIVE.

The command has the following syntax:

```
EXEC CICS CONVERSE CONVID(name)
                   FROM(data_area)
                   FROMLENGTH(data_value)
                   INTO(data_area)
                   TOLENGTH(data_value)
                   [ MAXLENGTH(data_value) ]
                   [ NOTRUNCATE ]
```

The parameters mean the same as for the SEND and RECEIVE commands.

ISSUE ABEND

The service for the UTM application program is terminated abnormally with this command. The call corresponds to the UTM call CTRL AB.

The command has the following syntax:

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

CONVID(name)
: Designates the session allocated by ALLOCATE.

ISSUE CONFIRMATION

A positive processing acknowledge is sent to the UTM application program with this command. The command is only allowed when sync-level 1 or 2 has been set and the UTM application program has requested a processing acknowledge. The UTM application program recognizes the arrival of the processing acknowledge when KCRMGT=C.

The command has the following syntax:

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

CONVID(name)
: Designates the session allocated by ALLOCATE.

ISSUE ERROR

An error message or a negative processing acknowledge is sent to the UTM application program with this command. The UTM application program receives KCRMGT=E for MGET.

The command has the following syntax:

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

CONVID(name)
: Designates the session allocated by ALLOCATE.

SYNCPOINT

All job-receiver services are requested to initiate the end of the transaction with this command. The command is only allowed for sync-level 2. The UTM application program recognizes this when KCTAST=P for MGET.

The command has the following syntax:

```
EXEC CICS SYNCPOINT
```

SYNCPOINT ROLLBACK

The transaction is rolled back with this command. The command is only allowed for sync-level 2. The UTM application program recognizes the rolling back of the transaction when KCTAST=R for MGET.

The command has the following syntax:

```
EXEC CICS SYNCPOINT ROLLBACK
```

ISSUE PREPARE

The partners are requested to initiate the end of the transaction with this command. The command is only allowed for sync-level 2.

The command has the following syntax:

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

CONVID(name)
: Designates the session allocated by ALLOCATE.

ISSUE PREPARE is only different from SYNCPOINT in that only one of several possible job-receiving services is requested to initiate the end of transaction. The ISSUE PREPARE must be followed later on by SYNCPOINT or SYNCPOINT ROLLBACK.

For CICS-CICS interconnection, the job-receiving service can still issue any errors after ISSUE PREPARE using ISSUE ERROR and allow the job-submitter to decide whether it should execute a SYNCPOINT or a SYNCPOINT ROLLBACK. A UTM application, however, cannot issue a call corresponding to ISSUE ERROR, rather it can only reply with a PEND variant. The decision on whether to end the transaction or roll back the transaction is then made by the UTM application program after ISSUE PREPARE has been called.

FREE

The command releases a session allocated by ALLOCATE.

The command has the following syntax:

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

CONVID(name)
: Designates the session allocated by ALLOCATE.

WAIT CONVID

The command ensures that data created before using CONNECT PROCESS or SEND are actually sent.

The command has the following syntax:

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

CONVID(name)
: Designates the session allocated by ALLOCATE.

ISSUE SIGNAL

The command sends a request to the UTM application program to release its send authorization. This command does not affect a connection to openUTM because the UTM application program is not informed of the arrival of such a request.

4.2.2 CICS commands for CICS job-receiving services

EXTRACT PROCESS

The command is used to query the parameters set in the UTM application program using APRO.

The command has the following syntax:

```
EXEC CICS EXTRACT PROCESS [ PROCNAME(data_area) ]
                          [ MAXPROCLENGTH(data_value) ]
                          [ PROCLENGTH(data_value) ]
                          [ SYNCLEVEL(data_area) ]
```

PROCNAME(data_area), MAXPROCLENGTH(data_value) and PROCLENGTH(data_value)
: Queries the CICS transaction code set by the UTM application program for APRO in the KCRN parameter.

SYNCLEVEL(data_area)
: Queries the synchronization level of the LU6.2 protocol set by the UTM application program for APRO in the KCOF parameter.
0 corresponds to KCOF=B (base functions),
1 corresponds to KCOF=H (handshake functions)
2 corresponds to KCOF=C (commit functions)

RECEIVE

This command is used to receive a message sent by a UTM job-submitting service.

The syntax and meaning are the same as for job-submitting services. However, CONVID does not have to be specified when reading the message from the job submitter.

SEND

This command sends a dialog message segment.

The syntax and meaning are the same as for job-submitting services. However, CONVID does not have to be specified when sending to the job submitter.

CONVERSE

This command causes a message to be sent to the partner sent and the program to wait for the reply. This command has the same effect as the command list SEND INVITE, RECEIVE.

The syntax and meaning are the same as for job-submitting services.

ISSUE ABEND

The service for the UTM application program is terminated abnormally with this command.

The syntax and meaning are the same as for job-submitting services. However, CONVID does not have to be specified for ISSUE ABEND to the UTM job-submitting service.

ISSUE CONFIRMATION

A positive processing acknowledge is sent to the UTM application program with this command.

The syntax and meaning are the same as for job-submitting services. However, CONVID does not have to be specified for ISSUE CONFIRMATION to the UTM job-submitting service.

ISSUE ERROR

An error message or a negative processing acknowledge is sent to the UTM application program with this command.

If the CICS job-receiving program calls ISSUE ERROR when it is requested to initiate the end of transaction by the UTM application program, then this leads to the rolling back of the transaction.

The syntax and meaning are the same as for job-submitting services. However, CONVID does not have to be specified for ISSUE ERROR to the UTM job-submitting service.

SYNCPOINT

The end of transaction is initiated upon the request of the job-submitter with this command.

It is important for an interconnection to openUTM that a CICS program only uses SYNCPOINT when the UTM application program has requested it to do so before it has requested the initiation of the end of the transaction. The CICS program can recognize if it has been requested to initiate the end of transaction in the STATE parameter or in the EIBSYNC field of every CICS command. STATE must then assume one of the values SYNCFREE, SYNCRECEIVE or SYNCSEND. EIBSYNC must assume the value X'FF'.

The syntax is the same as for job-submitting services.

SYNCPOINT ROLLBACK

The transaction is rolled back with this command.

The syntax and meaning are the same as for job-submitting services.

FREE

The command releases the session to the job-submitting service.

The syntax and meaning are the same as for job-submitting services. However, CONVID may not be specified for a job-submitting service.

WAIT CONVID

The command ensures that data sent before using SEND is actually sent.

The syntax and meaning are the same as for job-submitting services.

ISSUE SIGNAL

As described for job-submitting services, this command is not to be used.

ISSUE PREPARE

The command must not be issued in a job-submitting service.

RETURN

the RETURN command triggers the sending of any messages not yet sent by sending a security request to the partner and also triggers the release of the session afterwards. The RETURN command may only be used when the openUTM job-submitting service has already issued a CTRL PE or PEND FI (EIBFREE indicator in the EIB) or after the CICS program has abnormally terminated the service using ISSUE-ABEND.

The command has the following syntax:

```
EXEC CICS RETURN
```

4.2.3 CICS programming hints

There are also several restrictions in the scope of the functionality for openUTM-CICS interconnection as compared to the functionality normally available for LU6.2 interconnections.

1. The LU6.2 communication between CICS and openUTM can only be programmed with the help of DTP (distributed transaction processing) with CICS. Asynchronous processing by means of START and RETRIEVE or a distributed program link by means of EXEC CICS LINK is not possible with openUTM-LU62.
2. A CICS application program as the job submitter can never use SYNCPOINT or ISSUE PREPARE independently. It can only do this when requested to do so by the UTM application program.
3. Basic conversation is not possible. Basic conversation is programmed for CICS using commands that begin with GDS, such as EXEC CICS GDS ALLOCATE, for example.
4. If the CICS job-receiving program calls ISSUE ERROR when it has been requested to initiate the end of the transaction by the UTM application program, then the transaction is rolled back.
5. PIP data cannot be used for CONNECT Process. The data is lost.
6. It is not possible to use different mode names for different connections to the same partner. The mode name is set in the session definition for CICS/ESA V4.1 and can therefore be selected implicitly in the program interface via the SYSID parameter in the ALLOCATE call.
7. If an LU6.2 conversation to openUTM-LU62 cannot be opened because openUTM-LU62 cannot establish the corresponding connection to the UTM application, then CICS does not receive a detailed rejection message. The detailed rejection messages can only be found in the openUTM-LU62 protocol file.
8. If the generation parameter UTM-ASYNC=YES is set, openUTM-LU62 LU6.2 always establishes conversations with sync-level 1 or 2, never with sync-level 0. The sync-level of an incoming conversation can be requested by means of EXTRACT PROCESS using the CICS API.

4.2.4 Comparison with KDCS calls

We will compare the CICS command semantics with those of the KDCS calls in the following.

Addressing a remote service with transaction management

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

Addressing a remote service without transaction management

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

or

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

Exchanging messages without a synchronization point

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

Sending a message and requesting the end of a transaction

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

Sending a message and requesting the end of a transaction and a service

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

Exchanging messages and requesting the end of a transaction

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

Rolling back the transaction

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

Rolling back the transaction and cancelling the service

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

Exchanging messages with processing acknowledge

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

Sending a reply with a positive processing acknowledge

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

Sending a reply with a negative processing acknowledge

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

Cancelling the dialogs to the partner

```
ISSUE ABEND                          CTRL AB
```

4.2.5 Examples of openUTM-CICS communication

Starting an openUTM dialog service from within a CICS application program

1. Single-step service with transaction management

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

The “INIT PU” call was used in the UTM application program in the example above because the end of transaction (KCENDTA) and send authorization (KCSEND) can then be checked. If it is not necessary to query these values due to the procedural logic of the program, then the normal INIT can also be used. The KCVGST (service status) and KCTAST (transaction status) indicators were abbreviated to VGST and TAST, respectively, for MGET. The SYNCPOINT command in the CICS application program only returns after the UTM application program has called PEND FI. This is represented by the line containing “=NORMAL“.

RequestCommit and Committed are components of the LU6.2 protocol. If the CICS application program has more than one job-receiver, then it sometimes uses a two-phase protocol with the protocol elements Prepare, RequestCommit, Committed and Forget.

The ISSUE PREPARE command can still be called in the CICS application program before SYNCPOINT. This has certain advantages for a PEND ER/FR in the UTM application program. See below for more information.

2. No data from the UTM application program is required in the CICS application program and with transaction management.

If the procedural logic does not require that the UTM application program called send data back to the CICS application program, then communication can be programmed more easily as shown in the following example.

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

3. Two transactions in a service, first variant

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

4. Two transactions in a service, second variant

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

This variant is different from the first in that the reply message from the UTM application program is sent from within the first transaction. The reply message is sent in the second transaction for the first variant. The second variant is more difficult to program and produces more messages than the first variant. However, it has the advantage that any eventual database changes bound to the SYNCPOINT are only permanently written by the UTM application program after the message has been received.

5. PEND ER/FR in the UTM application program

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

In the case above, a PEND ER or PEND FR from UTM results in the cancellation of the CICS job-submitting program with Abend ASP3. This Abend cannot be intercepted by the CICS application program. If you want to avoid this, then the ISSUE PREPARE command must be used in the CICS application program:

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

6. PEND RS in the UTM application program

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

However, a PEND RS in the UTM application program only behaves as described in the example if the service has already reached a synchronization point beforehand. A PEND RS in the first dialog step has the same effect as a PEND FR.

7. One-step service without transaction management

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

8. Processing acknowledge without transaction management

```
--->
    RECEIVE Client
    ALLOCATE
    CONNECT PROCESS CONVID
        SYNCLEVEL(1)
    SEND INVITE CONFIRM
        CONVID
                            -- FMH5+data ----->
                                                INIT PU ENDTA=0,SEND=Y
                                                MGET    VGST=0,TAST=U
                                                MGET    KCRMGT=H
                                                MPUT NE
                                                PEND FI
                            <- data+CEB -------
      =OK EIBERR=X'00'
    RECEIVE CONVID
    SEND Client
<---
    RETURN
```

9. Negative processing acknowledge without transaction management

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

Starting an openUTM asynchronous service from a CICS application program

10. With transaction management

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

openUTM sends the acknowledge message immediately after it has received and stored the queued job. It may take some time before the queued job is executed depending on the availability of system resources.

It is not possible to use EXEC CICS START to start a queued job for LU6.2.

11. Without transaction management

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

By sacrificing transaction management, the application program is subjected to the following disadvantages:

– The sending of queued messages in the CICS program cannot be linked automatically to other processes in the application program. For instance, a queued message may thus be sent to the UTM application despite a failed local database update.

– If the acknowledgment is lost due to a communication error, the queued message may be sent to the UTM application several times.

Starting a CICS dialog service from a UTM application program

12. Single-step service with transaction management, first variant

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

13. Single-step service with transaction management, second variant

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

In example 13, the 3rd program unit sees whether CICS has issued a commit or a rollback.

In example 12, the program unit does not learn any more about commit or rollback in CICS. This is solely under the control of the UTM system code.

14. No data from the CICS application program is required in the UTM application program and with transaction management.

If the procedural logic does not require the CICS application program called to send data back to the UTM application program, then communication can be programmed more easily as shown in the following example:

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

15. Two transactions in a service, first variant

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

If no data is sent at the second `Prepare`, there is no need for the subsequent UTM calls, as in examples 12 and 13:

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

16. Two transactions in a service, second variant

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=C
    MPUT NE KCRN=>A
    PEND KP
                            -- FMH5+data ----->
                                                RECEIVE STATE(SEND)
                                                SEND
                            <- Data -----------
    INIT
    MGET NT KCRN=>A
        VGST=O,TAST=O
    MPUT NE KCRN=>A KCLM=0
    CTRL PR KCRN=>A
    PEND KP
                            -- Prepare -------->
                                                RECEIVE STATE(SYNCRECEIVE)
                                                SYNCPOINT
                            <- RequestCommit ---
    INIT
    MGET NT KCRN=>A
        VGST=O,TAST=P
    MPUT NE Client
    PEND RE
                            -- Committed ------>
                            <- Forget ----------
                                                =NORMAL
<---
--->
    INIT
    MGET    Client
    MPUT NE KCRN=>A
    CTRL PE KCRN=>A
    PEND KP
                            -- Date+Prepare --->
                                                RECEIVE STATE(SYNCFREE)
                                                SYNCPOINT
                            <- RequestCommit ---
```

Continued on the next page

Continued

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

As in examples 12 and 13, some UTM calls are not required if you do without the roll-back/commit information from CICS in the program unit. The 2nd and 3rd sections of the program are then reduced to the following sequence:

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

This variant is different from the first in that the reply message from the CICS application program is sent from within the first transaction. The reply message is sent in the second transaction in the first variant. The second variant is more difficult to program and produces more messages than the first variant. However, it has the advantage that any eventual database changes bound to the PEND RE are only permanently written by the CICS application program after the message has been received.

17. The CICS application program rolls back the transaction and cancels the service, first variant

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=C
    MPUT NE KCRN=>A
    PEND KP
                            -- FMH5+data ----->
                                                RECEIVE STATE(SEND)
                                                SYNCPOINT ROLLBACK
                                                ISSUE ABEND
                                                RETURN
                            <- FMH7 ------------
                            -- FMH7-Response -->
    Roll back the transaction
    possibly with message K034 on the client
    If a synchronization point has already been
    reached, then start this synchronization
    point's successor program:
    INIT    KCKNZVG=R
    MGET NT KCRN=>A
        VGST=O,TAST=R
    ...
```

18. The CICS application program rolls back the transaction and cancels the service, second variant

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

In contrast to example 17, the program unit in example 18 receives a message after rollback by CICS.

19. The CICS application program rolls back the transaction, the service continues to run

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=C
    MPUT NE KCRN=>A
    PEND KP
                            -- FMH5+data ----->
                                                RECEIVE STATE(SEND)
                                                SEND
                            <- Data -----------
    INIT
    MGET NT KCRN=>A
        VGST=O,TAST=O
    MPUT NE KCRN=>A KCLM=0
    CTRL PR KCRN=>A
    PEND KP
                            -- Prepare -------->
                                                RECEIVE STATE(SYNCRECEIVE)
                                                SYNCPOINT
                            <- RequestCommit ---
    INIT
    MGET NT KCRN=>A
        VGST=O,TAST=P
    MPUT NE Client
    PEND RE
                            -- Committed ------>
                            <- Forget ----------
                                                =NORMAL
<---
--->
    INIT
    MGET    Client
    MPUT NE KCRN=>A
    CTRL PE KCRN=>A
    PEND KP
                            -- Prepare -------->
                                                RECEIVE STATE(SYNCFREE)
                                                SYNCPOINT ROLLBACK
                            <- FMH7 ------------
                            -- FMH7-Response -->
                                                =NORMAL
<--- K034-message
--->
```

Continued on the next page

Continued

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

Note that in this case the UTM application program does not receive a status message after the SYNCPOINT ROLLBACK from the job-receiver. openUTM only sends the application program a status message from the job-receiver when the job-receiver was terminated due to the rolling back of the transaction.

In addition to message K034, the message to the client from the last synchronization point is sent to the client again.

As in examples 12 and 13, some UTM calls are not required if you do without the roll-back/commit information from CICS in the program unit. The 2nd and 3rd sections of the program are then reduced to the following sequence:

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

20. One-step service without transaction management

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

21. Processing acknowledge without transaction management

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

22. Negative processing acknowledge without transaction management

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

Starting a CICS asynchronous service from a UTM application program

23. Starting a CICS asynchronous service from a UTM application program without transaction management

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

For this kind of communication, an openUTM-LU62 instance generated with the parameter UTM-ASYNC=YES must be used.

The case shown in example 20 is similar. The key difference is that the CICS program in example 20 can still send data back. On the UTM side, the advantage of the dialog is that the application is notified of the success or failure of the call.

If the CICS program sends a negative processing acknowledge using ISSUE ERROR, then openUTM deletes the job. However, the UTM application program obviously cannot be notified of this fact any more.

24. Starting a CICS asynchronous service from a UTM application program with transaction management

```
    INIT
    APRO AM KCPI=>A KCOF=C
    FPUT NE KCRN=>A
    PEND FI
                            -- FMH5+data+
                               Prepare -------->
                                                RECEIVE STATE(SYNCFREE)
                                                SYNCPOINT
                            <- RequestCommit ---
                            -- Committed ------>
                            <- Forget ----------
                                                =NORMAL
                                                RETURN
```

If the CICS program rolls back the transaction using SYNCPOINT ROLLBACK, then openUTM deletes the job. However, the UTM application program obviously cannot be notified of this fact any more.

The transaction management in example 24 has the following advantages for the application programs compared to example 23, where there is no transaction management:

– The sending of queued messages in the UTM application program is automatically linked to other processes in the application program, e.g. database update.

– If PEND FI is successful, it is guaranteed that the CICS asynchronous process will be started only once.

This case is similar to example 14. The CICS programming is identical. On the UTM side, the dialog case (example 14) has the advantage that the application is notified of the success or failure of the call. The asynchronous case (example 24) is suitable, above all, for time-driven jobs.

4.2.6 Distributed Program Link

In CICS you have the capability to link programs on different systems together using a Distributed Program Link (DPL). A CICS program on one system can call a CICS program as a subroutine on another system using this technique. It uses the CICS call EXEC CICS LINK to do this. It is not possible to open a distributed program link between CICS and openUTM. In a multi-level hierarchy, however, UTM application programs can also be used. The following two examples show such communication. The first example shows the use of EXEC CICS LINK between 2 CICS systems. In the second example the distributed program link is called by a system with a CICS client. The program interface “External Call Interface” (ECI) is used for the CICS client.

1. Distributed program link between two CICS systems

```
    CICS system 1         CICS system 2         openUTM
--->
    RECEIVE Client
    LINK SYSID(...)
        TRANSID(...)
               ---------->
                          ALLOCATE
                          CONNECT PROCESS CONVID
                              SYNCLEVEL(2)
                          SEND INVITE WAIT
                              CONVID
                                     ---------->
                                                INIT PU ENDTA=O,SEND=Y
                                                MGET    VGST=O,TAST=O
                                                MPUT NE
                                                PEND KP
                                     <----------
                          =NORMAL
                          RECEIVE CONVID
                          RETURN
               <----------
    =NORMAL
    SYNCPOINT
               ---------->
                                     ---------->
                                                INIT PU ENDTA=R,SEND=N
                                                MGET    KCRLM=0
                                                        VGST=O,TAST=P
                                                PEND RE
                                     <----------
               <----------
               ---------->
                                     ---------->
                                                INIT PU ENDTA=F,SEND=N
                                                MGET    KCRLM=0
                                                        VGST=O,TAST=P
                                                PEND FI
                                     <----------
               <----------
    =NORMAL
    SEND Client
<---
    RETURN
```

2. Distributed program link for a CICS client

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

CICS_ExternalCall
    (program)
         -------->
                  LINK SYSID(...)
                      TRANSID(...)
                           -------->
                                    ALLOCATE
                                     CONNECT PROCESS
                                        CONVID
                                        SYNCLEVEL(2)
                                    SEND INVITE WAIT
                                        CONVID
                                             -------->
                                                      INIT PU
                                                          ENDTA=O,SEND=Y
                                                      MGET
                                                          VGST=O,TAST=O
                                                      MPUT NE
                                                      PEND KP
                                             <--------
                                    =NORMAL
                                    RECEIVE CONVID
                                    RETURN
                           <--------
                  =NORMAL
         <--------
=ECI_NO_ERROR
CICS_ExternalCall
    (ECI_COMMIT)
         -------->
                           -------->
                                             -------->
                                                      INIT PU
                                                          ENDTA=R,SEND=N
                                                      MGET      KCRLM=0
                                                          VGST=O,TAST=P
                                                      PEND RE
                                             <--------
                           <--------
         <--------
```

Continued on the next page

Continued

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

         -------->
                           -------->
                                             -------->
                                                      INIT PU
                                                          ENDTA=F,SEND=N
                                                      MGET      KCRLM=0
                                                          VGST=0,TAST=P
                                                      PEND FI
                                             <--------
                           <--------
         <--------
=ECI_NO_ERROR
```

4.2.7 openUTM programming hints

The following points are to be considered when programming UTM applications if the applications are to communicate with LU6.2 partners:

1. If a UTM application program requests the LU6.2 partner as a job-submitter to initiate the end of a transaction using CTRL PR or CTRL PE, then it cannot receive any more data from the partner, regardless of whether or not it has called an MPUT before the CTRL. An additional dialog step is required in many cases due to this limitation.
2. User data can only be transported in the UDT format. This means that the KCMF field must always contain a space character for MPUT, FPUT and DPUT on an LU6.2 partner.
3. When using TRANSIT, only user data up to a length of 32763 bytes can be sent to CICS by UTM. The maximum value that KCLM can assume is therefore 32763 for MPUT, FPUT and DPUT on an LU6.2 partner.
4. If KCSECTYP=P is specified for APRO, i.e. the UTM application program wants to send the partner a user ID, then a password must also be specified, i.e. KCPWDLTH must be larger than 0. The user ID and password may be no longer than a maximum of 10 bytes long each.
5. If openUTM-LU62 cannot establish the connection desired by a UTM application program to a CICS transaction code, then the UTM application does not receive any detailed rejection justification. The information in the openUTM SYSLOG file usually does not contain enough information for a diagnosis. Detailed reasons for the rejection can only be found in the openUTM-LU62 protocol file.
6. openUTM offers the possibility of generating a maximum wait time for the allocation of a session to the partner (time1) and a maximum wait time for the arrival of a reply from the job-receiver (time2) for the LTAC generation. The default value for time1 is set to 30 seconds and the default value for time2 is infinity. Only the wait time for the allocation a connection between openUTM and openUTM-LU62 is monitored with time1 for openUTM-CICS interconnection. If it is impossible for openUTM-LU62 to establish a connection to CICS, then the wait time for the allocation of this connection must be limited in openUTM-LU62 (using ALLOC-TIME) or using time2.

4.3 Using the CPI-C program interface

The CPI-C program interface can also be used instead of the CICS commands in CICS for LU6.2 communication. Because of the fact that, unlike CICS, the LU6.2 partner is often implemented using the CPI-C program interface for interconnections between openUTM and LU6.2 partners, the examples given above will be shown using the CPI-C program interface for the openUTM-CICS communication. Examples of these kinds of partners are APPC/MVS, IMS and System i5 (previously AS/400).

CPI-C can also be used in openUTM instead of KDCS for OSI-TP communication. If both partners use CPI-C, then this situation is very similar on both sides to an openUTM-openUTM interconnection with CPI-C or an openUTM-OpenCPIC interconnection. Both are described in detail in the openUTM manual “Creating Applications with X/Open Interfaces“.

If transaction-oriented communication via OSI-TP or LU6.2 is to be programmed with CPI-C, then an additional program interface is required in addition to CPI-C to request and confirm a synchronization point and to roll back transactions. The TX program interface is used in openUTM and for X/Open for this purpose. On the other hand, the CPI-RR (Common Programming Interface for Resource Recovery) program interface is usually used with the SRRCMIT and SRRBACK calls for IBM products. For this reason, the openUTM-CICS examples above are reformulated in the following with CPI-C and CPI-RR.

4.3.1 Comparison to KDCS calls

The semantics of CICS commands will be compared with those of the KDCS calls in the following.

Addressing a remote service with transaction management

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

Addressing a remote service without transaction management

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

or

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

Receiving a message in the job-receiver

```
CMACCP                               INIT or INIT PU
CMRCV                                MGET
```

Exchanging messages without a synchronization point

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

Sending a message and requesting the end of a transaction

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

Sending a message and requesting the end of a transaction and a service

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

Exchanging messages and requesting the end of a transaction

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

Rolling back the transaction

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

Rolling back the transaction and cancelling the service

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

Exchanging messages with processing acknowledge

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

Sending a reply with a positive processing acknowledge

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

Sending a reply with a negative processing acknowledge

```
CMSERR                             MPUT EM
```

Cancelling the dialogs to the partner

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

4.3.2 Examples of openUTM-CPIC communication

Starting an openUTM dialog service from a CPIC application program

1. Single-step service with transaction management

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

2. No data from the UTM application program is required in the CPIC application program and with transaction management.

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

3. Two transactions in a service, first variant

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

4. Two transactions in a service, second variant

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

5. PEND ER/FR in the UTM application program

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

6. Call PEND RS in the UTM application program

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

7. One-step service without transaction management

```
CMINIT
CMALLC
CMSEND
CMRCV
                       -- FMH5+data ----->
                                           INIT PU ENDTA=0,SEND=Y
                                           MGET    VGST=0,TAST=U
                                           MPUT NE
                                           PEND FI
                       <- data+CEB -------
=CM_DEALLOCATED_NORMAL
```

8. Processing acknowledge without transaction management

```
CMINIT
CMSSL (CM_CONFIRM)
CMALLC
CMSST (CM_SEND_AND_PREP_TO_RECEIVE)
CMSEND
                       -- FMH5+data ----->
                                           INIT PU ENDTA=0,SEND=Y
                                           MGET    VGST=0,TAST=U
                                           MGET    KCRMGT=H
                                           MPUT NE
                                           PEND FI
                       <- data+CEB -------
=CM_OK
CMRCV
=CM_DEALLOCATED_NORMAL
```

9. Negative processing acknowledge without transaction management

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

Starting an openUTM asynchronous service from a CPIC application program

10. With transaction management

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

openUTM sends the acknowledge message immediately after it has received and stored the queued job. It may take some time before the queued job is executed depending on the availability of system resources.

11. Without transaction management

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

openUTM sends the acknowledge message immediately after it has received and stored the queued job. It may take some time before the queued job is executed depending on the availability of system resources.

For information on the advantages of transaction management for asynchronous processes, see page 134.

Starting a CPIC dialog service from a UTM application program

12. Single-step service with transaction management, first variant

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=C
    MPUT NE KCRN=>A
    PEND KP
                            -- FMH5+Data ------>
                                                CMACCP
                                                CMRCV
                                                -> CM_SEND_RECEIVED
                                                CMSEND
                            <- Data ------------
    INIT
    MGET NT KCRN=>A
        VGST=0,TAST=0
    MPUT NE Client
    PEND FI
                            -- Prepare -------->
                                                CMRCV
                                                -> CM_TAKE_COMMIT_
                                                      DEALLOCATE
                                                SRRCMIT
                            <- RequestCommit ---
    (UTM system code)
                            -- Committed ------>
                            <- Forget ----------
                                                =RR_OK
<---
```

13. Single-step service with transaction management, second variant

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=C
    MPUT NE KCRN=>A
    PEND KP
                            -- FMH5+data ----->
                                                CMACCP
                                                CMRCV
                                                -> CM_SEND_RECEIVED
                                                CMSEND
                            <- Data -----------
    INIT
    MGET NT KCRN=>A
        VGST=O,TAST=O
    MPUT NE KCRN=>A KCLM=0
    CTRL PE KCRN=>A
    PEND KP
                            -- Prepare -------->
                                                CMRCV
                                                -> CM_TAKE_COMMIT_
                                                      DEALLOCATE
                                                SRRCMIT
                            <- RequestCommit ---
    INIT
    MGET NT KCRN=>A
        VGST=C,TAST=P
    MPUT NE Client
    PEND FI
                            -- Committed ------>
                            <- Forget ----------
                                                =RR_OK
<---
```

In example 13, the 3rd program unit sees whether CPI-C has issued a commit or a rollback.

In example 12, the program unit learns nothing further about commit or rollback in CPI-C. This takes place solely under the control of UTM system code.

14. No data from the CPIC application program is required in the UTM application program and with transaction management.

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

15. Two transactions in a service, first variant

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

If no data is sent at the second `Prepare`, as in examples 12 and 13 the subsequent UTM calls are not required:

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

16. Two transactions in a service, second variant

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=C
    MPUT NE KCRN=>A
    PEND KP
                            -- FMH5+data ----->
                                                CMACCP
                                                CMRCV
                                                -> CM_SEND_RECEIVED
                                                CMSEND
                            <- Data -----------
    INIT
    MGET NT KCRN=>A
        VGST=O,TAST=O
    MPUT NE KCRN=>A KCLM=0
    CTRL PR KCRN=>A
    PEND KP
                            -- Prepare -------->
                                                CMRCV
                                                -> CM_TAKE_COMMIT
                                                SRRCMIT
                            <- RequestCommit ---
    INIT
    MGET NT KCRN=>A
        VGST=O,TAST=P
    MPUT NE Client
    PEND RE
                            -- Committed ------>
                            <- Forget ----------
                                                =RR_OK
<---
--->
    INIT
    MGET    Client
    MPUT NE KCRN=>A
    CTRL PE KCRN=>A
    PEND KP
                            -- Data+Prepare --->
                                                CMRCV
                                                -> CM_TAKE_COMMIT_
                                                      DEALLOCATE
                                                SRRCMIT
```

Continued on the next page

Continued

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

As in examples 12 and 13, some UTM calls are not required if you do without the rollback/commit information from CPI-C in the program unit. The 2nd and 3rd program sections are then reduced to the following sequence:

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

17. The CPIC application program rolls back the transaction and cancels the service, first variant

```
--->
    INIT
    MGET    Client
    APRO DM KCPI=>A KCOF=C
    MPUT NE KCRN=>A
    PEND KP
                           -- FMH5+data ----->
                                                 CMACCP
                                                 CMRCV
                                                 -> CM_SEND_RECEIVED
                                                 SRRBACK
                                                 CMSDT
                                                     (CM_DEALLOCATE_ABEND)
                                                 CMDEAL
                            <- FMH7 ------------
                           -- FMH7-Response -->
    Roll back the transaction
    possibly with message K034 on the client
    If a synchronization point has already been
    reached, then start this synchronization
    point's successor program:
    INIT    KCKNZVG=R
    MGET NT KCRN=>A
        VGST=O,TAST=R
    ...
```

18. The CPI-C application program rolls back the transaction and cancels the service, second variant

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

In contrast to example 17, in example 18 the program unit receives a message back after the transaction is rolled back by CPI-C.

19. The CPIC application program rolls back the transaction, the service continues to run

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

Continued on the next page

Continued

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

Note that the UTM application program does not receive a status message from the job-receiver after the SRRBACK in this case. openUTM only sends a status message from the job-receiver to the application program if the job-receiver has been terminated by the rolling back of the transaction.

In addition to message K034, the message to the client from the last synchronization point is sent to the client again.

As in examples 12 and 13, some UTM calls are not required if you do without the roll-back/commit information from CICS in the program unit. The 2nd and 3rd sections of the program are then reduced to the following sequence:

```
INIT
MGET NT KCRN=>A VGST=0,TAST=0
CTRL PR KCRN=>A
MPUT NE Client
PEND KCOM=RE
```

20. One-step service without transaction management

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

21. Processing acknowledge without transaction management

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

22. Negative processing acknowledge without transaction management

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

Starting a CPIC asynchronous service from a UTM application program

23. Starting a CPIC asynchronous service from a UTM application program without transaction management

```
    INIT
    APRO AM KCPI=>A KCOF=B
    FPUT NE KCRN=>A
    PEND FI
                            -- FMH5+data ----->
                                                CMACCP
                                                CMRCV
                                                 -> CM_DATA
                                                CMRCV
                                                -> CM_CONFIRM_DEALLOC_
                                                      RECEIVED
                                                CMCFMD
                            <- data+CEB -------
```

For this kind of communication, an openUTM-LU62 instance generated with the parameter UTM-ASYNC=YES must be used.

The case shown in example 20 is similar. The key difference is that the CPI-C program in example 20 can still send data back. On the UTM side, the advantage of the dialog is that the application is notified of the success or failure of the call.

If the CPIC program sends a negative processing acknowledge using CMSERR, then openUTM deletes the job. However, the UTM application program obviously cannot be informed of this any more.

24. Starting a CPIC asynchronous service from a UTM application program with transaction management

```
    INIT
    APRO AM KCPI=>A KCOF=C
    FPUT NE KCRN=>A
    PEND FI
                            -- FMH5+data+
                               Prepare -------->
                                                CMACCP
                                                CMRCV
                                                -> CM_TAKE_COMMIT_
                                                      DEALLOCATE
                                                SRRCMIT
                            <- RequestCommit ---
                            -- Committed ------>
                            <- Forget ----------
                                                =RR_OK
                                                RETURN
```

If the CPIC program rolls back the transaction using SRRBACK, then openUTM deletes the job. However, the UTM application program obviously cannot be informed of this any more.

The transaction management in example 24 has the following advantages for the application programs compared to example 23, where there is no transaction management:

– The sending of queued messages in the UTM application program is automatically linked to other processes in the application program, e.g. database update.

– If PEND FI is successful, it is guaranteed that the CICS asynchronous process will be started only once.

This case is similar to example 14. The CPI-C programming is identical. On the UTM side, the dialog case (example 14) has the advantage that the application is notified of the success or failure of the call. The asynchronous case (example 24) is suitable, above all, for time-driven jobs.

5 openUTM-IMS interconnection via LU6.2

This chapter covers generation and programming aspects of openUTM-IMS interconnection via LU6.2. IMS is a transaction monitor and an IBM database. Since version 6, the IMS Transaction Monitor (IMS TM) has been capable of transaction-oriented LU6.2 communication to partner applications. This can be used in an interconnection to openUTM.

5.1 Generating an openUTM-IMS interconnection

5.1.1 IMS startup parameters

If an IMS application is to use LU6.2, the APPC/IMS component must be used for this. APPC/IMS is part of IMS TM. It uses two components of the z/OS operating system: APPC/MVS and RRS/MVS. If an IMS application is to use LU6.2 protocols, the value APPC=Y must be specified in the IMS startup parameters or in the DFSPBxxx member of IMS.PROCLIB. In transaction-oriented communication, RRS=Y must be specified as well.

Example of IMS startup parameters in the DCC procedure:

```
//        PROC RGN=2000K,SOUT=A,DPTY='(14,15)',
//          SYS=,SYS1=,SYS2=,
//          RGSUF=IV1,PARM1=APPC=Y,PARM2=RRS=Y,APPLID1=IMS81CR1,AOIS=R
//IEFPROC EXEC PGM=DFSMVRC0,DPRTY=&DPTY,
//          REGION=&RGN,
//          PARM='CTL,&RGSUF,&PARM1,&PARM2,&APPLID1,&AOIS'
```

APPC/MVS is a subsystem of the z/OS operating system that implements the LU6.2 protocol. RRS/MVS (Resource Recovery Services) is an operating system component that acts as a sync point manager, coordinating the two-phase commit between databases and LU6.2 partners.

5.1.2 Defining the LU name of IMS

For LU6.2 communication, IMS needs an APPLID. This must be different from the APPLID used for other SNA communication. The APPLID serves as an LU name. It must be defined in VTAM. Here is an example of this kind of VTAM definition:

```
IMS      VBUILD TYPE=APPL             APPLICATION MAJOR NODE
IMSAPPL1 APPL ACBNAME=IMSAPPL1,       ACBNAME FOR APPC
         APPC=YES,
         ATNLOSS=ALL,
         DLOGMOD=APPCMODE,
         DMINWNL=5,
         DSESLIM=10,
         MODETAB=LOGMODES,
         PARSESS=YES,
         SECACPT=NONE,
         SRBEXIT=YES,
         SYNCLVL=SYNCPT
```

IMSAPPL1 is the APPLID of IMS. APPCMODE is the MODE name in this example. You use DSESLIM to specify the maximum number of LU6.2 sessions and DMINWNL to specify the minimum number of contention winner sessions. SECACPT controls whether the LU6.2 partner programs have to pass a user ID and possibly a password when setting up a conversation. SYNCLVL=SYNCPT is required for two-phase commit communication.

The APPLID must also be specified for APPC/MVS. This is done by means of an LUADD statement in SYS1.PARMLIB(APPCPMxx):

```
LUADD ACBNAME(IMSAPPL1) BASE SCHED(IMSREG) TPDATA(SYS1.APPCTP)
```

IMSAPPL1 is the APPLID of IMS. The IMS region is specified by means of the SCHED parameter. The value specified for TPDATA is the name of the file containing the TP_Profile definitions.

The IMS system definition is specified by means of a number of Assembler macros. Changes can be made to the system definition at runtime by means of the add-on product ETO (Extended Terminal Option).

5.1.3 Defining IMS transactions

DL/I must be defined as an API. IMS transaction codes implemented with DL/I must be defined in the IMS system definition by means of a TRANSACT macro. In addition, the associated application program must be defined by means of an APPLCTN macro. IMS transaction codes implemented with CPI-C, on the other hand, must be defined in the TP_Profile of APPC/MVS.

Example of an APPLCTN and TRANSACT macro

```
APPLCTN GPSB=IMS1PG1,PGMTYPE=TP
TRANSACT CODE=IMS1TR1,MSGTYPE=(,RESPONSE),INQUIRY=NO,MODE=SNGL
```

IMS1PG1 is the name of the program, and IMS1TR1 is the name of the transaction code.

In multi-step transactions, the parameter SPA must be specified in the TRANSACT macro. The value of this parameter specifies the size of a scratch pad area. It serves to keep data available over a number of dialog steps.

Example of a TP_Profile definition

```
TPADD TPSCHED_EXIT(DFSTPPE0)
           TPNAME(IMS1TPN)
           SYSTEM
           ACTIVE(YES)
           TPSCHED_DELIMITER(##)
              TRANCODE=IMS1SRVR
              CLASS=15
              MAXRGN=20
              RACF=NONE
              CPUTIME=100
              ##
```

IMS1TPN specifies the LU6.2 TP name, which is the name that must be addressed from openUTM. The name IMS1SVR used for the TRANCODE parameter can be different from this name and is then used in IMS as a transaction code.

5.1.4 Defining partner LUs and openUTM transactions

If IMS acts only as a job recipient, no further definitions are required in IMS. If IMS also acts as a job submitter, the address information must be generated for the partner program.

IBM recommends that partner transactions should be defined as side information in APPC/MVS. The term side information is used in CPI-C. The side information contains a symbolic destination name, a partner LU name, a partner TP name and a MODE name. The symbolic name is then used in the application programs to address the partner transaction. Here is an example of side information in APPC/MVS:

```
SIADD
    DESTNAME(DESTUTM1)
    TPNAME(UTMTAC01)
    MODENAME(APPCMODE)
    PARTNER_LU(APPCLUX)
```

The name of the side information file must be made known to IMS. This is done by means of a SIDEINFO statement in SYS1.PARMLIB(APPCPMxx):

```
SIDEINFO DATASET(SYS1.APPCSI)
```

If the standard DL/I program interface is used without the LU6.2 extensions of the CHNG call, the openUTM transaction cannot be addressed by means of side information. Instead, an LTERM name must be used. As in openUTM, an LTERM in IMS is a symbolic name for a communication partner that is used at the DL/I program interface. LTERM names must be assigned to a real communication partner by means of IMS generation parameters.

In this kind of addressing you begin by defining an alternate PCB. Generation statements are required at three different points to define an alternate PCB:

1. The PCB statement in IMS PSBGEN

Example:

```
PCB TYPE=TP,MODIFY=YES
PSBGEN PSBNAME=IMSTOJMS,CMPAT=YES,LANG=COBOL
```

2. The definition of ACBGEN

Example:

```
//ACBGEN   EXEC ACBGEN,COMP='POSTCOMP'
//G.SYSIN  DD *
  BUILD PSB=IMSTOJMS
```

3. The APPLCTN macro with the parameter PSB and the TRANSACT macro

Example:

```
APPLCTN  PSB=IMSOJMS,PGMTYPE=TP
TRANSACT CODE=IMSD,MSGTYPE=(,RESPONSE),INQUIRY=NO,MODE=SNGL
```

In addition, an LTERM name must be defined with an LU6.2 descriptor in IMS.PROCLIB(DFS62DTx) and assigned to this LU name, TP name and MODE name. Here is an example of what this looks like:

```
A UTMIMS01 LUNAME=APPCLUX TPNAME=UTMTAC01 MODE=APPCMODE SYNCLEVEL=N
```

The LTERM name in this example is UTMIMS01.

You can also combine the two types of partner definition by using the parameter SIDE instead of LUNAME, TPNAME and MODE for the LU6.2 descriptor, thus referencing side information.

In contrast to CICS, the maximum number of sessions to a partner LU is not specified in the case of IMS. This maximum number is specified in the APPL definition of VTAM.

5.1.5 VTAM generation

An initial part of the required VTAM generation is described above in the description of the APPL macro.

The MODE name used for the LU6.2 communication must be defined in VTAM. Here is an example of the definition of a MODE name:

```
APPCMODE MODEENT LOGMODE=APPCMODE,                                      *
               RUSIZES=X'8989',                                         *
               PSNDPAC=X'00',                                           *
               SRCVPAC=X'00',                                           *
               SSNDPAC=X'01'
```

RUSIZES specifies the maximum RU size in the sending and receiving direction. It is encoded in 2 hexadecimal characters: X'abab'. The actual RUSIZE is then calculated using the formula n = a* 2b. The values for PSNDPAC, SRCVPAC and SSNDPAC influence the pacing count.

Example of the VTAM definition of the partner PU and partner LU

```
P02CG3   PU    ADDR=C1,                                                 *
               DISCNT=NO,                                               *
               DLOGMOD=SNX32702,   WITH QUERY                           *
               IDBLK=017,          IDBLK FOR IBM-3174                   *
               IDNUM=20003,                                             *
               ISTATUS=ACTIVE,                                          *
               MAXDATA=1033,                                            *
               MAXOUT=7,                                                *
               MAXPATH=2,                                               *
               MODETAB=MOD3270,    LOGMODE TABLE FOR REMOTE 3270        *
               PACING=0,                                                *
               PUTYPE=2,                                                *
               SSCPFM=USSSCS,                                           *
               USSTAB=USSSCS,                                           *
               VPACING=0
*
APPCLUX LU     LOCADDR=0,           INDEPENDENT LU                      *
               USSTAB=USSSCS,       ACF/VTAM - USS-TABLE                *
               DLOGMOD=APPCMODE,    ACF/VTAM - DEFAULT LOGMODE          *
               MODETAB=LOGMODES,    LOGMODE TABLE                       *
               RESSCB=4,            RESERVE 4 SCB'S FOR THIS LU         *
               PACING=3,                                                *
               VPACING=2,                                               *
               SSCPFM=FSS
```

Example of the definition of a partner system linked by means of Enterprise Extender without the definition of the LUs

```
SMNEEA2  VBUILD TYPE=SWNET
*
PUEEA1   PU    IDBLK=003,                                              *
               IDNUM=00002,                                            *
               MAXPATH=5,                                              *
               MAXDATA=256,                                            *
               ADDR=01,                                                *
               CPNAME=MCH00XYC,                                        *
               CPCP=YES,                                               *
               HPR=YES,                                                *
               PUTYPE=2
PATH2A   PATH  SAPADDR=8,                                              *
               IPADDR=111.22.333.144,
               GRPNM=GPP390,                                         *
```

5.1.6 LU6.2 security

By LU6.2 security we understand the protection of transactions against unauthorized access. There are 3 security variants when setting up a conversation:

- NONE: No user ID is passed.
- ALREADY_VERIFIED: A user ID is passed, together with an indication to the job recipient that the user has already been verified by the job submitter.
- PROGRAM: A user ID and a password are passed.

IMS uses the product RACF for the purpose of access control. User IDs sent by partner applications therefore always have to be entered in RACF. Which user can call which transaction code is entered in RACF. The passwords for the user IDs are also entered in RACF.

The SECACPT parameter of the VTAM-APPL macro specifies whether a user ID and possibly a password have to be passed when an LU6.2 conversation is set up to IMS. The details of the RACF checking of incoming conversations are specified in the RACF parameter of the TP_Profile definition.

5.1.7 Full generation example

The following example contains the generation statements of an openUTM-IMS interconnection in the following circumstances:

- openUTM-LU62 is running on a Windows system.
- The interconnection between the IBM host and the Windows system is implemented by means of Enterprise Extender.

The file names of the IMS and VTAM definitions should be regarded as examples. Different file names can be chosen.

IMS definitions

IMS system definitions in IMS.CNTL(STAGE1):

```
//STAGE1   EXEC PGM=ASMA90,PARM='NOOBJ,DECK',REGION=0M
//SYSPRINT DD SYSOUT=*
//SYSLIB   DD DISP=SHR,DSN=IMS810.ADFSMAC
//SYSPUNCH DD DISP=SHR,DSN=USER.IMS.CNTL(STAGE2)
//SYSUT1   DD UNIT=3390,SPACE=(CYL,(05,05)),DCB=OPTCD=C
//SYSUT2   DD UNIT=3390,SPACE=(CYL,(05,05)),DCB=OPTCD=C
//SYSUT3   DD UNIT=3390,SPACE=(CYL,(05,05)),DCB=OPTCD=C
//SYSIN    DD *
         APPLCTN  GPSB=TESTIMS1,PGMTYPE=TP
         TRANSACT CODE=IMS1,MSGTYPE=(,NONRESPONSE),INQUIRY=NO,MODE=SNGL
         APPLCTN  GPSB=IMSTAC0,PGMTYPE=TP
         TRANSACT CODE=IMS0,MSGTYPE=(,RESPONSE),INQUIRY=NO,MODE=SNGL
        END ,
```

TP_Profile definition in IMS.CNTL(APPCTP):

```
//         EXEC PGM=ATBSDFMU
//SYSPRINT DD   SYSOUT=*
//SYSSDLIB DD   DSN=SYS1.APPCTP,DISP=SHR
//SYSSDOUT DD   SYSOUT=*
//SYSIN    DD   DATA,DLM=XX
     TPADD TPSCHED_EXIT(DFSTPPE0)
          TPNAME(IMSX)
          SYSTEM
          ACTIVE(YES)
          TPSCHED_DELIMITER(##)
             TRANCODE=TESTIMSX
             CLASS=1
             MAXRGN=1
             RACF=NONE
             CPUTIME=0
XX
```

Side information in IMS.CNTL(APPCSI):

```
//         EXEC PGM=ATBSDFMU
//SYSPRINT DD   SYSOUT=*
//SYSSDLIB DD   DSN=SYS1.APPCSI,DISP=SHR
//SYSSDOUT DD   SYSOUT=*
//SYSIN    DD   DATA,DLM=XX
     SIDELETE
          DESTNAME(DESTJW1)
     SIADD
          DESTNAME(DESTJW1)
          TPNAME(DATAECHO)
          MODENAME(APPCHOST)
          PARTNER_LU(P390.IMSJ)
XX
```

LTERM definition in PROCLIB(DFS62DTI):

```
A LTERMJW1 SIDE=DESTJW1  SYNCLEVEL=N CONVTYPE=M
```

IMS LU definition in PARMLIB(APPCPM1A):

```
LUADD ACBNAME(MVSLU01) BASE TPDATA(SYS1.APPCTP)
LUADD ACBNAME(IMSA)    BASE SCHED(IVP1) TPDATA(SYS1.APPCTP)
SIDEINFO DATASET(SYS1.APPCSI)
```

VTAM generation

Starting options:

```
SSCPNAME=P390SSCP,
NETID=P390
```

Switched major node definition in VTAMLST(SWXCAEE):

```
SWXCA1A VBUILD TYPE=SWNET,MAXNO=256,MAXGRP=256
SW1A13A     PU IDBLK=05D,IDNUM=25129,MAXPATH=5,MAXDATA=256,ADDR=01,    -
               CPNAME=MHPB02GC,                                         -
               CPCP=YES,HPR=YES,                                        -
               PUTYPE=2
PATH13A   PATH SAPADDR=4,                                               -
               IPADDR=111.22.33.233,                                    -
               GRPNM=GPP390
```

Major node definition for Enterprise Extender in VTAMLST(XCAEE1):

```
XCAEEE1 VBUILD TYPE=XCA
PORT1A    PORT MEDIUM=HPRIP,IPPORT=12000,                                  -
               IPTOS=(20,40,80,C0,C0),LIVTIME=10,                          -
               SRQTIME=15,SRQRETRY=3,SAPADDR=4
GPP390   GROUP DIAL=YES,ANSWER=ON,ISTATUS=ACTIVE,CALL=INOUT,               -
               DYNPU=YES,IPADDR=111.22.33.244
LNEE01    LINE
PUEE01      PU
LNEE02    LINE
PUEE02      PU
```

APPL definition in VTAMLST(A0IMS):

```
A0IMS    VBUILD TYPE=APPL              APPLICATION MAJOR NODE
IMSA     APPL  ACBNAME=IMSA,           ACBNAME MUST BE EQ LABEL        C
               APPC=YES,                                               C
               ATNLOSS=ALL,                                            C
               AUTOSES=0,                                              C
               DDRAINL=NALLOW,                                         C
               DLOGMOD=APPCHOST,                                       C
               DMINWNL=5,                                              C
               DMINWNR=5,                                              C
               DRESPL=NALLOW,                                          C
               DSESLIM=10,                                             C
               LMDENT=19,                                              C
               PARSESS=YES,                                            C
               SECACPT=NONE,                                           C
               SRBEXIT=YES,                                            C
               SYNCLVL=SYNCPT,                                         C
               VPACING=1
```

MODE definition in VTAM.SOURCE(ISTINCLM):

```
ISTINCLM MODETAB
APPCHOST MODEENT LOGMODE=APPCHOST,                                     *
               RUSIZES=X'8989',   4096                                 *
               SRCVPAC=X'00',                                          *
               SSNDPAC=X'01'
```

IBM Communications Server

```
NODE=(
          ANYNET_SUPPORT=NONE
          CP_ALIAS=MHPB02GC
          DEFAULT_PREFERENCE=NATIVE
          DISCOVERY_SUPPORT=DISCOVERY_SERVER
          DLUR_SUPPORT=NORMAL
          FQ_CP_NAME=P390.MHPB02GC
          MAX_LOCATES=150
          MAX_LS_EXCEPTION_EVENTS=200
          NODE_ID=05D25129
          NODE_TYPE=NETWORK_NODE
          REGISTER_WITH_CDS=1
          REGISTER_WITH_NN=NONE
          SEND_TERM_SELF=0
          TP_SECURITY_BEHAVIOR=VERIFY_EVEN_IF_NOT_DEFINED
)
PORT=(
          ACTIVATION_DELAY_TIMER=30
          ALLOW_ABM_XID_MISMATCH=0
          DELAY_APPLICATION_RETRIES=1
          DLC_NAME=IBMEEDLC
          IMPLICIT_BRANCH_EXTENDER_LINK=0
          IMPLICIT_CP_CP_SESS_SUPPORT=1
          IMPLICIT_DEACT_TIMER=600
          IMPLICIT_DSPU_SERVICES=NONE
          IMPLICIT_HPR_SUPPORT=1
          IMPLICIT_LIMITED_RESOURCE=NO
          IMPLICIT_LINK_LVL_ERROR=0
          LINK_STATION_ROLE=NEGOTIABLE
          MAX_ACTIVATION_ATTEMPTS=0
          MAX_IFRM_RCVD=8
          MAX_RCV_BTU_SIZE=1500
          PORT_NAME=IBMEEDLC
          PORT_TYPE=SATF
          RETRY_LINK_ON_DISCONNECT=1
          RETRY_LINK_ON_FAILED_START=1
          RETRY_LINK_ON_FAILURE=1
```

```
DEFAULT_TG_CHARS=(
          COST_PER_BYTE=0
          COST_PER_CONNECT_TIME=0
          EFFECTIVE_CAPACITY=133
          PROPAGATION_DELAY=LAN
          SECURITY=NONSECURE
          USER_DEFINED_1=0
          USER_DEFINED_2=0
          USER_DEFINED_3=0
)
PORT_OEM_SPECIFIC_DATA=(
OEM_LINK_DATA=(
          OEM_DATA=010000000400000004000000030000000F00000000000000
          OEM_DATA=0A00000000000000000
)
OEM_PORT_DEFAULTS=(
          COST_PER_CONNECT_TIME=0
          EFFECTIVE_CAPACITY=133
          INB_LINK_ACT_LIM=128
          OUT_LINK_ACT_LIM=127
          PROPAGATION_DELAY=LAN
          SECURITY=NONSECURE
          TOT_LINK_ACT_LIM=255
)
)
)
LINK_STATION=(
          ACTIVATE_AT_STARTUP=1
          ACTIVATION_DELAY_TIMER=-1
          ADJACENT_BRANCH_EXTENDER_NODE=PROHIBITED
          ADJACENT_NODE_TYPE=LEARN
          AUTO_ACTIVATE_SUPPORT=0
          BRANCH_EXTENDER_LINK=1
          CP_CP_SESS_SUPPORT=1
          DEFAULT_NN_SERVER=0
          DELAY_APPLICATION_RETRIES=0
          DEPENDENT_LU_COMPRESSION=0
          DEPENDENT_LU_ENCRYPTION=OPTIONAL
          DEST_ADDRESS=F45519AC
          DISABLE_REMOTE_ACT=0
          DSPU_SERVICES=NONE
          FQ_ADJACENT_CP_NAME=P390.P390SSCP
          HPR_LINK_LVL_ERROR=0
          HPR_SUPPORT=1
          INHERIT_PORT_RETRY_PARMS=1
          LIMITED_RESOURCE=NO
          LINK_DEACT_TIMER=600
          LINK_STATION_ROLE=NEGOTIABLE
```

```
                LS_NAME=P390SSCP
                MAX_ACTIVATION_ATTEMPTS=-1
                MAX_IFRM_RCVD=7
                MAX_SEND_BTU_SIZE=1500
                NODE_ID=05D25129
                NULL_ADDRESS_MEANING=USE_WILDCARD
                PORT_NAME=IBMEEDLC
                PU_NAME=P390SSCP
                RETRY_LINK_ON_DISCONNECT=0
                RETRY_LINK_ON_FAILED_START=0
                RETRY_LINK_ON_FAILURE=0
                REVERSE_ADDRESS_BYTES=0
                SOLICIT_SSCP_SESSION=0
                TG_NUMBER=0
                USE_DEFAULT_TG_CHARS=1
                USE_PU_NAME_IN_XID=0
      TG_CHARS=(
                COST_PER_BYTE=0
                COST_PER_CONNECT_TIME=0
                EFFECTIVE_CAPACITY=0
                PROPAGATION_DELAY=MINIMUM
                USER_DEFINED_1=0
                USER_DEFINED_2=0
                USER_DEFINED_3=0
      )
      LINK_STATION_OEM_SPECIFIC_DATA=(
      OEM_LINK_DATA=(
                OEM_DATA=0100000004000000040000000300000000F00000000000000
                OEM_DATA=0A000000F45519AC
      )
      )
      )
      DLUR_DEFAULTS=(
                DEFAULT_PU_NAME=MHPB02GC
                DLUS_RETRY_LIMIT=3
                DLUS_RETRY_TIMEOUT=5
      )
      LOCAL_LU=(
                LU_NAME=IMSJ
                DEFAULT_POOL=0
                LU_ALIAS=IJWIMSJ
                LU_SESSION_LIMIT=0
                NAU_ADDRESS=0
                ROUTE_TO_CLIENT=0
                SYNCPT_SUPPORT=1
      )
```

```
MODE=(
          MODE_NAME=APPCHOST
          AUTO_ACT=6
          COMPRESSION=PROHIBITED
          COS_NAME=#CONNECT
          DEFAULT_RU_SIZE=0
          ENCRYPTION_SUPPORT=NONE
          MAX_INCOMING_COMPRESSION_LEVEL=NONE
          MAX_NEGOTIABLE_SESSION_LIMIT=128
          MAX_OUTGOING_COMPRESSION_LEVEL=NONE
          MAX_RU_SIZE_UPPER_BOUND=4096
          MIN_CONWINNERS_SOURCE=6
          PLU_MODE_SESSION_LIMIT=12
          RECEIVE_PACING_WINDOW=1
)
PARTNER_LU=(
          FQ_PLU_NAME=P390.IMSA
          ADJACENT_CP_NAME=
          CONV_SECURITY_VERIFICATION=1
          MAX_MC_LL_SEND_SIZE=32767
          PARALLEL_SESSION_SUPPORT=1
          PARTNER_LU_ALIAS=PA713236
          PREFERENCE=USE_DEFAULT_PREFERENCE
)
SPLIT_STACK=(
          STARTUP=1
          POOL_NAME=
)
SHARED_FOLDERS=(
          CACHE_SIZE=256
EXTENSION_LIST=(
)
)
LOAD_BALANCING=(
          ADVERTISE_FREQUENCY=1
          APPC_LU_LOAD_FACTOR=0
          DEFAULT_MAX_LU62_SESSIONS=512
          ENABLE_LOAD_BALANCING=0
          HOST_LU_LOAD_FACTOR=0
          LOAD_VARIANCE=3
)

VERIFY=(
          CFG_MODIFICATION_LEVEL=12
          CFG_VERSION_LEVEL=1
          CFG_LAST_SCENARIO=6
)
```

openUTM-LU62 generation without using TNSX

```
INSTANCE
     LOC-LU-ALIAS=IJWIMSJ,
     LOC-TSEL=T'IJWOSICL',
     LOC-LISTENER-PORT=29664,
     LOC-APT=(1,2,29664),
     LOC-AEQ=712614,
     REM-LU-ALIAS=PA713236,
     MODENAME=APPCHOST,
     ALLOC-TIME=0,
     LU62-CODE=*NO,
     REM-NSEL=utmhost,
     REM-LISTENER-PORT=23000,
     REM-TSEL=T'SMP23000',
     REM-APT=(1,2,29660),
     REM-AEQ=1,
     APPL-CONTEXT=UDTSEC,
     ASSOCIATIONS=12,
     CONNECT=12,
     CONTWIN=6,
     OSITP-CODE=*NO
```

openUTM-LU62 generation using TNSX

```
INSTANCE
     LOC-LU-ALIAS=IJWIMSJ,
     LOC-AE=IJWOSICL,
     LOC-APT=(1,2,29664),
     LOC-AEQ=712614,
     REM-LU-ALIAS=PA713236,
     MODENAME=APPCHOST,
     ALLOC-TIME=0,
     LU62-CODE=*NO,
     REM-AE=BCOSIO.SMP23000,
     REM-APT=(1,2,29660),
     REM-AEQ=1,
     APPL-CONTEXT=UDTSEC,
     ASSOCIATIONS=12,
     CONNECT=12,
     CONTWIN=6,
     OSITP-CODE=*NO
```

TNSX generation

```
BCOSIO.SMP23000\
          TA        RFC1006 111.33.55.111 PORT 23000 T'SMP23000'
          PSEL      V''
          SSEL      V''
IJWOSICL\
          TSEL      RFC1006 T'IJWOSICL'
          TSEL      LANINET A'29664'
          PSEL      V''
          SSEL      V''
```

UTM generation

```
UTMD    APT=(1,2,29660),                            -
        MAXJR=200,                                  -
        CONCTIME=(180,60),                          -
        PTCTIME=0
ACCESS-POINT BCOSIO,                                -
        P-SEL=*NONE,                                -
        S-SEL=*NONE,                                -
        T-SEL=C'SMP23000',                           -
        AEQ=1,                                      -
        T-PROT=RFC1006,                             -
        LISTENER-PORT=23000,                        -
        TSEL-FORMAT=T
OSI-CON IJWOSICL,                                   -
        ACTIVE=YES,                                 -
        LISTENER-PORT=29664,                        -
        LOCAL-ACCESS-POINT=BCOSIO,                  -
        MAP=USER,                                   -
        N-SEL=C'MHPBO2GC',                          -
        OSI-LPAP=IJWOSICL,                          -
        P-SEL=*NONE,                                -
        S-SEL=*NONE,                                -
        T-PROT=RFC1006,                             -
        T-SEL=C'IJWOSICL',                          -
        TSEL-FORMAT=T
OSI-LPAP IJWOSICL,                                  -
        APPLICATION-CONTEXT=UDTSEC,                 -
        APPLICATION-ENTITY-QUALIFIER=712614,        -
        APPLICATION-PROCESS-TITLE=(1,2,29664),      -
        ASSOCIATION-NAMES=IJW,                      -
        ASSOCIATIONS=12,                            -
        CONNECT=12,                                 -
        CONTWIN=6,                            -
        IDLETIME=0,                           -
        PERMIT=ADMIN,                         -
        STATUS=ON
TAC DATAECHO,                                 -
        PROGRAM=comims,                      -
        TYPE=D
LTAC    LIMS0,                                -
        RTAC=IMS0,                            -
        LPAP=IJWOSICL,                        -
        TYPE=D
LTAC    LIMS1,                                -
        RTAC=IMS1,                            -
        LPAP=IJWOSICL,                        -
        TYPE=A
LTAC    ICPC,                                 -
        RTAC=CSVRCPIC,                        -
        LPAP=IJWOSICL,                        -
        TYPE=D
```

The essential parameters in this example are as follows:

Parameter	**IMS host**	**openUTM-LU62 Windows**	**openUTM end system**
IP address	111.22.33.244 (F45519AC)	111.22.33.233	111.33.55.111
LU name	P390.IMSA	P390.IMSJ	
LU alias	PA713236	IJWIMSJ	
CP name	P390.P390SSCP	P390.MHPB02GC	
IDBLK/IDNUM		05D 25129	
MODE name	APPCHOST		
T selector		IJWOSICL	SMP23000
Port number		29664	23000
AP title		(1,2,29664)	(1,2,29660)
AE qualifier		712614	1
APPL context		UDTSEC	

5.2 Programming an openUTM-IMS interconnection

IMS application programs that use LU6.2 can be implemented by means of the DL/I or CPI-C program interface.

In the IMS literature, distinctions are drawn between three IMS application program variants in LU6.2 communication:

- Standard DL/I: The application program uses DL/I calls exclusively.
- Modified DL/I: DL/I is used to read the message from the job submitter and to send the reply to the job submitter. In addition, CPI-C is used to establish communication with a job recipient.
- CPI-C: The application program uses CPI-C calls exclusively.

5.2.1 DL/I program interface

DL/I is the conventional IMS program interface. Existing DL/I programs that receive messages from communication partners and send a reply back to the same partner can often communicate with LU6.2 partners without the need for any changes. Two-phase commit transactions can also be dealt with in this way. In addition, it is also possible with DL/I to send an LU6.2 partner a queued message without transaction management, as in the case of FPUT with openUTM. However, DL/I does not allow a two-phase commit transaction to be requested from the partner. Nor is it possible with DL/I to set a user ID when sending a queued message.

The following calls are available for data communication in DL/I:

Name	Parameters	Meaning	Purpose
AUTH	I/O PCB, I/O Area	authorization	To check whether the user can call specific functions and use resources
CHNG	Alt PCB, dest name, opt list	change	To address a partner
CMD	I/O PCB, I/O Area	command	To issue an administration command
GCMD	I/O PCB, I/O Area	get command	To read the response to the administration command
GU	I/O PCB, I/O Area	get unique	To read the first message
GN	I/O PCB, I/O Area	get next	To read a subsequent message
ISRT	I/O PCB or Alt PCB, I/O Area [, mod name]	insert	To send a message

PURG	I/O PCB or Alt PCB	purge	To send a message immediately (see Flush in CPI-C)
SETO	I/O PCB or Alt PCB, I/O Area, opt list	set options	To send Send_Error or Deallocate_Abend

In addition, the following DL/I calls can be used for system services in the LU6.2 environment:

Name	**Parameters**	**Meaning**	**Purpose**
INQY	aib, I/O Area	inquiry	To query properties of the output PCB
ROLB	I/O PCB, I/O Area	rollback	To send Deallocate_Abend and roll back the transaction. The application program resumes normally.
ROLL	None	roll	To send Deallocate_Abend and roll back the transaction. The program is terminated abnormally, and the input message is deleted.
ROLS	I/O PCB, I/O Area, token	rollback to sets/setu	To send Deallocate_Abend and roll back the transaction. The program is terminated, the input message remains in the queue, and the reset point can be specified.
SETS	I/O PCB, I/O Area, token	set backout point	To set a reset point for a subsequent ROLS
SETU	I/O PCB, I/O Area, token	set unconditional	As for SETS, but with slightly different behavior in error situations
SYNC	I/O PCB	sycnpoint	To set a synchronization point

Meaning of the parameters

I/O PCB and Alt PCB:

When a message is received and sent, a PCB (program communication block) containing information such as the status, partner name (LTERM), etc. is transferred. If a message is sent back to the job submitter, the I/O PCB is used. To send a message to a job recipient, an alternate PCB must be used for which a new destination address is set beforehand by means of the CHNG call. One component of the PCB is the partner type. The following partner types are possible:

- APPC (LU6.2 partner)
- OTMA (TCP/IP partner)
- TERMINAL (terminal or LU6.1 partner)
- TRANSACTION (transaction code).

I/O Area
: The message area

dest name (destination name):
: LTERM or transaction code name of the partner

opt list (options list):
: In the case of CHNG, parameters of the LU6.2 partner can be specified here: LU name, MODE name, TP name and sync-level (none or confirm). Alternatively, a symbolic destination name can also be specified. In LU6.2 communication, addressing is carried out either by means of opt list or by means of an LTERM name in the dest name field.

  In the case of SETO, it is specified here whether Send_Error or Deallocate_Abend is to be sent.

mod name (message output descriptor name):
: Format name

aib (application interface control block):
: This contains the PCB address and some additional information.

token: Name of a reset point

It is also possible to program longer dialogs alternately with GU/GN and ISRT.

Example of an extract from a DL/I-COBOL program (GN call)

```
ID DIVISION.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
  77 GN-FUNC      PIC X(4) VALUE 'GN  '.
  01 IOAREA       PIC X(256).
LINKAGE SECTION.
  01 D1PC.
    02 D1PCDBN  PIC X(8).
    02 D1PCLEVL PIC 9(2).
    02 D1PCSTAT PIC X(2).
    02 D1PCPROC PIC X(4).
    02 D1PCRESV PIC S9(5) COMP.
    02 D1PCSEGN PIC X(8).
    02 D1PCKFBL PIC S9(5) COMP.
    02 D1PCNSSG PIC S9(5) COMP.
    02 D1PCKFBA PIC X(20).
PROCEDURE DIVISION.
  ENTRY 'DLITCBL' USING D1PC.
  ...
```

```
CALL 'CBLTDLI' USING GN-FUNC, D1PC, IOAREA.
IF D1PCSTAT NOT = '  '
  ...
GOBACK.
```

5.2.2 CPI-C program interface

CPI-C allows the entire scope of LU6.2 to be exploited. Only with this program interface is it possible from IMS programs to set up sync-level, sync-point conversations to partners. CPI-RR must be used in this case to control the two-phase commit. For more information, see also the section “Using the CPI-C program interface” on page 153.

5.2.3 LU6.2 Edit Exit routine

IMS always runs through the LU6.2 Edit Exit routine (DFSLUEE0) when a DL/I program sends or receives an LU6.2 message. IBM supplies a ready-made exit routine that can be modified by the customer. The exit routine can be used for the following purposes:

- To change the synchronization level of an asynchronous LU6.2 conversation
- To change the contents of messages
- To delete message segments
- To cancel (DEALLOCATE_ABEND) an LU6.2 conversation

5.2.4 Use of format names

MFS (Message Format Service) is the IMS component for handling formatted inputs and outputs on character-oriented screens (type 3270).

When reading in a 3270 message, DL/I application programs receive a format name known as the MOD name in the GU call. MOD stands for Message Output Descriptor. In the case of ISRT they can pass a MOD name. If these kinds of programs are to be used unchanged for LU6.2 communication, the LU6.2 Edit Exit routine (DFSLUEE0) must be used for these purposes. The LU6.2 partner programs then have to send the format name at the beginning of the message and receive a format name at the beginning of the return message.

5.2.5 Examples of execution sequences with DL/I programs

The following examples are of execution sequences with openUTM and IMS programs. On both sides, only the program calls for the LU6.2 communication are listed. Further program calls are required for communication with the openUTM or IMS client.

The return values in the UTM application program are shown in *italics* in the following examples.

1. Synchronous call of an IMS transaction by openUTM without transaction management

IMS application		**UTM application**
		APRO DM KCOF=B, KCRN=A
		MPUT NE, KCRN=A
		PEND KP
	<-- FMH5 + data	
GU IOPCB		
ISRT IOPCB		
Exit		
	Data + CEB -->	
		MGET NT *VGST=C,* KCRN=A

2. Asynchronous call of an IMS transaction by openUTM without transaction management

IMS application		**UTM application**
		APRO AM KCOF=B, KCRN=A
		FPUT NE, KCRN=A
		PEND FI
	<-- FMH5 + data + CEB	
	Acknowledgment -->	
GU IOPCB		
ISRT IOPCB		
Exit		
	FMH5 + data + CEB -->	
	<-- Acknowledgment	
		TAC DFSASYNC
		FGET
		PEND FI

An openUTM-LU62 instance generated with the parameter UTM-ASYNC=YES must be used for this kind of communication.

Note:
The TAC DFSASYNC must be generated in the UTM application as ASYNTAC. The name DFSASYNC cannot be freely chosen. It is defined by IMS.

3. Asynchronous call of an IMS transaction by openUTM without transaction management, IMS program does not send a message

IMS application		**UTM application**
		APRO AM KCOF=B, KCRN=A
		FPUT NE, KCRN=A
		PEND FI
	<-- FMH5 + data + CEB	
	Acknowledgment -->	
GU IOPCB		
Exit		

An openUTM-LU62 instance generated with the parameter UTM-ASYNC=YES must be used for this kind of communication.

4. Synchronous call of an IMS transaction by openUTM without transaction management, multi-step transaction

IMS application		**UTM application**
		APRO DM KCOF=B, KCRN=A
		MPUT NE, KCRN=A
		PEND KP
	<-- FMH5 + data	
GU IOPCB		
ISRT IOPCB		
	Data -->	
		MGET NT *VGST=O,* KCRN=A
		MPUT NE, KCRN=A
		PEND KP
	<-- Data	
GU IOPCB		
ISRT IOPCB		
Exit		
	Data + CEB -->	
		MGET NT *VGST=C,* KCRN=A

In this example, only the IMS DL/I calls for sending and receiving messages are listed. To keep data available over several dialog steps, IMS DL/I programs typically work with a scratch pad area. Further GU, GN or ISRT calls are required to read and write the scratch pad area.

5. Synchronous call of an IMS transaction by openUTM with transaction management, first variant

IMS application		**UTM application**
		APRO DM KCOF=C, KCPI=>A
		MPUT NE, KCRN=A
		PEND KP
	<--FMH5 + data	
GU IOPCB		
ISRT IOPCB		
	Data + RQD2-->	
		(System code)
	<- +DR2	
		MGET NT *VGST=O,* KCRN=A
		PEND FI
	<-- Prepare	
	Req Commit-->	
	<-- Committed	
	Forget + CEB-->	

6. Synchronous call of an IMS transaction by openUTM with transaction management, second variant

IMS application		**UTM application**
		APRO DM KCOF=C, KCRN=A
		MPUT NE, KCRN=A
		PEND KP
	<--FMH5 + data	
GU IOPCB		
ISRT IOPCB		
	Data + RQD2-->	
		(System code)
	<- +DR2	
		MGET NT *VGST=O,* KCRN=A
		MPUT NE KCLM=0, KCRN=A
		CTRL PE
		PEND KP
	<-- Prepare	
	Req Commit-->	
		MGET NT *VGST=C,* KCRN=A
	<-- Committed	
	Forget + CEB-->	
		PEND FI

In example 6, the 3rd program unit sees whether IMS issues a commit or a rollback.

In example 5, the program unit learns no more about commit or rollback in IMS. This is controlled only by the UTM system code.

7. Synchronous call of an IMS transaction by openUTM with transaction management, rollback by openUTM

IMS application		**UTM application**
		APRO DM KCOF=C, KCRN=A
		MPUT NE, KCRN=A
		PEND KP
	<--FMH5 + data	
GU IOPCB		
ISRT IOPCB		
	Data -->	
		MGET NT *VGST=O,* KCRN=A
		PEND FR
	<-- Backout	
	Acknowledgment -->	

8. Synchronous call of an IMS transaction by openUTM with transaction management, rollback by the IMS program

IMS application		**UTM application**
		APRO DM KCOF=C, KCRN=A
		MPUT NE, KCRN=A
		PEND KP
	<--FMH5 + data	
GU IOPCB		
ISRT IOPCB		
ROLB		
	FMH7 -->	
	Error data -->	
	FMH7 + CEB -->	
		Rollback of the UTM transaction

5.2.6 Examples of execution sequences with CPI-C programs

1. Synchronous call of an IMS CPI-C program by openUTM without transaction management

IMS application		**UTM application**
		APRO DM KCOF=B, KCRN=A
		MPUT NE, KCRN=A
		PEND KP
	<-- FMH5 + data	
Accept		
Receive_Data		
Receive_SEND		
Send_Data		
SRRCMIT (local sync point)		
Deallocate		
	Data + CEB -->	
		MGET NT *VGST=C,* KCRN=A

   This example corresponds to example 20 in the section "Examples of openUTM-CPIC communication" on page 156.

2. Synchronous call of an IMS CPI-C program by openUTM with transaction management

IMS application		**UTM application**
		APRO DM KCOF=C, KCPI=A
		MPUT NE KCRN=A
		PEND KP
	<--FMH5 + data	
Accept		
Receive DATA		
Receive SEND		
Send_Data		
Prep_To_Rcv		
	Data -->	
		MGET NT KCRN=A *VGST=O,*
		PEND FI

IMS application		**UTM application**
	<-- Prepare	
Receive SYNCP-DEAL		
SRRCMIT		
	Req Commit-->	
		(UTM system code)
	<-- Committed	
	Forget ------->	
		PEND FI

This example corresponds to example 12 in the section “Examples of openUTM-CPIC communication” on page 156, where you will find further examples.

5.2.7 Examples of execution sequences with standard IMS transactions

1. Synchronous call of an IMS command by openUTM (e.g. /DISPLAY APPC)

IMS command		**UTM application**
		APRO DM KCOF=B "/DISPLAY", KCRN=A
		MPUT NE "APPC", KCRN=A
		PEND KP
	<-- FMH5 + data	
/DISPLAY APPC		
Output		
	Data + CEB -->	
		MGET NT *VGST=C,* KCRN=A

Note:
You can use the "/DISPLAY APPC" command to output the status of all APPC resources.

2. Asynchronous call of an IMS command of openUTM (e.g. /DISPLAY APPC)

IMS command		**UTM application**
		APRO AM KCOF=B "/DISPLAY", KCRN=A
		FPUT NE "APPC", KCRN=A
		PEND FI
	<-- FMH5 + data + CEB	
	Acknowledgment -->	
/DISPLAY APPC		
Output		
	FMH5 + data + CEB -->	
	<-- Acknowledgment	
		TAC DFSCMD
		FGET
		PEND FI

An openUTM-LU62 instance generated with the parameter UTM-ASYNC=YES must be used for this kind of communication.

Note: The TAC DFSCMD must be generated as ASYNTAC in the UTM application. The name DFSCMD cannot be freely chosen. It is defined by IMS.

3. Synchronous call of the IMS Message Switch by openUTM

DFSAPPC		**UTM application**
		APRO DM KCOF=B "DFSAPPC", KCRN=A
		MPUT NE, KCRN=A
		PEND KP
	<-- FMH5 + data	
Message Switch call		
	CEB -->	
		MGET NT *VGST=C,* KCRN=A

Note: The IMS Message Switch allows queued messages to be sent to any LU6.2 partners of the IMS application. For example, the message "DFSAPPC (TPN=REPORT LU=FRED) REP1" causes IMS to send the message "REP1" to the TP name "REPORT" of the partner LU "FRED".

5.2.8 Examples of execution sequences with IMS as the job submitter

1. Call of a UTM asynchronous TAC by IMS without transaction management

IMS application		**UTM application**
GU IOPCB		
CHNG ALTPCB		
ISRT ALTPCB		
PURG ALTPCB		
	FMH5 + data + CEB -->	
	<-- Acknowledgment	
		INIT
		FGET
		PEND FI

2. Call of a UTM dialog TAC by IMS with transaction management

IMS application		**UTM application**
Allocate synclev 2		
Send_Data		
Prep_To_Rcv		
	FMH5 + data -->	
		INIT
		MGET *VGST=O*
		MPUT NE
		PEND KP
	<-- Data	
Receive DATA		
Deallocate		
SRRCMIT		
	Prepare -->	
		INIT
		MGET *TAST=P*
		PEND FI

IMS application		UTM application
	<-- Req Commit	
	Committed -->	
	<-- Forget + CEB	

Transaction-oriented communication with IMS as the job submitter is only possible when the IMS application programs use the CPI-C program interface.

5.3 IMS administration

Administration commands are essentially required for the following tasks:

- To display the connection statuses
- To set up LU6.2 sessions to the partner
- To resolve exchange log name problems

The following IMS commands are available for displaying the connection statuses

/DISPLAY APPC,LU,ALL	Displays all LUs.
/DISPLAY APPC,TP,ALL	Displays the LU6.2 TP names used.
/DISPLAY APPC,CONV	Displays current conversations.
/DISPLAY APPC,UOR,ALL	Displays the current two-phase commit transactions.

You can use the IMS command /ALLOCATE to set up sessions to LU6.2 partners.

The LU6.2 protocol contains an exchange log name when a session is set up. The two partners exchange their log names. It is thus possible to ascertain whether one of the two partners has deleted its log records with a cold start since the last connection cleardown. If one partner has carried out a cold start, but the other partner has not, there is a warm-cold mismatch. This can only be dealt with by manual intervention. In the case of openUTM-LU62, a warm-cold mismatch is displayed as the XLN status "err". If IMS has carried out a cold start, but openUTM-LU62 has not, you can deal with this by restarting the openUTM-LU62 instance using the cold-start option. Conversely, if openUTM-LU62 has been cold-started and IMS warm-started, intervention is required on the IMS side.

6 openUTM-CICS interconnection via LU6.1

This chapter covers the generation and programming of openUTM-CICS interconnections via LU6.1.

For CICS users, the generation and programming of openUTM-CICS interconnection are largely identical to the procedure for IMS-CICS interconnection via LU6.1 (see the “CICS for MVS/ESA Intercommunication Guide“ and the “CICS for MVS/ESA Distributed Transaction Programming Guide“).

6.1 CICS definitions for openUTM-CICS sessions

This section describes the generation of connections between CICS and openUTM.

Generations for CICS are possible via macros or online (resource definition online, RDO). Increasingly, only the RDO definition is described with newer CICS versions; for this reason, the RDO definition is described in this section.

For a connection to openUTM, RDO must be used to define the following:

```
CONNECTION
SESSIONS
```

In addition, the name of the CICS application is specified in the system initialization table (SIT).

DFHSIT APPLID=cics_netname
: Name (1-8 characters) which is used to define the SNA network name for the CICS application. This name must match the RNETNAME operand in the corresponding LPAP statement used for openUTM generation.

DEFINE CONNECTION

This definition is used to describe the remote system, in this case the openUTM application. It thus corresponds to the LPAP statement in the openUTM generation.

```
DEFINE CONNECTION   connection_name
       GROUP        group_name
       NETNAME      netname
       ACCESSMETHOD VTAM
       PROTOCOL     LU61
       DATASTREAM   USER
       RECORDFORMAT U | VB
              ...
```

The operands have the following meaning:

CONNECTION connection_name
: Name (1-4 characters) used to identify the openUTM application. This name only exists within CICS. It must be specified in CICS programs in the SYSID parameter in ALLOCATE when the CICS program wants to call a UTM transaction code. It must also appear in a session definition.

GROUP group_name
: Name (1-8 characters). The CICS system administrator must assign this name. It only appears in the CICS definitions.

NETNAME netname
: Name (1-8 characters) under which the openUTM application is known in the SNA network. This name must match the LNETNAME operand in the corresponding LPAP statement used for openUTM generation.

ACCESSMETHOD VTAM
: VTAM access method.

PROTOCOL LU61
: An LU6.1 connection is defined.

DATASTREAM USER
: The user data is completely defined by the application program.

RECORDFORMAT U | VB
: The format must correspond to the record format as it is sent from CICS. The default value U (=unformatted) is valid for records without a length field. If VB (=variable blocked) is specified, the records provided by CICS have to correspond to the VB format provided by LU6.1, i.e. they must also contain the length field.

DEFINE SESSIONS

This is used to define **one** LU6.1 session. This corresponds to the LSES statement used for openUTM generation.

```
DEFINE SESSIONS     session_name
       GROUP        group_name
       CONNECTION   connection_name
       SESSNAME     loc_ses_name
       NETNAMEQ     rem_ses_name
       PROTOCOL     LU61
       RECEIVECOUNT 01 | space
       SENDCOUNT    01 | space
       SENDSIZE     ru_size
       RECEIVESIZE  ru_size
       AUTOCONNECT  NO | YES
       BUILDCHAIN   YES
       IOAREALEN    value1, value2
              ...
```

The operands have the following meaning:

SESSIONS session_name
: Name of the session definition. This name only appears in the CICS definitions.

GROUP group_name
: Name (1-8 characters). The CICS system administrator must assign this name. It only appears in the CICS definitions. It should have the same name as the name selected for the associated connection definition.

CONNECTION connection_name
: Name of the associated CONNECTION definition.

SESSNAME loc_ses_name
: Name (1-4 characters) used to define the local half-session qualifier. This name must match the RSES operand in the corresponding LSES statement used for openUTM generation.

NETNAMEQ rem_ses_name
: Name (1-8 characters) used to define the remote half-session qualifier. This name must match the LSES name in the corresponding LSES statement used for openUTM generation.

PROTOCOL LU61
: An LU6.1 session is defined.

RECEIVECOUNT/SENDCOUNT

This is used to determine which partner is the PLU and which is the SLU. In the openUTM application, the corresponding specifications are made in the SESCHA statement.

RECEIVECOUNT 01
SENDCOUNT spaces

For this session, the CICS application is the PLU (primary logical unit) and contention loser. In the corresponding SESCHA statement used for openUTM generation, the following must be specified: PLU=Y,CONTWIN=N.

RECEIVECOUNT spaces
SENDCOUNT 01
For this session, the CICS application is the SLU (secondary logical unit) and contention winner. In the corresponding SESCHA statement used for openUTM generation, the following must be specified: PLU=N,CONTWIN=Y. This is not allowed for interconnection via TRANSIT-CLIENT.

SENDSIZE / RECEIVESIZE ru_size

This defines the maximum size of the request units (RU) for this session. The value for RU-size is determined by the maximum data length including the function management header (FMH). The function management headers have a size between 7 and 50 bytes. A value of 1024 is to be selected here for interconnections via TRANSIT-CLIENT.

AUTOCONNECT NO | YES

The connections are automatically activated (YES) or not activated (NO) when the system is started.

BUILDCHAIN YES

This operand must be specified for all openUTM-CICS sessions.

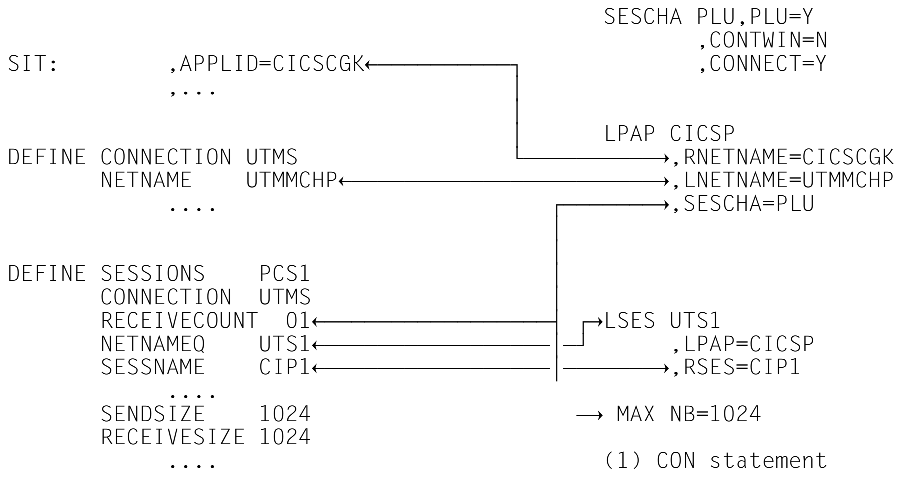
IOAREALEN value1, value2

The minimum size of the terminal input/output area is defined using *value1*. This value should be chosen so that it is at least as large as the maximum message length for the application program.

⚠ With interconnection via TRANSIT-CLIENT, CICS must always be the PLU, i.e. RECEIVECOUNT = 1. The maximum RU size must not exceed 1024 bytes. In addition, only one SESSION is allowed for each CONNECTION (no parallel sessions). If the data transfer is to be distributed over more than one session, several CONNECTIONs must be defined and the selection of the various sessions must be controlled via the openUTM or CICS application program.

Example of dependencies in CICS and openUTM generation

CICS definitions **openUTM generation**



```
                                                    SESCHA PLU,PLU=Y
                                                           ,CONTWIN=N
SIT:         ,APPLID=CICSCGK                               ,CONNECT=Y
             ,...

                                                    LPAP CICSP
DEFINE CONNECTION UTMS                                   ,RNETNAME=CICSCGK
       NETNAME    UTMMCHP                                ,LNETNAME=UTMMCHP
             ....                                        ,SESCHA=PLU


DEFINE SESSIONS    PCS1
       CONNECTION  UTMS
       RECEIVECOUNT  01                             LSES UTS1
       NETNAMEQ    UTS1                                  ,LPAP=CICSP
       SESSNAME    CIP1                                  ,RSES=CIP1
             ....
       SENDSIZE    1024                              MAX NB=1024
       RECEIVESIZE 1024
             ....                                   (1) CON statement
```

6.2 TRANSIT generation for openUTM-CICS sessions

The openUTM gateway 1 (UTMGW1) uses an internal interface from TRANSIT-SERVER. The LU6.1 protocols are transparently passed through by openUTM. In addition, session setup and cleardown and the conversion of the transmission header (FID1 ← → FID2) are handled. Communication with the openUTM system occurs via CMX.

Communication with the IBM application occurs via TRANSIT-SERVER and - depending on the interconnection mode - via CMX or directly via the SINIX kernel. For a more detailed description of the openUTM gateway, please refer to the "TRANSIT-CLIENT" manual.

For a more detailed description of the possible interconnections between TRANSIT-SERVER and the IBM system, please refer to the "TRANSIT-SERVER" manual.

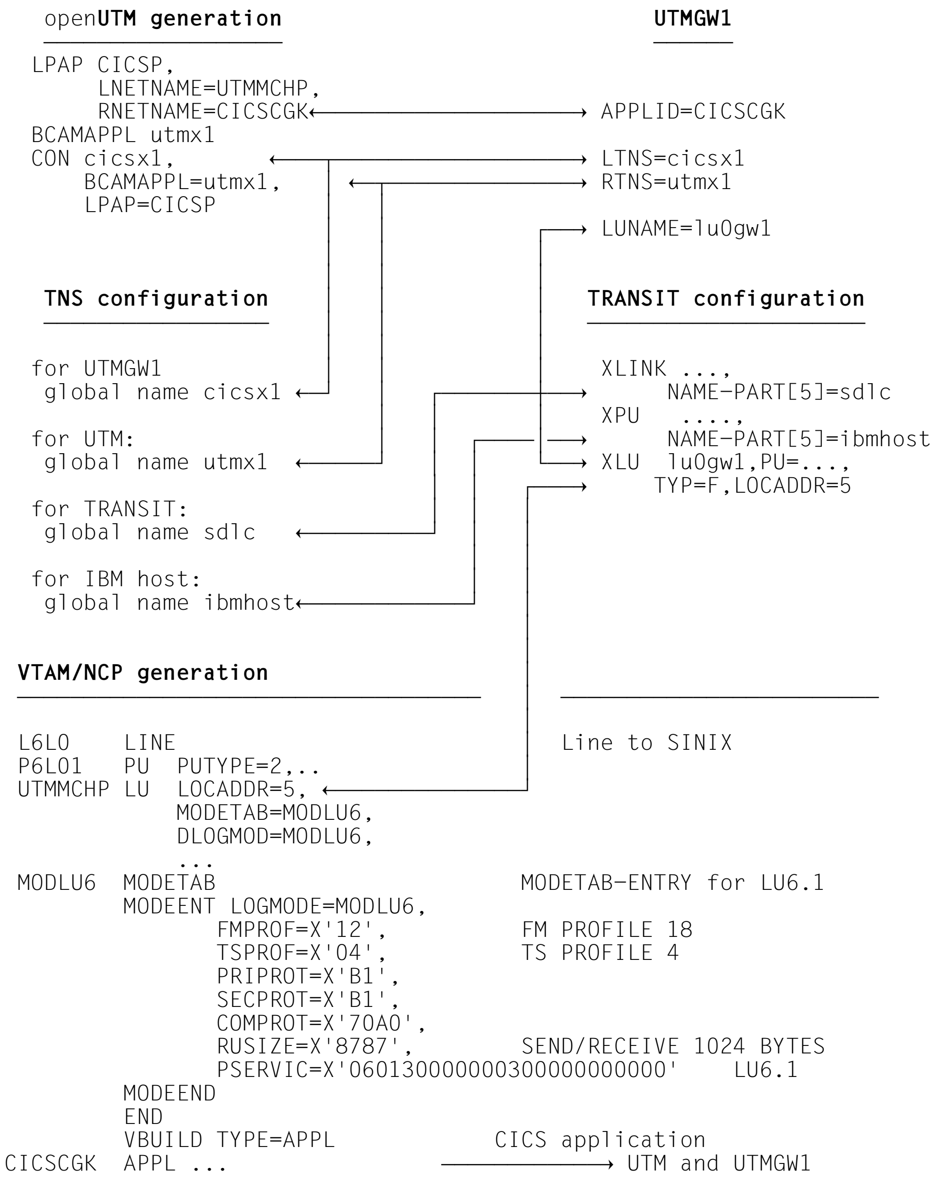
When openUTM is interconnected with CICS, the following components are run through.

```
openUTM → CMX → UTMGW1 → TRANSIT-SERVER → CMX → CCP → NCP/VTAM → CICS
```

Some interconnections don’t need CMX and CCP between TRANSIT-SERVER and NCP/VTAM. Definitions must be made for all these components.

The previous chapter described the dependence between openUTM and CICS. The following example provides a rough overview of the dependencies in the UNIX systems definitions.

The following example applies to a session; openUTM and UTMGW1 are running on one processor.



```
openUTM generation                              UTMGW1

LPAP CICSP,
     LNETNAME=UTMMCHP,
     RNETNAME=CICSCGK  <---------------->  APPLID=CICSCGK
BCAMAPPL utmx1
CON cicsx1,            <---------------->  LTNS=cicsx1
    BCAMAPPL=utmx1,    <---------------->  RTNS=utmx1
    LPAP=CICSP
                                       -->  LUNAME=lu0gw1

TNS configuration                          TRANSIT configuration

for UTMGW1                                  XLINK ...,
 global name cicsx1                              NAME-PART[5]=sdlc
                                            XPU   ....,
for UTM:                                         NAME-PART[5]=ibmhost
 global name utmx1                          XLU  lu0gw1,PU=...,
                                                TYP=F,LOCADDR=5
for TRANSIT:
 global name sdlc

for IBM host:
 global name ibmhost

VTAM/NCP generation

L6L0    LINE                                Line to SINIX
P6L01   PU  PUTYPE=2,..
UTMMCHP LU  LOCADDR=5,
            MODETAB=MODLU6,
            DLOGMOD=MODLU6,
            ...
MODLU6  MODETAB                          MODETAB-ENTRY for LU6.1
        MODEENT LOGMODE=MODLU6,
               FMPROF=X'12',             FM PROFILE 18
               TSPROF=X'04',             TS PROFILE 4
               PRIPROT=X'B1',
               SECPROT=X'B1',
               COMPROT=X'70A0',
               RUSIZE=X'8787',           SEND/RECEIVE 1024 BYTES
               PSERVIC=X'060130000000300000000000'    LU6.1
        MODEEND
        END
        VBUILD TYPE=APPL              CICS application
CICSCGK APPL ...                 ---------> UTM and UTMGW1
```

6.3 Defining CICS transactions

6.3.1 Local transactions

CICS transactions are defined via RDO.

```
DEFINE TRANSACTION <transaction code>
       PROGRAM     <program name>
       INDOUBT     WAIT
        .........
```

The operands have the following meanings:

TRANSACTION <transaction code>
: Transaction code (1-4 characters) for the CICS transaction. This name must match the RTAC operand in the corresponding LTAC statement used for openUTM generation.

PROGRAM <program name>
: Name of a CICS program.

INDOUBT WAIT
: This specification is necessary to achieve proper synchronization with openUTM after session errors occur.

6.3.2 Remote transactions

openUTM transactions are usually not defined in CICS. The transaction code appears only in the programs. Remote dialog transactions are determined via the BUILD ATTACH call and asynchronous programs with START. With the START call, the transaction code can only have a length of 4 bytes. This can be taken into account in openUTM generation or this transaction can be defined via RDO.

```
DEFINE TRANSACTION  <local transaction code>
       REMOTESYSTEM <UTM system ID>
       REMOTENAME   <remote transaction code>
      .........
```

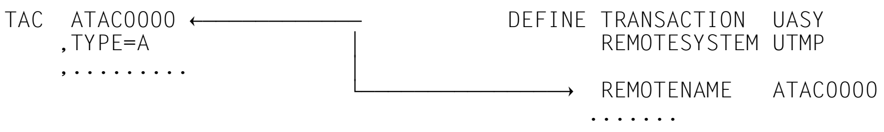
The operands have the following meaning.

TRANSACTION <local transaction code>
: Local transaction code (1-4 characters) for a openUTM service. This name is used in the START call.

REMOTESYSTEM <UTM system ID>
: Name of the openUTM application in the CICS application (see page 216).

REMOTENAME <remote transaction code>
: Transaction code (1-8 characters) for a service in the openUTM application. This name must correspond to the TAC name in the corresponding TAC statement used for openUTM generation.

Example:



```
TAC  ATAC0000                 DEFINE TRANSACTION  UASY
    ,TYPE=A                          REMOTESYSTEM UTMP
    ,.........
                                     REMOTENAME   ATAC0000
                                    .......
```

6.4 CICS programming during interconnection with openUTM

Two types of communication are possible between CICS and openUTM.

– Distributed Transaction Processing (DTP) LU6.1

– Asynchronous Processing LU6.1 with START and RETRIEVE

No other interconnection options offered by CICS can be used (e.g. DTP LU6.2).

The following section provides a rough overview of the special features of openUTM-CICS communication.

6.4.1 Rules and restrictions for CICS programming

The following factors should be taken into consideration when creating CICS programs:

1. openUTM cannot distinguish between a remote CICS application and a remote openUTM application. In other words, openUTM does not know whether a remote service is a CICS conversation or an openUTM service.
2. Commands or command strings in CICS programs must correspond to a call or a sequence of calls on the KDCS program interface to ensure that they can communicate with openUTM services.
3. At the end of a CICS transaction (logical unit of work), selected resources (not all) are logged. In contrast, at the end of an openUTM transaction all modified resources are logged. In other words: openUTM operates fully transaction-oriented, and openUTM logging is "indivisible". CICS conversations that require comprehensive transaction logging must as a result treat the dialog or queued message as logged (PROTECTED).
4. A session between an openUTM service and a CICS conversation is involved in logging and should not be released without a sync point.
5. In openUTM, the job-submitting service and the job-receiving service do not have equal rights when it comes to terminating a session release (bracket): The job-submitting service cannot be the first party to initiate the release of the session (with PEND FI) before its job-receiving service. For this reason, when cooperating with a openUTM service, a CICS conversation can initiate a session release if it is the job receiver.
6. In openUTM, indicators for sync points, session termination or change to SEND or RECEIVE state are sent in the message. CICS commands and their combinations that can cause isolated indicators (i.e. without message contents) to be sent to openUTM should not be used (e.g. WAIT option with SEND in "CICS for MVS/ESA Application Programming Reference“).

7. openUTM interprets the arrival of the transmit authority (change direction) or of "end bracket" as the end of the message. Only after this is it considered a complete message and the destination program is activated. The CICS command CONVERSE or SEND INVITE can be used to add the change direction indicator to the message. The CICS command SEND LAST, RETURN or SEND LAST, SYNCPOINT can be used to add the end bracket indicator to the message.
8. In openUTM, a sync point is always connected with the transmission of a complete dialog message. As a result, a CICS conversation must see to it that the transmit authority (change direction) is transferred to the openUTM partner service in the case of a SYNCPOINT command.
9. openUTM always expects the ATTACH function management header for the addressing of services. The CICS command BUILD ATTACH must be used in the CICS job-submitting conversation. The openUTM transaction code cannot be placed at the beginning of the message.

6.5 CICS commands for CICS job-submitting conversations

The following section describes how an openUTM service is started from a CICS application.

ALLOCATE

A CICS job-submitting conversation uses this command to attempt to assign a session for a job-receiving conversation.

The command has the following syntax:

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

SYSID(name)
: Specifies the name for the remote openUTM application just like it was specified in the CONNECTION definition. Immediately after the session is assigned, the name of the assigned session is located in the EIBRSRCE field of the exec interface blocks (EIB). This name must be specified in all subsequent job-submitting commands (SEND/RECEIVE/CONVERSE) that relate to this session.

PROFILE(name)
: Specifies a certain profile for communication with the partner. PROFILEs are determined during CICS generation (see "CICS for MVS/ESA Resource Definition“).

NOQUEUE
: Means that the program should not wait if a session is not immediately available. A session is "not immediately available" in the following situations:

1. All sessions to the remote openUTM application are busy.
2. Existing sessions are not yet active (bound).
3. The sessions still available are all contention loser sessions (RECEIVECOUNT 1).

If NOQUEUE is not used, CICS adds the request to a queue and the program waits until a session is available.

NOQUEUE should not be used for openUTM(SINIX) since CICS is always the contention loser and hence no session is immediately available.

BUILD ATTACH

The CICS job-submitting conversation uses this command to create an ATTACH header (FMH5) in which the TAC should be entered for the remote openUTM service. This FMH is necessary to initiate openUTM services and is sent to the partner using SEND or CONVERSE.

The command has the following syntax:

```
EXEC CICS BUILD ATTACH  ATTACHID(name)
                        PROCESS(name)
                        RECFM(data-area)
```

ATTACHID(name)
: Defines the name of the ATTACH header. This name is specified in the SEND or CONVERSE command.

PROCESS(name)
: Specifies the transaction code of the remote conversation.

RECFM(data-area)
: data-area is a field (two bytes in length) that contains the record format of the records sent by CICS.

```
x'01'  data in VLVB format
x'04'  unformatted data
```

Other options (see "CICS for MVS/ESA Application Programming Reference“) of this command are irrelevant when it comes to interconnection with openUTM.

SEND

This command creates a dialog message segment.

The command has the following syntax:

```
EXEC CICS SEND [SESSION(name)]
               [ATTACHID(name)]
               FROM(area)
               LENGTH(area)
               INVITE
```

SESSION(name)
: Specifies the name of the session. This is known as the return value (EIBRSRCE) from the ALLOCATE command. In this case, the message is meant for the job receiver (alternate facility). If the option is omitted, the message is meant for the job submitter (principal facility).

ATTACHID(name)
: Specifies the name of the ATTACH-ID as it was defined in the BUILD ATTACH command.

FROM(area) and LENGTH(area)
: Specify the address and the length of the message. The format must correspond to what was specified for RECFM in the BUILD ATTACH command. If VLVB was specified in BUILD ATTACH, then the first 2 bytes of the message are the length field. Several message segments can be sent with one SEND by placing another length field after the first message segment and before the second message segment. If "Chain of RUs“ was specified in BUILD ATTACH (default value), then length fields may not be placed at the beginning of the message.

INVITE
: Explicitly causes the transmit authority (change direction) to be passed to the partner when the message is sent. openUTM interprets the arrival of the transmit authority as the end of the partner’s dialog message.

RECEIVE

This command is used to receive a message sent from a openUTM service.

The command has the following syntax:

```
EXEC CICS RECEIVE [SESSION(name)]
                  [INTO(area) | SET(pntr)]
                  LENGTH(area)
                  [MAXLENGTH(area)]
                  [NOTRUNCATE]
```

SESSION(name)
: Specifies the name of the session. This is known as the return value (EIBRSRCE) from the ALLOCATE command. In this case, the message comes from the job receiver (alternate facility), and therefore from the UTM application program. If the option is omitted, a message should be received from the job submitter (principal facility).

INTO(area)
: Specifies the address of the memory area in which CICS is to write the messages. Messages received by UTM normally contain a length field in the first two bytes, i.e. openUTM usually uses the VLVB format. If the UTM application program has sent several message segments using MPUT NT, then they are received in a RECEIVE command. Every message segment has a length field in the first two bytes in this case. If the UTM application program has set KCDF to a value other than 0 or KCMF to a value other than the space character in MPUT, then openUTM still sends a function management header 4 (FMH4) before the message, and the data does not have a leading length field. The CICS program can query whether the data has arrived with or without a length field using EXTRACT ATTACH. See page 231 for more detailed information on EXTRACT ATTACH. The CICS program can also recognize whether the incoming data contains an FMH4 with the INBFMH indicator after the RECEIVE command has been executed. See page 231 for more information on FMH4.

NOTRUNCATE
: If NOTRUNCATE is not specified, a message with a maximum length of MAXLENGTH is delivered. The rest is rejected. If NOTRUNCATE is specified, other parts can be requested with RECEIVE.

See "CICS for MVS/ESA Application Programming Reference“ for the remaining parameters.

CONVERSE

This command ensures that a message is sent to the partner and that the program waits for the partner's answer. This command has the same effect as the command string SEND INVITE, RECEIVE.

The command has the following syntax:

```
EXEC CICS CONVERSE [SESSION(name)]
                   [ATTACHID(name)]
                   [FROM(area)]
                   FROMLENGTH(area)
                   [INTO(area) | SET(pntr)]
                   TOLENGTH(area)
                   [MAXLENGTH (area)]
                   [NOTRUNCATE]
```

The parameters have the same meaning as those described for the SEND or RECEIVE command.

EXTRACT ATTACH

This command is used to read information from the received ATTACH header (EIBATT indicator in EIB after RECEIVE).

The command has the following syntax:

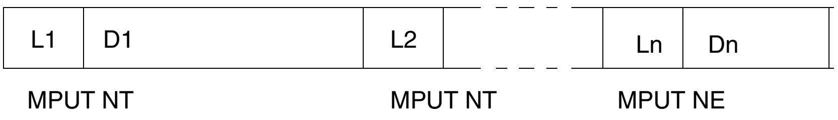
```
EXEC CICS EXTRACT ATTACH       RECFM(data-area)
```

An FMH5 sent by openUTM only contains information regarding the message format. In other fields (see "CICS for MVS/ESA Application Programming Reference“) no information is supplied after the call.

RECFM(data-area)
This parameter describes a field (two bytes in length) which contains information for the deblocking algorithm in the CICS application program in its least significant byte.

– The value X’01’ means that the data is available in VLVB format (variable length, variable blocked). openUTM always sends data in VLVB format when KCDF=0 and KCMF=space have been specified in MPUT.-

Every message segment created with an MPUT NT/NE contains a 2-byte length field.:



The deblocking of the message segments must occur in the CICS program.

– The value X’04’ means that the data is in the "Chain of RUs“ format. openUTM sends the data in this format when KCDF was specified as a value other than 0 or KCMF was specified as a value other than the space character in MPUT.

In this case, openUTM places a function management header 4 (FMH4) in front of the message. This FMH4 contains the information sent with KCDF and KCMF in the following manner:

```
Byte    0     X’12’ (length of the FMH4)
Byte    1     X’04’ (code for FMH4)
Bytes  2- 5   no meaning
Byte    6     X’08’ (length of the FMH4BN)
Bytes  7-14   FMH4BN corresponds to KCMF
Byte    15    X’02’ (length of the FMH4BDT)
Bytes  16-17  FMH4BDT corresponds to KCDF
```

Every message segment created with MPUT NT/NE must be picked up in the CICS program with its own RECEIVE command.

SYNCPOINT command

This command is used for the following:

1. To request a sync point and
2. To confirm a sync point

The command string SEND, SYNCPOINT requests a sync point. The transaction is then in the PET (preliminary end of transaction) state.

The command string RECEIVE, SYNCPOINT confirms a received request to set a sync point (EIBSYNC indicator in EIB). The transaction is then in ET (end of transaction) state.

The command has the following syntax:

```
EXEC CICS SYNCPOINT
```

RETURN command

The RETURN command implies the transmission of messages that have not yet been sent with a sync point request to the partner and subsequent session release. The RETURN command is valid only if the openUTM-AN service has already returned PEND FI (EIBFREE indicator in EIB).

The command has the following syntax:

```
EXEC CICS RETURN
```

6.6 CICS commands for CICS job-receiving conversations

RECEIVE

This command is used to receive a message sent by an openUTM job-submitting service. Immediately after initialization by a remote openUTM service, a CICS conversation is in the RECEIVE state and must use this command to read in the message.

The command has the following syntax:

```
EXEC CICS RECEIVE {INTO(area) | SET(pntr)}
                  LENGTH(area)
                  [MAXLENGTH(area)]
                  [NOTRUNCATE]
```

The NOTRUNCATE option ensures that the rest of the message is made available to a subsequent RECEIVE command if the message is longer than specified in MAXLENGTH. See page 229 for the other options.

EXTRACT ATTACH

This command is used to read information from the received ATTACH header (EIBATT indicator in EIB).

The command has the following syntax:

```
EXEC CICS EXTRACT ATTACH
          PROCESS(data-area)
          RECFM(data-area)
```

An FMH5 sent by openUTM contains the CICS transaction code (PROCESS option) and information regarding the message format used. In other fields (see "CICS for MVS/ESA Application Programming Reference“) no information is supplied after the call.

PROCESS option
: After the call, this contains the transaction code as it was sent by openUTM to CICS in FMH5, i.e. the RTAC of the openUTM generation.

RECFM option
: See page 231.

BUILD ATTACH

With this command an ATTACH header (FMH5) is created by the CICS job-receiving conversation that contains the message format for the messages thereafter for openUTM.

The command has the following syntax:

```
EXEC CICS BUILD ATTACH  ATTACHID(name)
                        RECFM(data-area)
```

ATTACHID(name)
: Defines the name of the ATTACH header. This name is entered in a SEND or CONVERSE command.

RECFM(data-area)
: The data area is a two-byte field that contains the record format of the records sent by CICS.

```
x'01'  Data in the VLVB format
x'04'  Data in the “Chain of RUs“ format
```

The other options (see "CICS for MVS/ESA Application Programming Reference“) for this command do not have any meaning for interconnection with openUTM.

SEND

The SEND command is used to create a dialog message segment.

The command has the following syntax:

```
EXEC CICS SEND  [ATTACHID(name)]
                FROM (area)
                LENGTH(area)
                [INVITE | LAST]
```

The message is meant for the job submitter (principal facility).

The INVITE operand explicitly causes the transmit authority (change direction) to be passed to the partner when the message is sent. The dialog is continued during this process. openUTM must then respond with MGET-MPUT (no PEND FI).

If data is sent by the CICS application program that is longer than the generated RU size, then this data is received by the UTM application program in several message segments. To avoid this, an ATTACHID parameter must be specified in the SEND command that refers to a previous BUILD ATTACH. The message format can then be defined in this BUILD ATTACH.

CONVERSE

The CONVERSE command is used to send a message to the partner while simultaneously waiting for the answer. This command has the same effect as the command string SEND INVITE,RECEIVE.

The command has the following syntax:

```
EXEC CICS CONVERSE FROM(area)
                   FROMLENGTH(area)
                   [ATTACHID(name)]
                   {INTO(area) | SET(pntr)}
                   TOLENGTH(area)
                   [MAXLENGTH (area)]
                   [NOTRUNCATE]
```

SYNCPOINT

See page 232.

RETURN

The RETURN command in a CICS job-receiving conversation implies the transmission of messages that have not yet been sent with a sync point request (implicit sync point for protected resources) to the partner, and the subsequent session release.

The command has the following syntax:

```
EXEC CICS RETURN
```

6.7 Comparison with KDCS calls

The following section attempts to compare the semantics of CICS commands with KDCS calls.

Addressing a remote dialog service

```
ALLOCATE                                  APRO DM
BUILD ATTACH
```

Communication without sync point:

```
SEND                                      MPUT
RECEIVE                                   PEND KP
                                          .
                                          INIT
                                          MGET
```

Communication and end of transaction request:

```
SEND                                      MPUT
SYNCPOINT                                 PEND RE
RECEIVE                                   .
                                          INIT
                                          MGET
```

Communication and end of transaction and service request:

```
SEND                                      MPUT
RETURN                                    PEND FI
```

Confirmation of end of transaction after message receipt:

```
                                          INIT
RECEIVE (EIBSYNC=X'FF')                   MGET (KCTAST='P')
SEND                                      MPUT
SYNCPOINT/RETURN                          PEND RE/FI
```

If the end of transaction (a sync point) is requested from the predecessor, CICS sets EIBSYNC=X’FF’ and openUTM sets the transaction state KCTAST=’P’. After these values are received, a command must be issued to write a sync point (SYNCPOINT/RETURN or PEND RE/FI).

6.8 Programming examples of CICS-openUTM communication

1. Starting a openUTM service from a CICS application program

```
--->
    RECEIVE Client
    ALLOCATE
    BUILD ATTACH
    SEND INVITE SESSION(...)
    RECEIVE SESSION(...)
                             -- FMH5+data ---->
                                                 INIT
                                                 MGET
                                                 MPUT NE
                                                 PEND FI
                             <- FMH5+data+SRQ -
    Return of the RECEIVE
    SEND Client
    RETURN
<---                         -- SRSP ---------->
```

2. Starting a CICS conversation from a openUTM application program

```
--->
    INIT
    MGET Client
    APRO DM KCPI=“>A“
    MPUT NE KCRN=“>A“
    PEND KP
                             -- FMH5+data ---->
                                                 RECEIVE
                                                 SEND LAST
                                                 RETURN
                             <- data+SRQ ------
    INIT
    MGET NT KCRN=“>A“
    MPUT NE Client
    PEND FI
<---                         -- SRSP ---------->
```

```
SRQ:  Syncpoint Request
SRSP: Syncpoint Response
```

6.9 CICS commands for queued jobs

With CICS, the START and RETRIEVE commands are used to send and receive queued jobs.

START command

This command is used to create a queued job.

The command has the following syntax:

```
EXEC CICS START
          [SYSID(name)]
          TRANSID(name)
          FROM(area)
          LENGTH(area)
          NOCHECK
          PROTECT
```

The parameters have the following meanings:

SYSID(name)
: Specifies the name of the openUTM application.

TRANSID(name)
: Specifies the TAC for the asynchronous program with openUTM. With CICS, the name must not be longer than 4 characters.

  A local TAC name can also be used for a "remote transaction" with CICS (see page 223). In this case, the SYSID option is not used since the partner application was already determined through the generation.

NOCHECK
: An answer is not expected. If no session is available, the message is added to a queue in a CICS environment. If a session becomes available, the message is sent.

PROTECT
: The remote asynchronous service should not be started before the CICS conversation has issued a SYNCPOINT or RETURN command.

RTRANSID and RTERMID support is described starting on page 255. openUTM does not support additional options such as TERMID and FMH (see "CICS for MVS/ESA Application Programming Reference“).

Within a CICS transaction (LUW = logical unit of work), only a Start command can occur for a openUTM application.

RETRIEVE

The RETRIEVE command is used to read a queued message.

The command has the following syntax:

```
EXEC CICS RETRIEVE
          {INTO(area) | SET(pntr)}
```

RTRANSID and RTERMID support (see "CICS for MVS/ESA Application Programming Reference“) is described starting on page 255.

Examples of the exchange of queued jobs

1. Starting an openUTM queued job from within a CICS application program

```
START
SYNCPOINT
                         -- FMH6+data+SRQ ->
                         <- SRSP ------------
                                             INIT
                                             FGET
                                             PEND FI
```

2. Starting a CICS queued job from within a UTM application program

```
INIT
APRO AM KCPI=“>A“
FPUT NE KCRN=“>A“
PEND FI
                         -- FMH6+data+SRQ ->
                         <- Data+SRQ -------
                                             RETRIEVE
                                             RETURN
```

```
SRQ:  Syncpoint Request
SRSP: Syncpoint Response
```

6.10 Notes regarding openUTM-CICS programming

For openUTM services that wish to communicate with CICS conversations, the following programming notes should be observed.

1. Specification of KCMF (format name) and KCDF (device features)

   If the format name and/or KCDF are specified in openUTM programs, they are transferred via FMH4. FMH4 handling must be provided for in the CICS application program.

2. Message segments (MPUT NT/FPUT NT)

   Message segments are usually transferred by openUTM in VLVB records. All segments with their associated length field are sent in one or more RUs (depending on the NB size). CICS programs must provide for their own deblocking.

   However, if a format name and/or KCDF is specified in the UTM application program, then openUTM sends the message segments in the "Chain of RUs“ format.

3. Service restart

   openUTM assumes that processing will continue at the last sync point if communication is interrupted (e.g. loss of connection).

   CICS cannot provide for this function. This leads to service status ‘Z’ in the openUTM program after the restart.

4. Converting user data

   User data must be converted from ASCII to EBCDIC and vice-versa for interconnection between openUTM under UNIX systems or Windows systems and CICS. If the user data consists exclusively of printable characters, then the conversion can be handled using the parameter MAP=SYSTEM in the SESCHA statement of openUTM. Note, however, that this converts from BS2000/OSD-EBCDIC to ASCII and vice-versa, but a different EBCDIC is usually used on IBM systems. Language-specific characters (such as the German umlauts) and some special characters will be incorrectly converted. For this reason it is recommended to carry out the ASCII-EBCDIC conversion in the openUTM or CICS application program. Conversion should also be carried out in the application program when interconnecting between openUTM under BS2000/OSD and CICS for the reasons listed above.

Definitions for openUTM-CICS programming

Between the openUTM and CICS partners, the following items need to be defined for programming purposes.

- transaction names
- data format, record format (undefined/VLVB)
- synchronous (LU6.1) / asynchronous
- command string
- values in FMH6 (asynchronous)

7 openUTM-IMS interconnection via LU6.1

This chapter covers openUTM-IMS interconnection via LU6.1, including the associated generation and programming.

For the IMS user, generation and programming of openUTM-IMS interconnection is largely identical to the procedure for IMS-CICS interconnection (see “IMS/ESA Installation Volume 2: System Definition and Tailoring” and “IMS/ESA Application Programming: Transaction Manager”).

7.1 IMS generation for openUTM-IMS sessions

This section describes the most important IMS macros required to generate openUTM-IMS interconnection.

COMM macro

This macro describes the basic communications requirements independent of the terminal type. It has to be specified exactly once.

```
COMM   RECANY=(number,size),
       APPLID=username,
       EDTNAME=name,
       ..........,
       other parameters
```

The operands have the following meanings:

RECANY=(number,size)
"number" specifies the number of VTAM RECEIVE ANY buffers available in the IMS system. "size" specifies the size of these buffers. The value of size must be larger than or equal to the value of TRMSGLTH in the MAX statement used for openUTM generation. The permissible size values are listed in "IMS/ESA Installation Volume 2: System Definition and Tailoring".

APPLID=username
: Name (1-8 characters) under which the IMS application is known in the SNA network (SNA netname). This name must match the RNETNAME operand in the corresponding LPAP statement used for openUTM generation.

EDTNAME=name
: Name (1-8 characters) under which the ISC edit process is known in the IMS system. This name must match the DPN operand in the corresponding SESCHA statement used for openUTM generation.

TYPE macro

This macro is used to introduce the description of a group of VTAM terminals of the same type. It is followed by the TERMINAL and NAME macros for the terminals in the group.

```
TYPE    UNITYPE=LUTYPE6,
        ...,
        other parameters
```

The operand has the following meaning:

UNITYPE=LUTYPE6
: Defines an LU6.1 device type.

TERMINAL macro

This macro defines the physical and logical characteristics of a VTAM node of the type specified in the preceding TYPE macro.

```
TERMINAL NAME=nodename,
         MSGDEL=SYSINFO,
         OPTIONS=FORCSESS,
         COMPT1=(SINGLE1 [,DPM-Bn],IGNORE),
         SEGSIZE=size,
         OUTBUF=buffer size,
         SESSION=n,
         ...........,
         other parameters
```

The operands have the following meanings:

NAME=nodename
: Name (1-8 characters) which is used to define the SNA network name (SNA netname) for the openUTM application. This name must match the LNETNAME operand in the corresponding LPAP statement used for openUTM generation.

MSGDEL=SYSINFO
: Specifies which message types will not be sent. SYSINFO must be specified for openUTM links.

OPTIONS=(TRANRESP,FORCSESS)
: FORCSESS specifies session restart. FORCSESS must be specified for openUTM links.

  TRANRESP is required to maintain a dialog with IMS transactions (default).

COMPT1=(SINGLE1[,DPM-Bn],...)
: SINGLE1 specifies the bracket protocol for output. SINGLE1 must be specified for openUTM links.

  DPM-Bn specifies whether MFS (Distributed Presentation Management) is available. If this parameter is omitted, VLVB is assumed. In this case, the first 8 characters of a message to be sent to a openUTM program unit must contain the openUTM transaction code (possibly with trailing blanks). The complete message (including transaction code) is made available to the openUTM program unit in the message area.

SEGSIZE=size
: "size" specifies the size of an input segment. The value must at least be equal to the maximum message segment length in the openUTM application programs.

OUTBUF=buffer size
"buffer size“ specifies the size of the output buffer. This value must be smaller than or equal to the value of TRMSGLTH in the MAX statement used for openUTM generation. Permissible size values are listed in the manual "IMS/ESA Installation Volume 2: System Definition and Tailoring“.

SESSION=n
n specifies the number of parallel sessions for a node. For openUTM(SINIX), parallel sessions are not possible (n=1).

VTAMPOOL macro

This macro must be specified in order to use parallel sessions. The macro appears at the beginning of the definition of one or several LU6 subpools. These are defined using SUBPOOL macros, each of which may be followed by one or more NAME macros.

```
VTAMPOOL
```

This macro has no operands.

SUBPOOL macro

This macro defines an LU6 subpool. A SUBPOOL macro is required for each session to be set up dynamically.

```
SUBPOOL NAME=subpool-name,
        MSGDEL=SYSINFO
```

The operands have the following meanings:

NAME=subpool-name
"subpool-name“ matches the RSES operand in the corresponding LSES statement used for openUTM generation.

MSGDEL=SYSINFO
This parameter must match the MSGDEL operand in the TERMINAL macro.

NAME macro

This macro defines a logical terminal name for a physical terminal. This name is used to address the openUTM session in the IMS program.

Specifying the NAME macro after the SUBPOOL macro means that the logical terminal will be created dynamically. This is necessary if parallel sessions are to be used (TERMINAL SESSION=n, where n> 1).

```
NAME     lterm,
         COMPT=1,
         ..........,
         other parameters
```

The operands have the following meanings:

lterm defines a logical terminal name (1-8 characters).

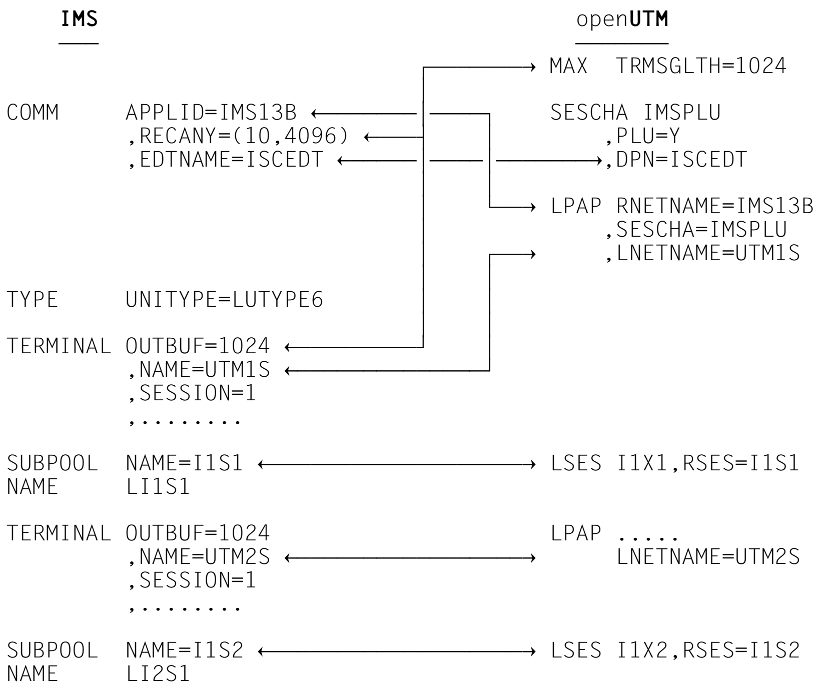
COMPT=1
: Specifies the output components for this terminal. The value 1 corresponds to the COMPTn operand in the TERMINAL macro.

7.1.1 Examples of dependencies in openUTM/IMS generation

The IMS and openUTM users have to establish definitions for the parameters marked with arrows.

Example: openUTM under UNIX systems or openUTM under BS2000/OSD example using TRANSIT-CLIENT

IMS is the PLU, openUTM the SLU, 2 sessions



```
    IMS                                                 openUTM

                                                  →  MAX  TRMSGLTH=1024

COMM     APPLID=IMS13B ←                             SESCHA IMSPLU
         ,RECANY=(10,4096) ←                             ,PLU=Y
         ,EDTNAME=ISCEDT ←                           →   ,DPN=ISCEDT

                                                  →  LPAP RNETNAME=IMS13B
                                                         ,SESCHA=IMSPLU
                                                  →      ,LNETNAME=UTM1S

TYPE     UNITYPE=LUTYPE6

TERMINAL OUTBUF=1024 ←
         ,NAME=UTM1S ←
         ,SESSION=1
         ,........

SUBPOOL  NAME=I1S1 ←                              →  LSES I1X1,RSES=I1S1
NAME     LI1S1

TERMINAL OUTBUF=1024                                 LPAP .....
         ,NAME=UTM2S ←                            →       LNETNAME=UTM2S
         ,SESSION=1
         ,........

SUBPOOL  NAME=I1S2 ←                              →  LSES I1X2,RSES=I1S2
NAME     LI2S1
```

Session setup

In IMS, the LSES name (e.g. I1X1) is only needed when the operator initializes the session using the /OPN command.

For example, the session containing the SUBPOOL I1S1 is set up using the following call:

```
/OPN NODE UTM1S SUBPOOL I1S1 ID I1X1
```

The assignment between TERMINAL and SUBPOOL is not fixed with IMS. Instead, it is dynamic, based on the connection establishment data.

7.2 TRANSIT generation for openUTM-IMS sessions

The statements made about TRANSIT regarding openUTM-CICS interconnection also apply to openUTM-IMS interconnection.

See page 220 for details.

7.3 Generating IMS transactions

TRANSACT macro

This macro is used to define IMS transactions. It corresponds to the TAC statement used for openUTM generation.

```
TRANSACT CODE=transaction-code,
         MSGTYPE=(MULTSEG,RESPONSE),
                .........
```

The operands have the following meanings:

CODE=transaction-code
: Name (1-8 characters) for the IMS transaction. This name must match the RTAC operand in the corresponding LTAC statement used for openUTM generation.

MSGTYPE=(MULTSEG,RESPONSE)
: MULTSEG specifies that an input message can consist of several segments. RESPONSE is used for dialog programs, NONRESPONSE for asynchronous programs.

The SPA parameter must not be used.

Examples

```
LTAC  IMSDIA1P                   TRANSACT
     ,RTAC=DIALOG1 <------------------>  CODE=DIALOG1
     ,TYPE=D                            ,MSGTYPE=(MULTSEG,RESPONSE)
     ,.........                         ,........

LTAC  IMSASY1S                   TRANSACT
     ,RTAC=ASYNC1  ------------------>  CODE=ASYNC1
     ,TYPE=A                            ,MSGTYPE=(MULTSEG,NONRESPONSE)
     ,.........                          ,........
```

7.4 IMS programming for links to openUTM

This section offers an introductory overview of openUTM-IMS communication.

7.4.1 IMS interconnection options

Synchronous jobs (dialog)

Due to its structure (message queues), IMS does not allow synchronous transactions if IMS is the job submitter (front-end), i.e. if a request from the terminal is passed from IMS to openUTM and a synchronous response from openUTM is expected.

If IMS is the job receiver (back-end), it is possible to establish a dialog between openUTM and IMS. In IMS, the MSGTYPE parameter in the TRANSACT macro (RESPONSE or NONRESPONSE) is used to specify whether the IMS transaction is processed synchronously or asynchronously. In addition to this internal specification, the type of processing can also be determined from the form of the incoming message. If openUTM starts dialog traffic (MPUT), IMS sends a synchronous response even if NONRESPONSE was specified in the TRANSACT macro.

Queued jobs

Due to dialog restrictions, asynchronous data traffic is often used for IMS links. Synchronizing the responses and the corresponding requests is left to the user. However, it is possible to automate the routing of responses. The destination for the response is supplied in the request, making it possible to establish a dummy dialog. The routing information is stored in function management header 6 (FMH6) and has to be evaluated by the partner. Access to the data in FMH6 from openUTM is described in chapter “LU6.1 dummy dialogs between asynchronous services” on page 255ff.

7.4.2 IMS calls

IMS programs send and receive messages using the following calls:

GU <I/O-PCB>,<I/O-area>
: Receives the first or only message segment in the <I/O-area>.

GN <I/O-PCB>,<I/O-area>
: Receives any additional message segments in the <I/O-area>.

CHNG <Alt-PCB>,<destination name>
: Changes the destination address to <destination name> (lterm from NAME macro used for generation).

ISRT <I/O-PCB>/<Alt-PCB>,<I/O-area>,<MOD-Name>
: Sends messages from the <I/O-area>.

RETURN
: Ends the transaction and requests a sync point.

When a message is received and sent, a PCB (program communication block) is passed. This PCB stores information such as status, partner name (lterm) etc. When sending a message to the partner that called a transaction, the I/O PCB is used. Otherwise, an alternate PCB is used in which a new destination address is set using the CHNG call.

The transaction code is placed in front of the message or defined using an MOD (message output description) name in the MFS (message format service).

7.4.3 Comparison to KDCS calls

– The GU and GN calls for the IOPCB correspond to the KDCS calls MGET/FGET. The GN call provides a display indicating if additional message segments exist.

– A sequence of MPUT/FPUT NT and MPUT/FPUT NE must be read by IMS using GU and GN.

– The ISRT call corresponds to the KDCS calls MPUT (IOPCB) and FPUT (alternate PCB).

– The ROLL call corresponds to the KDCS call PEND FR.

– The ROLLB call corresponds to the KDCS call PEND RS.

– Program end implies a sync point request and corresponds to PEND FI.

7.4.4 Examples of openUTM-IMS communication

1. Starting an IMS job from a UTM application program

```
--->
    INIT
    MGET Client
    APRO DM KCPI=“>A“
    MPUT NE KCRN=“>A“
    PEND KP
                             -- FMH5+data ---->
                                                   GU       (IO/PCB)
                                                   ISRT
                                                   RETURN
                             <- Data+SRQ ------
    INIT
    MGET NT KCRN=“>A“
    MPUT NE Client
    PEND FI
<---                         -- SRSP ---------->
```

2. The same, but with PEND RE in the UTM application program

```
--->
    INIT
    MGET Client
    APRO DM KCPI=“>A“
    MPUT NE KCRN=“>A“
    PEND RE
                             -- FMH5+data+SRQ ->
                                                   GU       (IO/PCB)
                                                   ISRT
                                                   RETURN
                             <- Data+SRSP ------
                             <- SRQ -------------
    INIT
    MGET NT KCRN=“>A“
    MPUT NE Client
    PEND FI
<---                         -- SRSP ---------->
```

```
SRQ:  Syncpoint Request
SRSP: Syncpoint Response
```

The second variant differs from the first in its transaction-oriented logic. In the first variant, the entire run consists of a single transaction. In the second version, any database changes made after the MGET KCRN=>A will not belong to the same transaction as the database changes made on the IMS side.

If database changes are to be made on both sides, then the second variant must be selected. If the transaction is rolled back in openUTM in the first variant after the MGET KCRN=>A (e.g. by PEND ER), then IMS is no longer able to roll back the database changes.

7.4.5 Examples of the exchange of queued jobs

1. Starting an openUTM queued job from an IMS application program

```
ISRT   (Alt.-PCB)
RETURN
                         -- FMH6+data+SRQ ->
                         <- SRSP ------------
                                             INIT
                                             FGET
                                             PEND FI
```

2. Starting an IMS queued job from a UTM application program

```
INIT
APRO AM KCPI=“>A“
FPUT NE KCRN=“>A“
PEND FI
                         -- FMH6+data+SRQ ->
                         <- Data+SRQ -------
                                             GU    (IO/PCB)
                                             RETURN
```

```
SRQ:  Syncpoint Request
SRSP: Syncpoint Response
```

7.4.6 Notes on openUTM-IMS programming

When IMS is the job submitter, then the name of the calling openUTM transaction is normally sent with the data. The openUTM user must consider this in the input area.

Similar to interconnection with CICS, the problem of converting the database must also be considered when interconnecting with IMS (see page 240).

Additional notes regarding the exchange of queued jobs can be found starting on page 255.

Definitions for openUTM-IMS programming

Between the two partners, openUTM and IMS, the following items need to be defined for programming purposes:

- transaction names
- manner in which the transaction code is passed
- data format (VLVB or Chain of RUs)
- synchronous/asynchronous
- values in FMH6 (asynchronous)

8 LU6.1 dummy dialogs between asynchronous services

It is often difficult to program an LU6.1 interconnection between openUTM and IMS because synchronous services cannot be started by IMS in a UTM application. The LU6.1 protocol therefore provides the capability to create dummy dialogs between asynchronous services. These dummy dialogs can also be used in UTM applications using a special KDCS enhancement that is not described in the normal openUTM manuals. This enhancement is used when linking openUTM-IMS and openUTM-CICS.

Asynchronous services are principally addressed in LU 6.1 with the SCHEDULER Function Management Header FMH6. The following fields are important when using this header:

DPN Destination Process Name
- transaction code in the job-receiving application (openUTM, CICS)
- name of an editor evaluating the message, or format name in the job-receiving application (IMS).

PRN Primary Resource Name
- transaction code or LTERM name in the job-receiving application (IMS)
- LTERM name in the job-receiving application (CICS).

RDPN Return Destination Process Name
- DPN in the local application.

RPRN Return Primary Resource Name
- PRN in the local application.

The RDPN/RPRN fields in FMH6 can be used to maintain a dummy dialog between asynchronous services. A job submitter sends a message containing RDPN/RPRN to a queued job-receiving service which sends the result to the asynchronous service in the job-submitting application, as specified by RDPN/RPRN.

In CICS, the RDPN/RPRN fields can be assigned using the RTRANSID/RTERMID parameters in the START command. Using the RETRIEVE command, a CICS application program can retrieve the values of RDPN/RPRN from the input FMH into RTRANSID/RTERMID.

IMS handles the assignment and evaluation of the data in FMH6 partly automatically or using the Message Format Service (MFS) component.

In openUTM, there are special function calls for openUTM application programs that make it possible to send a message to the job-submitting application destination specified in RDPN/RPRN and to assign RDPN and RPRN in output messages.

8.1 Program interfaces for openUTM

The function call

```
INFO GN (get names)
```

can be used by an asynchronous service started by a remote application to retrieve the values of RDPN/RPRN from the input FMH.

The function call

```
APRO IN (insert names)
```

can be used to address an asynchronous service in a remote application; fields DPN, PRN, RDPN and RPRN can be assigned explicitly for the output FMH. In this case, APRO IN replaces APRO AM.

For the calls, a data area must be specified in addition to the KDCS parameter area.

The data area has the following structure:

Field name	**Meaning**	
KCDPN	Destination Process Name	Destination in the remote application
KCPRN	Primary Resource Name	
KCRDPN	Return Destination Process Name	Destination in the local application
KCRPRN	Return Primary Resource Name	

Each of the fields has a length of 8 bytes.

When calling APRO IN, the UTM application program has to supply the data area with values. After a successful INFO GN call, the fields have been assigned correct values for a reply in a dummy dialog by openUTM.

INFO GN

The following table shows the required specifications in the KDCS parameter area when INFO GN is called:

KCOP	KCOM	KCLA
"INFO"	"GN"	32

All other parameter fields must be set to binary null.

COBOL format for the call:

```
CALL "KDCS" USING <parm1>, <parm2>
```

<parm1>: KDCS parameter area
<parm2>: data area

openUTM returns the following:

- If KCRCCC = 000 is specified, the KCRDPN and KCRPRN fields in the data area <parm2> contain spaces, KCDPN and KCPRN contain the destination in the other application, taken from the input message.
- KCRLM contains the length of the data area (32). If KCRCCC>=40Z applies, the length is 0.
- The KCRCCC return field can contain one of the following values:

000	The function has been executed.
40Z	The function could not be executed. – The names do not exist (KCRCDC contains KD20).
41Z	The call is not allowed at this location. – INFO GN was issued in a dialog service. – INFO GN was issued in a service that was not started by a remote application.
42Z	KCOM is invalid.
43Z	KCLA is invalid.
47Z	The address of the data area is missing or invalid.
49Z	Parameter fields not used by this call have not been set to binary null.
71Z	No INIT has been issued during the UTM application program run.

After a successful INFO GN call, the values contained in the data area <parm2> may be used to send an answer to the partner program in CICS or IMS. In this case, the partner program has to be addressed by APRO IN and spaces in KCRN (see below).

APRO IN

The following table shows the required specifications in the KDCS parameter area when APRO IN is called:

KCOP	KCOM	KCLM	KCRN	KCPA	KCPI
"APRO"	"IN"	32	LTAC name	LPAP name or spaces	Service ID
			spaces	LPAP name	

All other parameter fields must be set to binary null.

All fields in the data area must be assigned; spaces indicate that the name was not specified. If a field in the data area contains characters other than spaces, openUTM will accept it as a valid name. No checking is done to determine if a field contains only alpha-numerical characters or if KCRDPN/KCRPRN, for example, is a destination in the local application.

Addressing the job submitter using APRO IN can be handled as follows:

1. Addressing via KCRN (LTAC), as with APRO AM
   In this case, KCDPN and KCPRN must contain spaces.
2. Addressing via KCDPN and KCPRN
   KCRN must contain spaces, and KCPA the LPAP name. This type of addressing is useful after a successful INFO GN.
3. Addressing via the message
   KCRN, KCDPN and KCPRN contain spaces, and KCPA contains the LPAP name. The first 8 characters in FPUT/DPUT contain the transaction code.

COBOL format for the call:

```
CALL "KDCS" USING <parm1>, <parm2>
```

<parm1>: KDCS parameter area

<parm2>: data area

openUTM returns:

In addition to the return codes described in the “Programming openUTM Applications with KDCS for COBOL, C and C++“, the KCRCCC return field can contain one of the following values:

44Z	When KCDPN and/or KCPRN are used for addressing, KCRN does not contain spaces (KCRCDC contains KD21).
47Z	The address of the data area is missing or invalid.
49Z	Parameter fields not used by this call have not been set to binary null.

If KCRN is not used for addressing, openUTM supplies DPN and PRN in the output FMH with the values from the data area. DPN generation in the SESCHA statement is ignored.

⚠ The APRO IN call causes the loss of data protection functions if addressing is not handled via KCRN.

For the data area, there are no data structures created by openUTM.

8.2 Passing FMH6 parameters in openUTM, IMS and CICS

This section provides a more detailed description of the assignment and evaluation options for the parameters in FMH6 which are available in the various systems for addressing the respective partner.

8.2.1 Transaction code handling in openUTM

The transaction code can be passed to openUTM as follows:

1. in the DPN field of FMH6
2. in the PRN field of FMH6
3. in the first 8 bytes of the data.

Upon receipt from the remote system, openUTM determines the transaction code as follows:

1. value in DPN, if assigned, and there is no specification in SESCHA DPN=
2. value in PRN, if assigned, and SESCHA DPN=name is specified
3. from the first 8 bytes of the data.

openUTM sends the transaction code to the remote system as follows:

1. if addressing is handled via APRO IN, assignment corresponds to the parameter area for the call (see APRO IN and INFO GN)
2. in the DPN field if there is no specification in SESCHA DPN=
3. in the PRN field if SESCHA DPN=name is specified.
   In this case, DPN in FMH6 contains the name from SESCHA DPN

8.2.2 Transaction code handling in IMS

IMS normally sends the transaction code in the data. DPN is assigned to the name of ISC Edit (iscedt-name). For the openUTM user, this means the following:

Specification of SESCHA DPN=iscedt-name and use of the transaction code in the definition of the input area. openUTM does not remove the transaction code from the data. If DPN=iscedt-name is not specified, iscedt-name can be used as TAC. This name can then be used to execute initially common functions for several transactions before the actual transaction (name contained in the data) is run.

When responding to an asynchronous request, IMS acts like openUTM after INFO GN. For output, IMS sets the RDPN/RPRN input parameters to the DPN/PRN fields.

If IMS is to start a dummy dialog, the RPRN field has to be assigned to the transaction code of the receiver in IMS. By default, this field contains the name of the calling terminal or program. Other values must be assigned via MFS.

8.2.3 Transaction code handling in CICS

CICS can assign the fields for queued jobs using parameters from the START call.

```
START TRANSID(T1)      ⟶ DPN
      TERMID (T2)      ⟶ PRN
      RTRANSID(T3)     ⟶ RDPN
      RTERMID(T4)      ⟶ RPRN
```

Normally, CICS sends the transaction code in the DPN field.

CICS uses the RETRIEVE command to receive data from openUTM. TAC is taken from the DPN field.

```
RETRIEVE ...
      RTRANSID(T1)     ⟵ RDPN
      RTERMID(T2)      ⟵ RPRN
```

9 Error diagnosis

You will find instructions on how to diagnose errors in this chapter.

9.1 Diagnostic aids

You should use the following diagnostic aids when communication problems arise between the UTM application and the partner application in an IBM system:

1. The `u62_sta` command used to check the status of openUTM-LU62. This command is described on page 59.
2. When using TRANSIT, the `SNA_status` command used to check the status of TRANSIT. This command is described in the TRANSIT-SERVER manual.
3. When using the SNAP-IX or the IBM Communications Server, the status display functions of this product.
4. The SYSLOG file of the UTM application. See the corresponding openUTM manuals for more information.
5. When using openUTM-LU62 for UNIX systems, the diagnostic files
   `/opt/lib/utmlu62/PROT/prot.`*luname*
   When using openUTM-LU62 for Windows systems, the diagnostic files
   `Programs\utmlu62\PROT\prot.`*luname*`.txt`
   The messages contained in these files are described starting on page 269.
6. When using openUTM-LU62 with TRANSIT, the diagnostic files
   `/opt/lib/transit/PROT/prot.nukleus`
   `/opt/lib/transit/PROT/prot.lu62`
   `/opt/lib/transit/PROT/cnos.mlog`
   See also the TRANSIT manuals.
7. When using SNAP-IX or the IBM Communications Server, the log display functions of this product.
8. When using LU6.1 interconnection with TRANSIT, use the diagnostics files
   `/opt/lib/transit/PROT/prot.nukleus`
   `/opt/lib/transit/PROT/prot.utmgw1`
   The messages are described in the TRANSIT manuals.

9. The JES job log with the console messages from CICS or IMS. See the corresponding CICS or IMS manuals for more information.

10. The BCAM trace from openUTM, the SNA trace from TRANSIT-SERVER, or the appropriate trace from SNAP-IX or the IBM Communications Server. The procedure to follow in order to create an SNA trace is described in the TRANSIT-SERVER manual and in the documentation of SNAP-IX and IBM Communications Server. Basic knowledge of the SNA protocol and, for LU6.2, basic knowledge of the OSI-TP protocol are required to be able to read these traces. It can occur for LU6.1 interconnection that openUTM sends a negative response or a function management header 7 (FMH7) when an error occurs. Both contain a 4-byte field containing encoded error information, also called the sense data. The sense data sent by openUTM is described starting on page 265.

11. When openUTM-LU62 is used, the protocol trace described on page 67.

⚠ Some of the diagnostics files created by openUTM-LU62 in the directory `/opt/lib/utmlu62/PROT` (UNIX systems) or `Programs\utmlu62\PROT` (Windows systems) are deleted when an entity is restarted by openUTM-LU62. You should therefore save the diagnostics data beforehand.

The following diagnostics files are deleted from the directory `utmlu62/PROT` when an entity is started by openUTM-LU62:

`in.dump.`*luname*
`xaplog.`*luname.**
`xap.dump.`*luname.**
`prot.`*luname*`.old`
`inlog.`*luname.**`.old`
`core.luname`

The files `prot.`*luname* and `inlog.`*luname.** are assigned the suffix `.old`. On Windows systems, `prot.`*luname* also has the suffix `.txt`.

9.2 LU6.1 sense data

The sense data sent by openUTM in a LU6.1 interconnection is described in the following section:

0006 0000	LUSTAT NOOP
0007 0000	LUSTAT QUEUE EMPTY
0809 xxxx	MODE INCONSISTENCY
080B 0000	BRACKET RACE ERROR
0812 0000	INSUFFICIENT RESOURCE. Resource bottleneck, for example when the pagepool is full.
0813 0000	BRACKET BID REJECT. A request to reserve a session from the contention loser application was rejected because the contention winner application has reserved the session for a job.
0814 0000	BRACKET BID REJECT, RTR FOLLOWS
0819 0000	RTR NOT REQUIRED
081B 0000	TRANSMISSION RACE. A request to reserve a session from the contention loser application was rejected because the contention winner application has reserved the session for a job.
082D 0000	LU BUSY
0835 xxxx	INVALID PARAMETER. The number xxxx is the address of the incorrect parameter in the protocol element received.
0846 0000	FMH7 FOLLOWS
084B 0000	REQUESTED RESOURCES NOT AVAILABLE, LPAP NOT ALLOWED
084D xxxx	INVALID BIND
0864 0101	FUNCTION ABORT. LUSTAT CMD has been received, CDI=0 and ERI=0.
0864 0104	FUNCTION ABORT. A dialog message was received with BB. The message cannot be processed because, for example – the TAC is invalid – the pagepool is full – the message was received with EB.
0864 0106	FUNCTION ABORT. The length of the message received is invalid or the length field of a message in the VLVB format is invalid.

0864 0107	FUNCTION ABORT. The total length of the message portion received exceeds the limit (larger than 32767).
0864 0108	FUNCTION ABORT. End of chain, but no message end (or message section end. This means that the length field received in the VLVB format contains a length value that is too large.
0864 C5D9	FUNCTION ABORT. PEND ER from the UTM application program
0864 C5E2	FUNCTION ABORT. PEND ER by openUTM
0864 D9E2	FUNCTION ABORT. PEND RS by openUTM. For example: The local service was terminated and the last input message contains CD or DR2.
0864 D9E4	FUNCTION ABORT. PEND RS from the UTM application program
0866 D9E2	FUNCTION ABORT REC RESPONSE. PEND RS by openUTM
0866 D9E4	FUNCTION ABORT REC RESPONSE. PEND RS from the UTM application program
0867 0000	SYNC EVENT RESPONSE
1002 0000	INVALID LENGTH
1003 0000	INVALID TAC. Invalid or locked transaction code.
1008 0106	INVALID FM HEADER. A message with an FMH was received, but the message does not contain the complete FMH.
2001 0000	SEQENCE NUMBER ERROR
2002 0000	CHAINING ERROR
2003 0000	BRACKET ERROR
2004 0000	DIRECTION ERROR
2005 0000	DATA TRAFFIC RESET
2007 0000	DATA TRAFFIC NOT RESET
2008 0000	NO BEGIN BRACKET
2009 0000	SESSION CONTROL PROTOCOL VIOLATION
200A 0000	IMMEDIATE REQUEST MODE ERROR
200B 0000	QUEUED RESPONSE ERROR
200C 0000	ERP SYNC EVENT ERROR
200D 0000	RESPONSE OWED BEFORE SENDING REQUEST
4001 0000	INVALID SC OR NC RH

4003 0000	BB NOT ALLOWED
4004 0000	CEB OR EB NOT ALLOWED
4006 0000	EXCEPTION RESPONSE NOT ALLOWED
4007 0000	DEFINITE RESPONSE NOT ALLOWED
4009 0000	CD NOT ALLOWED
400A 0000	NO-RESPONSE NOT ALLOWED
400B 0000	CHAINING NOT SUPPORTED
400C 0000	BRACKETS NOT SUPPORTED
400D 0000	CD NOT SUPPORTED
400F 0000	INCORRECT USE OF FORMAT INDICATOR
4010 0000	ALTERNATE CODE NOT SUPPORTED
4011 0000	RU CATEGORY NOT CORRECT
4012 0000	REQUEST COED NOT CORRECT
4013 0000	SDI/RTI NOT CORRECT
4014 0000	DR1I/DR2I/ERI NOT CORRECT
4015 0000	QRI NOT CORRECT
4016 0000	EDI NOT CORRECT
4017 0000	PDI NOT CORRECT

10 openUTM-LU62 messages

This chapter contains all messages of the openUTM-LU62 product, their meanings and any measures to be taken when the message appears.

Messages from the *u62_tp* program are output to the file

`/opt/lib/utmlu62/PROT/prot.`***luname***

on UNIX systems, and to the file

`Programs\utmlu62\PROT\prot.`***luname***`.txt`

on Windows systems. The `u62_gen`, `u62_adm` and `u62_sta` utilities output their messages to `stderr`.

On Windows systems, additional messages are output to the files

```
Programs\utmlu62\PROT\stdout.txt
Programs\utmlu62\PROT\stderr.txt
```

if openUTM-LU62 is automatically started as a service at each system startup.

All messages can be output in the German or English language. On UNIX systems, the `$LANG` environment variable controls which language is used. If the value of `$LANG` begins with the letters 'De', then the messages are output in the German language. The messages are output in English in all other cases. On Windows systems, messages are output in German if the system country settings are set to German. Otherwise, they are output in English.

The message text appears in the English language in this manual. Names in capital letters that begin with '&' (e.g. &FILENAME) serve as placeholders for variable elements in the message text.

10.1 Messages from the u62_tp program

All messages of the `u62_tp` program are described in the following section.

Messages from the `u62_tp` program are output to the file

`/opt/lib/utmlu62/PROT/prot.`*luname*

on UNIX systems, and to the file

`Programs\utmlu62\PROT\prot.`*luname*`.txt`

on Windows systems, where *luname* is the name of the corresponding LU. All messages begin with the string

u62_tp [komp] nnn

The word *komp* designates the components of the `u62_tp` program and *nnn* the message number.

There are the following components:

BD	Boundary (handles the APPC and XAP-TP interfaces as well as the interface to the system)
DM	Dialogue Manager (handles the OSI-TP dialogs and LU6.2 conversations)
RS	Resync Service Transaction Program (restarts in the start phase and after a line failure)
TM	Transaction Manager (handles the transaction management)

Messages from `u62_start` can therefore be found in the `prot.`*luname* file. These messages are assigned the prefix `u62_start` and are described starting on page 346.

BD component messages

002 Call syntax:
```
u62_tp -l <LU name> [-c|-k] [-t on[,<trace options>]]
-l LU name: name of the local LU of the openUTM-LU62 inst.
c:      cold start of openUTM-LU62
-k:     lukewarm start of openUTM-LU62
-t on:  explicit specification of the trace by
        additional options possible (separated by comma):
        IN[=<level>]:  with instance trace
        XAP:           with XAP-TP provider trace
```

The `u62_tp` program is called internally by `u62_start`. This message appears when incorrect parameters have been passed by `u62_start`.

003 No admittance to the base directory &DIRNAME,
errno &ERRNO (&ERRTEXT)

The `u62_tp` program cannot change to the *&DIRNAME* working directory. The exact cause of this is specified in the contents of the system variable `errno` = *&ERRNO*. *&ERRTEXT* contains a short text that describes the error more precisely.

004 Error on signal handling,
errno &ERRNO (&ERRTEXT)

An internal error has occurred during the initialization of the signal handling. The contents of the system variable `errno` = *&ERRNO* and the short text in *&ERRTEXT* contain more precise information on the cause of the error.

006 Error opening the configuration file &FILENAME
errno &ERRNO (&ERRTEXT)

007 Error reading the configuration file &FILENAME,
errno &ERRNO (&ERRTEXT)

008 Wrong version in configuration file &FILENAME (&VERS1 instead of &VERS2)

The configuration file *&FILENAME* created by the `u62_gen` generation program contains an incorrect version number. *&FILENAME* is either not a configuration file or an incompatible `u62_gen` version was used to generate it.

009 The local LU name &LUNAME is not configured.

The local LU *&LUNAME* for which the entity was started (-l switch) is not contained in the configuration file.

010 The instance for the local LU name &LUNAME is already running.

The entity for the local LU *&LUNAME* has already been started. Only one entity may be run for a local LU.

011 Shutdown due to lack of resources

The entity has terminated itself because an attempt to obtain temporary memory has failed. This message only appears during the initialization phase.

012 Internal error occurred

An internal error has occurred during initialization. The entity has terminated itself because of this. You should therefore save diagnostics documents.

013 Version: &VERSION &DATE &VERSIONID
System: &SYSTEM (&SYSTEMVERSION)
Host name: &SYSTEMNAME

At startup, the instance outputs information on the program version of `u62_tp` and on the system environment. This information is particularly important for the purpose of diagnostics. The contents of the variables are as follows:
&VERSION contains the version number of the program `u62_tp`.
&DATE is the creation date of the program `u62_tp`.
&VERSIONID contains an internal ID of the program `u62_tp`.
&SYSTEM is the name of the operating system (e.g. SunOS).
&SYSTEMVERSION is the operating system version.
&SYSTEMNAME is the host name of the local host.

015 PID file &PIDFILE cannot be created,
errno &ERRNO (&ERRTEXT)

The entity cannot create the system files (*&PIDFILE*) it uses internally. The system variable `errno` = *&ERRNO* contains information regarding the cause.

016 Write to PID file &PIDFILE failed,
errno &ERRNO (&ERRTEXT)

Information cannot be stored in the file *&PIDFILE* used internally. The file system is probably full.

017 Write to PID file &PIDFILE failed,
written &NUM1 bytes, expected: &NUM2 bytes

An attempt to write *&NUM2* bytes in the file *&PIDFILE* has failed. Probably on account of a full file system, only *&NUM1* bytes of the *&NUM2* bytes could be stored.

018 Error creating the input pipe &PIPEFILE,
errno &ERRNO (&ERRTEXT)

An attempt to create the named pipe *&PIPEFILE* has failed. The errno variable contains the cause of the error. This message can only appear immediately after the start because the input pipe is only created once, during initialization. The entity terminates itself in this case because communication with the other openUTM-LU62 components is not possible.

019 Error opening the input pipe &PIPEFILE,
errno &ERRNO (&ERRTEXT)

The named pipe *&PIPEFILE* just created cannot be opened for reading. Otherwise, the same is true here as described above for message 018.

020 Only &NUM1 of the desired &NUM2 XAP-TP instances could be created.

An attempt was made during the initialization phase to create an XAP-TP entity for every parallel connection desired (ASSOCIATIONS configuration parameter) and for the XAP-TP control entity (for transaction monitoring). If the XAP-TP provider only allows *&NUM1* XAP-TP entities instead of the desired *&NUM2* XAP-TP entities, then this message is output.

There seems to be a resource bottleneck. The system administrator should deal with it.

021 New log name created for openUTM-LU62:
&LOGNAME

If the value UDTCCR or UDTSEC was generated as the application context, then sync-level 2 must be supported in LU6.2 page conversations. A prerequisite for this is the exchange of log names with the LU6.2 partner. The log name *&LOGNAME* just created is output with this message by openUTM-LU62 to inform the system administrator during a cold start of the entity.

022 Log Name of openUTM-LU62 restored from synclog file:
&LOGNAME

No new log name is generated during a warm start of openUTM-LU62, rather the old log name from and earlier cold start is used. This message only appears during initialization and only serves to inform the system administrator.

023 Log Name of the partner restored from synclog file:
&LOGNAME

During an openUTM-LU62 warm start, its own log name as well as the log name *&LOGNAME* of the LU6.2 partner are taken from the synclog file if the log name of the LU6.2 partner became known during a previous run of the entity when the log names were exchanged. This message only appears during initialization and only serves to inform the system administrator.

025 openUTM-LU62 terminated.

This message is the last message written in the log file when an entity is terminated normally by openUTM-LU62 by the administration (`u62_adm -e`).

026 Crash of openUTM-LU62 in module &MODULNAME,
error number = ERRNUM, error code = ERRCODE

The openUTM-LU62 entity has terminated itself due to a fatal internal error. The *&MODULNAME*, *&ERRNUM* and *&ERRCODE* specifications are internal error information and are very important for the error analysis. Save the contents of the directory `/opt/lib/utmlu62/PROT` (UNIX systems) or `Programs\utmlu62\PROT` (Windows systems). If possible, you should reproduce the error with the entity trace activated.

027 Shutdown by the XAP-TP provider:
&REASON

The XAP-TP provider has detected a fatal error and initiates the termination of the openUTM-LU62 entity. *&REASON* contains more detailed information as to the case of the error.

The **Grp.** (group) column in the following table describes which group of causes the cancellation belongs to. There are the following groups:

A The cause is an application error, e.g. an error while
- generating and administrating openUTM-LU62
- generating the system (e.g. allocating the address space)

S The cause is an error in another system component (software or hardware).

M The cause is a shortage of memory.

X The cause is an internal error in openUTM-LU62.

It is possible that several causes are listed, e.g. XAS.

You should consult Siemens system services for all errors in groups S and X and for all error codes **not** listed in the following table. Save the contents of the directory `/opt/lib/utmlu62/PROT` (UNIX systems) or `Programs\utmlu62\PROT` (Windows systems). If possible, you should reproduce the error with the entity trace activated.

REASON	**Grp.**	**Cause**
AHQA00	XM	KCOCOTA module, function QueueAnno(). The mGetBufferSpace() macro returned the return code LB_NOREM.
ASA001	XA	The KCOASAM module was called with an invalid operation code
ASA002	XA	The KCOASAM module was called with the operation code ASAM_SET_ACCPTS. The number of access points specified is larger than the number of generated access points.
ASA003	XA	The KCOASAM module was called with the operation code ASAM_SET_ACCPTS. Bad return code from BerSetAet.
ASA004	XA	The KCOASAM module was called with the operation code ASAM_SET_ PARTNER. Bad return code from BerSetAet.
ASA006	ASX	The KCOASAM module was called with the operation code ASAM_ATTACH. Bad return code from bBuildPAddr.
ASA007	ASX	The KCOASAM module was called with the operation code ASAM_ATTACH. OSS returns the return code NOTFIRST. The entity has probably already been started.
ASA104	AX	The “bBuildPAddr“ function of the KCOASAM module was called. The presentation selector of a local access point or of a remote partner is too long.

REASON	Grp.	Cause
ASA105	AX	The “bBuildPAddr“ function of the KCOASAM module was called. The session selector of a local access point or of a remote partner is too long.
ASA106	AX	The “AssociationRequest“ function of the KCOASAM module was called. OSS returns the return code ERROR from an “assrq“ call.
ASA107	AX	The “AssociationRequest“ function of the KCOASAM module was called. OSS returns the return code INVREF from an “assrq“ call.
ASA108	AX	The “AssociationRequest“ function of the KCOASAM module was called. OSS returns bad return code from an “assrq“ call.
ASA123	A	The KCOASAM module was called with the operation code ASAM_SET_TRANSSYNS. The number of transfer syntaxes to be initiated is larger than the number generated.
ASA124	A	The KCOASAM module was called with the operation code ASAM_SET_TRANSSYNS. The number of elements in an object identifier that designates a transfer syntax is larger than allowed.
ASA125	A	The KCOASAM module was called with the operation code ASAM_ATTACH. The reference for the transfer syntax contains an invalid value.
CDTN02	M	KCOCOHF module, function CheckDtnidTtnid(). The mGetBufferSpace() macro returned the return code LB_NOREM.
CSND04	XM	Invalid return code after calling PutElement() to request a dynamic buffer for data transmitted by the concatenator.
DMCA00	M	KCOCODM module, function ConnectDynMemArea(). The function ConnectSharedMem() returns the return code MEM_NOMEM.
DMDI00	XA	KCOCODM module, function DynDynMemDmpInfo(). The parameter <pnAreaEntries> points to a value smaller than or equal to null.
EHHP00	M	KCOXFEH module, function HandlePresEvent(). The return code from mGetBufferSpace() was not equal to LB_OK.
EHRP01	MX	KCOXFEH module, function ReloadPresEvent(). The mGetBufferSpace() macro returned a return code not equal to LB_OK.
EHSP01	M	KCOXFEH module, function StorePresEvent(). The function PutElement() returned the return code DM_NOMEM. Measure: Increase the value for the size of the OSI scratch area in the KDCDEF generation (MAX OSI-SCRATCH-AREA parameter).
EVGE00	M	KCOXFEV module, function GetOssEvent(). The return code of the mGetBufferSpace() macro was not equal to LB_OK.

REASON	Grp.	Cause
FREE01 through FREE03	XA	KCOXFFO module, function ap_free(). There are more than APFREE_MAX_TO_REL memory areas to be released.
GSYS00	S	KCOCOHF module, function GetSystemInfo(). The UNIX C function uname() returned a negative return code.
LBBD00	AX	KCOCOLB module, function BufferDumpInfo(). The parameter <nPartEntries> contained a value smaller than or equal to null.
MACF02	M	The return code from mGetBufferSpace() was not equal to LB_OK.
MACF03	M	The return code from SetTimer() was not equal to TI_OK.
MACF04	M	The return code from GetLogRecord was not equal to MACF_OK.
MFCR07 through MFCR12 MFCR16 through MFCR21 MFCR24	M	The return code from the mGetBufferSpace() macro was not equal to LB_OK.
MFDM03 through MFDM06	M	The return code from PutElement was not equal to DM_OK.
MFRM05 through MFRM07	A	There were no free dialog table entries available for a transaction branch in TP_RECOVER_REQ. Possible cause: The number of associations in the previous application run was larger than the number of associations in the current application run.
MFRM08	M	The return code from the mGetBufferSpace() macro was not equal to LB_OK.
MFRM09 through MFRM19 MFRM21	M	The return code from PutElement was not equal to DM_OK.
MFRM25	A	No table entries free for a log damage record. Measure: Delete the log damage records with TP_UPDATE_LOG_DAMAGE_REQ or set nMaxLogDamRec to a larger value.
MFT102	M	The return code from the mGetBufferSpace() macro was not equal to LB_OK.
MFT104	M	The return code from ChangeDescriptor was not equal to DM_OK.
MFT105	M	The return code from PutElement was not equal to DM_OK.
MFT106	M	The return code from CopyElement was not equal to DM_OK.

REASON	Grp.	Cause
MFT107	M	The return code from CopyElement was not equal to DM_OK.
MFT108	M	The return code from the mGetBufferSpace() macro was not equal to LB_OK.
MFT109	M	The return code from CopyElement was not equal to DM_OK.
MFT110 MFT111	M	The return code from PutElement was not equal to DM_OK.
MFT112	M	Invalid internal message number (LOGREMOVE).
MFT113	M	The return code from ChangeDescriptor was not equal to DM_OK.
MFT114	M	The return code from CopyElement was not equal to DM_OK.
MFT115	M	The return code from ChangeDescriptor was not equal to DM_OK.
MFT119	M	The return code from GetLogRecord() was not equal to MACF_OK.
MFT120 through MFT127	M	The return code from PutElement was not equal to DM_OK.
MFT128 MFT129	M	The return code from CopyElement was not equal to DM_OK.
MFT130 MFT131 MFT132	M	The return code from PutElement was not equal to DM_OK.
MFT133	M	The return code from CopyElement was not equal to DM_OK.
MFT134 MFT135	M	The return code from PutElement was not equal to DM_OK.
MFT138 MFT139	M	The function PutElement has returned a return code not equal to DM_OK.
MFT141	M	The function PutElement has returned a return code not equal to DM_OK.
MFT142	M	The function CopyElement has returned a return code not equal to DM_OK.
MFT147 MFT151	M	The function PutElement has returned a return code not equal to DM_OK.
MFTP03	M	The return code from PutElement was not equal to DM_OK.
MFTP04	M	The return code from the mGetBufferSpace() macro was not equal to LB_OK.
MFTP05 MFTP06	M	The return code from PutElement was not equal to DM_OK.
MFTP07	M	The return code from SetTimer was not equal to TI_OK.

REASON	**Grp.**	**Cause**
MFTP10	M	The return code from the mGetBufferSpace() macro was not equal to LB_OK.
MFTP11	M	The return code from RequestBuffer() was not equal to LB_OK.
MFTP12 through MFTP19 MFTP24	M	The return code from the mGetBufferSpace() macro was not equal to LB_OK.
NMTE00	M	KCOCOHF module, function NewMemTabEntry(). The function RequestBuffer() returned the return code LB_NOMEM.
NMTE02	M	KCOCOHF module, function NewMemTabEntry(). The mGetBufferSpace() macro returned the return code LB_NOMEM.
PCTR00	M	KCOCOHF module, function PrepareCtrlReq(). The mGetBufferSpace() macro returned the return code LB_NOMEM.
POLL03	XM	KCOXFPL module, function ap_poll(). The return code of the mGetBufferSpace() macro was not equal to LB_OK.
RCV009	XM	KCOXFRV module, function ap_rcv(). The return code the function CopyElement() was not equal to DM_OK.
RCV012	XM	KCOXFRV module, function ap_rcv(). Inconsistency in the Boolean variables <bSwitchToNextTtnid> and <bClearTtnid>.
RQOB00	M	KCOCOHF module, function ReqOssInBuff(). The function RequestBuffer() returned the return code LB_NOMEM.
RVCA01	XA	KCOXFRV module, function CopyVarLthAttr(). The function AllocUserMem() returned an unexpected return code.
RVCS03	M	KCOXFRV module, function CheckSaRetc(). The return code returned by the function SetAttribute() was SA_NOMEM.
RVUO01	XS	KCOXFRV module, function UserDataOut(). The buffers provided by UTM in the multi-tasking variant to input the user data are not large enough to transfer all user data.
SACT14	XM	Invalid return code after calling PutElement() to request a dynamic memory area for the COPY action in SACF.
SDCS02	M	KCOXFSD module, function CheckSaRetc(). The SetAttribute() return code was SA_NOMEM.
SDUI01	M	KCOXFSD module, function UserDataIn(). The mGetBufferSpace() macro returned the return code LB_NOMEM.
SND007	XM	KCOXFSD module, function ap_snd(). The return code the function GetVarLthAttr() was not equal to GA_OK.

REASON	Grp.	Cause
XADM12	AX	KCOXFEX module, function apext_adm(). The function apext_adm() was called with a unknown opcode in the multitasking variant.
XATT04	XM	KCOXFEX module, function apext_att(). The return code of the function EstablishBuffer() was not equal to LB_OK.
XATT12	XM	KCOXFEX module, function apext_att(). The return code of the function EstablishBuffer() was not equal to LB_OK.
XATT13	XM	KCOXFEX module, function apext_att(). The return code of the function RequestBuffer() was not equal to LB_OK.
XFAU03	XA	KCOXFHF module, function AllocUserMem(). The parameter <Type> contained an invalid value.
XFDU02	XA	KCOXFHF module, function DeallocUserMem(). The parameter <Type> contained an invalid value.
XFGA07	XSA	KCOXFHF module, function GetAttribute(). The function AllocUserMem() returned an unexpected return code when reading the AP_DTNID attribute in the single-tasking variant.
XFGA11	M	KCOXFHF module, function GetAttribute(). The mGetBufferSpace() macro returned the return code LB_NOMEM.
XFGE02	AX	KCOXFHF module, function bCheckAndGetCallEnv(). The function bCheckAndSetState() returned a bad return code and the application status was not equal to WAITING_DUMP_APPL.
XFSA07	MX	KCOXFHF module, function SetAttribute(). The function PutElement() returned a return code not equal to DM_OK when setting the AP_DTNID attribute.

028 Shutdown by internal watch dog.

The `u62_tp` program constantly monitors to ensure that no function call takes longer than a defined period of time. In this manner, infinite loops and blocked system functions are recognized automatically. This error can occur once in while when terminating TRANSIT. If it occurs in a different situation, then a system error has occurred.

030 Cold start of openUTM-LU62, because the synclog file &SYNCLOG is missing.

openUTM-LU62 was called without the -c switch and is therefore to execute a warm start. If, however, openUTM-LU62 has not yet started successfully for the local LU specified, then the *&SYNCLOG* file with the information for a warm start is missing. In this case, message 030 means that a cold start is to be executed instead. This message only serves to inform the system administrator.

031 Error opening the synclog file &SYNCLOG,
errno &ERRNO (&ERRTEXT).
Since the CCR syntax is not configured, the initialization is continued.

This message is output during the initialization if openUTM-LU62 is started without the -c switch and the application context UDTAC or UDTDISAC was generated, but a *&SYNCLOG* synclog file from a previous start still exists. This can only happen if the configuration has been changed between two openUTM-LU62 starts, which changed the application context of UDTCCR or UDTSEC on UDTAC or UDTDISAC. There is no transaction management when UDTAC or UDTDISAC is used, and that is why openUTM-LU62 does not create a synclog file. The existing *&SYNCLOG* file is not deleted as a precaution. If this file is not actually needed any more, then it must be deleted manually by the system administrator.

032 Error opening the synclog file &SYNCLOG,
errno &ERRNO (&ERRTEXT).
Terminate openUTM-LU62, because CCR syntax is configured.

An error occurred during a warm start while opening the synclog file *&SYNCLOG* containing all information required for a successful restart. The error is explained in more detail by the contents of the variable `errno` = *&ERRNO*. openUTM-LU62 breaks off the initialization in this situation because this file may contain important information on cancelled transactions. If it can be guaranteed that no transactions need to be restarted, then a cold start can be initiated with the -c switch.

033 The synclog file &SYNCLOG remains unchanged, since the CCR syntax is not configured (any more).

The system has determined during a warm start that a synclog file *&SYNCLOG* still exists from a previous openUTM-LU62 start. openUTM-LU62 leaves the synclog file unchanged as a precaution and continues initialization because of the fact that the application context of UDTCCR or UDTSEC was changed in the meantime after UDTAC or UDTDISAC (APPL-CONTEXT parameter), meaning that transaction management is not required any more.

034 Wrong format of the synclog file &SYNCLOG, cold start required!

openUTM-LU62 has determined during a warm start that the version of the synclog file does not match the version of the `u62_tp` program (only possible after a new openUTM-LU62 version has been installed) or that the *&SYNCLOG* synclog file does not exist or is damaged. There are only two reasonable explanations:

Either the file was not created by openUTM-LU62 but by a system user, or the file could not be written completely to the hard drive due to a lack of file system space and has therefore been truncated. openUTM-LU62 therefore aborts the initialization. openUTM-LU62 must be cold started after this occurs because a warm start is not possible with the existing synclog file.

035 The entries in the synclog file &SYNCLOG are not consistent with the given configuration,
cold start required!

openUTM-LU62 has determined during a warm start that certain entries in the synclog file do not correspond to the current configuration. More specifically, the APT and AEQ of the local and the remote OSI-TP application are the problem. A successful openUTM-LU62 start is now only possible using the -c switch.

036 Error in allocating a new shared memory of length &LEN,
errno &ERRNO (&ERRTEXT)

If the application context allows transaction management, then `u62_tp` program allocates a shared memory area during the initialization phase with the system function shmget(). The shared memory area represents an image of the synclog file and reads the `u62_wlog` daemon process before it writes the information onto the hard drive. An error while creating this shared memory area leads to openUTM-LU62 aborting. The system variable `errno` informs you of the exact cause.

037 Error in attaching to the shared memory with the id &ID,
errno &ERRNO (&ERRTEXT)

After creating a shared memory area, the area is assigned an address using the system function shmat() to allow read and write access. If this fails, then openUTM-LU62 aborts the initialization.

040 Error in the configuration of the local LU &LLUNAME
or of the partner LU &RLUNAME.
The communication with the LU6.2 and with the OSITP partner will be started only after the configuration of the LU6.2 base software will have been corrected.

If the local LU *&LLUNAME* (configuration parameter LOC-LU-ALIAS) or the remote LU *&RLUNAME* (configuration parameter REM-LU-ALIAS) contained in the openUTM-LU62 generation is not configured in the LU6.2 basic software, then openUTM-LU62 waits to start the communication with the OSITP and LU6.2 partners.

When TRANSIT is used as the LU6.2 basic software, then the cause may also be the lack of a suitable PAIR parameter, which creates the association between the local and the remote LU. If TRANSIT is terminated, reconfigured and restarted, then everything should run again.

041 The error in the configuration of the local LU &LLUNAME
and of the partner LU &RLUNAME has been corrected.
The communication with the LU6.2 and with the OSITP partner is started now.

openUTM-LU62 has detected an inconsistency during the start in the LU6.2 basic software generation (e.g. of TRANSIT) and has output message 040. After the configuration of the LU6.2 basic software has been corrected, openUTM-LU62 outputs message 041 and starts communication with its two partners by initiating a RECEIVE_ALLOCATE call on the LU6.2 side and by opening the XAPTP dialog entities on the OSITP side.

042 The net name of the partner LU &ALIASNAME has been changed from &OLDNAME to &NEWNAME.
Allocation of conversations is not possible any more!

openUTM-LU62 detected the network name of the partner LU *&ALIASNAME* (in the TRANSIT configuration parameter REM-LU-ALIAS) configured in the LU6.2 basic software (e.g. in TRANSIT) during a cold start with the help of a control operator call. Each time a new conversation is opened to the LU6.2 partner the system will check to see of this name is still correct or if it has been recently reconfigured (*&NEWNAME*). The conversation is closed immediately when an error occurs.

043 The net name of the partner LU &ALIASNAME has been corrected again.

openUTM-LU62 has determined that the network name of the partner LU in the LU6.2 basic software (e.g. in TRANSIT) was reconfigured and message 042 was output for this reason. If openUTM-LU62 recognizes that the network name has been corrected in the LU6.2 basic software when attempting to open a conversation, then this message is output for informational purposes.

044 The net name of the local LU &ALIASNAME has been changed from &OLDNAME to &NEWNAME.
Allocation of conversations is not possible any more!

This message has the same meaning as number 042 with the difference that the local LU was affected by the network name change, not the partner LU.

045 The net name of the local LU &ALIASNAME has been corrected again.

This message has the same meaning as number 044 with the local instead of the remote LU.

046 Allocation of new conversations is possible again.

This message appears together with message 043 and/or 045 and shows that conversations to the LU6.2 partner can be opened again after a faulty network name change has been corrected.

047 The node is activated: Conversations can be established now.

The IBM Communications Server node has been started successfully. Conversations with the LU6.2 partner can now be opened (or resumed).

048 The node is deactivated: No conversations can be established.

The IBM Communications Server node was presumably deactivated by the administrator. It is no longer possible to open conversations with the LU6.2 partner.

049 The Attach Manager is already running:
Incoming conversations are not routed to openUTM-LU62.

An attach manager has already been started for the local LU of the openUTM-LU62 entity - possibly an openUTM-LU62 entity in another environment (U62_DIR variable). In this case, all requests to open a conversation received by the LU6.2 partner on the IBM Communications Server are forwarded to another application instead of to the openUTM-LU62 entity. In other words, only one active connection may be established with the LU6.2 partner.

050 Read on input pipe failed with error ENOBUFS.

The error `errno` = ENOBUFS has occurred while reading from a named pipe with read(), i.e. there is currently a shortage of system memory at the moment. `u62_tp` then repeats the attempt to read after a wait time of 1/4 second has passed; if the error ENOBUFS occurs again, then an error message is not output. If this error occurs more than 20 times in a row or more than 60 times during the entire `u62_tp run time`, then the program terminates itself with an abnormal termination.

051 Error opening the pipe to the administration program with PID = &ADMPID, errno &ERRNO (&ERRTEXT)

All jobs sent by the `u62_adm` and `u62_sta` administration programs to `u62_tp` are acknowledged by `u62_tp` with an acknowledgment (also `u62_adm -e`). If the opening of the named pipe used to send the acknowledgment using the system function open() fails, then `u62_tp` outputs this message. The contents of the variable `errno` = *&ERRNO* specifies the exact cause for this.

It can happen during long jobs, such as the status display, and when the load is high that the administration program has deleted its input pipe when `u62_tp` wants to send its acknowledge. This is because the administration programs only wait for a maximum of 5 seconds for the reply from `u62_tp,` and terminate themselves after this wait time has passed.

052 Write into pipe to process with PID = &PID failed with error ENOBUFS.

The error `errno` = ENOBUFS has occurred while an administration program or a job attempted to write an acknowledge to the write log daemon `u62_wlog` with the system function write(). `u62_tp` repeats the write attempt after a wait time of 1/4 second, but does not output this message any more when this error reoccurs.

If this error occurs 10 times in a row, then `u62_tp` simply throws out the internal message. The administration program would then announce that the time limit has been exceeded while waiting for acknowledgment. If the error occurs often enough so that is has occurred more than 60 times in this run, then `u62_tp` terminates itself with an abnormal termination.

053 The output pipe to the process with the PID = &PID is full.

The system cannot write any data at all at the moment through the named pipe to the administration program or to the write log daemon with PID = *&PID*. This message should not appear because the internal messages from openUTM-LU62 are relatively short and are also read immediately from their input pipes by the individual programs. `u62_tp` repeats the write operation after 1/4 second in a manner similar to that for error ENOBUFS and rejects the message after 20 unsuccessful attempts.

054 Error writing into the pipe to the process with PID = &PID,
errno &ERRNO (&ERRTEXT)

The error `errno` = *&ERRNO* has occurred while writing through the output pipe to the administration program or to the write log daemon with the PID = *&PID*. `u62_tp` does not repeat the write attempt, in contrast to the two errors above (ENOBUFS, EAGAIN) because this error must be a fatal error.

055 Response to the administration program discarded!

An error that has not yet lead to `u62_tp aborting` has occurred while opening a named pipe to an administration program or when writing to this pipe. This means that this message always precedes messages 051 - 054.

056 Unknown internal message 0x&HEX received via input pipe!

`u62_tp` has received an internal message with the unknown message type *&HEX* (4 digits, hexadecimal). These messages point either to a communication error between `u62_tp` and the administration programs or to a faulty openUTM-LU62 installation (incompatible versions of `u62_tp` and `u62_adm` or `u62_sta`).

057 Unknown internal message 0x&HEX received from an administration program

See comment on message 056.

058 SNMP not supported in this version; message 0x&HEX rejected!

See comment on message 056.

060 Error opening the pipe to the write log daemon with PID = &PID, errno &ERRNO (&ERRTEXT)

The write log daemon `u62_wlog` will be started by `u62_tp` during initialization if UDTCCR or UDTSEC is configured as the application context, meaning that restart information must therefore be written to the hard drive. As soon as `u62_wlog` has sent its “Ready” message to `u62_tp`, `u62_tp` opens its output pipe to this daemon for later jobs. If the open() system call fails, then this message is output with the reason for the error in the `errno variable`.

061 Termination of the write log daemon with PID &PID during start.

The write log daemon `u62_wlog` started by `u62_tp` could not execute its initialization successfully. In this case, it sends a corresponding internal message to `u62_tp` that leads to the output of message 061, and the daemon terminates itself. `u62_tp` then attempts to restart `u62_wlog` after a specific wait time has passed.

062 Write log daemon with PID &PID reports error in writing to the synclog file => instance crash!

The opening of or writing to the synclog file by the write log daemon has failed. This error leads to the immediate abort of `u62_tp`. This error should only occur if there is not enough space for the synclog file on the drive due to a full file system. This error can only occur during the start phase because the size of the synclog file is not changed while openUTM-LU62 is running.

063 Unknown message 0x&HEX received from the write log daemon!

`u62_tp` has received an internal message with the unknown message type *&HEX* (4 digits, hexadecimal) from the write log daemon. These messages point either to a communication error between `u62_tp` and the write log daemon or to a faulty openUTM-LU62 installation (incompatible versions of `u62_tp` and `u62_wlog`).

064 The write log daemon with PID &PID has crashed down!

The `u62_tp` program checks in regular intervals whether or not the write log daemon with PID = *&PID* that it has started is still alive. If the daemon no longer exists, then message 064 is output and `u62_wlog` is restarted.

065 Write log daemon with PID &PID has been started.

This message does not mean that an error has occurred. It only logs the PID of the child process just started using fork(), which still needs to be overlayed with the program `u62_wlog` using exec().

066 The maximum number &NUM of unsuccessful attempts to start the write log daemon has been exceeded.

`u62_tp` has unsuccessfully attempted to start the write log daemon `u62_wlog` several times in a row in constantly increasing intervals. If the maximum number *&NUM* of such attempts is reached, then `u62_tp` terminates itself.

067 Error starting the write log daemon by calling fork(), errno &ERRNO (&ERRTEXT)

The fork() system call used to start the write log daemon in the start phase or after `u62_tp` has determined that `u62_wlog` is not running any more has failed. This is usually the result of a lack of system resources in the system. The contents of the variable `errno` = *&ERRNO* contains more exact information as to the cause. `u62_tp` will attempt to start the daemon again after a specific wait time.

068 Error starting the write log daemon by calling execv(), errno &ERRNO (&ERRTEXT)

This message has the same meaning as message 067 with the difference that the execv() function, not the fork() function, to overlay the process with the `u62_wlog` program has failed. This error message means that there is either a resource shortage in the system or an incorrect environment (`u62_wlog` is not available or is not executable).

069 Internal message could not be sent to the write log daemon with PID &PID! Terminate and restart the daemon!

An error has occurred while writing an internal message through the named pipe to the write log daemon with write() that did not lead to `an abort of u62_tp`. This message always precedes messages 052 - 054. `u62_tp` terminates the write log daemon, releases all associated resources and restarts the write log daemon.

070 ap_set_env failed for instance &INSTNO, ap_errno = &APERRNO: &ERRTXT

This message indicates an internal error that occurred while using the XAP-TP interface for communication via OSI-TP. The parameters have the following meanings

&INSTNO:	Number of the XAP-TP entity
&APERRNO:	XAP-TP error number
&ERRTXT:	XAP-TP error text

071 ap_get_env failed for instance &INSTNO, ap_errno = &APERRNO: &ERRTXT???

See comment on message 070.

072 ap_free failed for instance &INSTNO, ap_errno = &APERRNO: &ERRTXT

See comment on message 070.

073 ap_bind failed for instance &INSTNO,
ap_errno = &APERRNO: &ERRTXT???

See comment on message 070.

074 ap_close failed for instance &INSTNO,
ap_errno = &APERRNO: &ERRTXT???

See comment on message 070.

075 ap_init_env failed for instance &INSTNO,
ap_errno = &APERRNO: &ERRTXT???

See comment on message 070.

076 ap_open failed,
ap_errno = &APERRNO: &ERRTXT

See comment on message 070.

077 ap_rcv failed for instance &INSTNO,
ap_errno = &APERRNO: &ERRTXT

See comment on message 070.

078 ap_snd failed for instance &INSTNO,
ap_errno = &APERRNO: &ERRTXT

See comment on message 070.

079 ap_poll failed,
ap_errno = &APERRNO: &ERRTXT

See comment on message 070.

080 apext_adm (operation code &OPCODE) failed,
ap_errno = &APERRNO: &ERRTXT

See comment on message 070. &OPCODE represents the code for the administration call.

081 apext_att failed,
ap_errno = &APERRNO: &ERRTXT

See comment on message 070.

082 apext_det failed,
ap_errno = &APERRNO: &ERRTXT

See comment on message 070.

083 apext_init for Type &INITTYPE failed,
ap_errno = &APERRNO: &ERRTXT

See comment on message 070. &INITTYPE represents the type of data to be initialized.

090 The dump file &FILENAME cannot be opened.

`u62_tp` has unsuccessfully attempted to output a dump of its control block to the file *&FILENAME* because of a fatal internal error or because of an administration command. This file could not be opened, however.

100 Message &MSGNUM of the XAP-TP provider:

This message initiates a system message of the XAP-TP provider. The message text depends on the message number *&MSGNUM*. The individual messages of the XAP-TP provider are listed in their own section starting on page 319.

DM component messages

301 Conversation rejected,
TP Name &TPNAME, job submitted by LU6.2 side:
CCR syntax not negotiated for the OSITP partner application

The LU6.2 partner has opened a conversation with sync-level 2 to the TP *&TPNAME*. (*&TPNAME* therefore designates the UTM transaction code.) openUTM-LU62 cannot establish an OSI-TP dialog with the functional unit commit to the OSITP partner as requested and rejects the conversation immediately because the UDTAC or UDTDISAC application context has been configured.

302 Conversation rejected,
TP Name &TPNAME, job submitted by LU6.2 side:
Restart of the nominated control instance not completed yet

The LU6.2 partner has opened a conversation with sync-level 2 to the TP *&TPNAME*. (*&TPNAME* therefore designates the UTM transaction code.) However, the XAP-TP control entity is not yet in a suitable state to support new OSI-TP dialogs with the functional unit commit because the restarting of transactions that are left over from a previous openUTM-LU62 start has not completed yet. For this reason, `u62_tp` rejects the conversation immediately. This can occur when the connection to the LU6.2 partner is re-established after a crash, but the connection between openUTM-LU62 and the UTM application has not yet been re-established.

303 Incoming conversation
(TP Name &TPNAME, job submitted by LU6.2 side)
supplies initialization data that will be discarded by openUTM-LU62.

This message informs you that an incoming conversation from the LU6.2 partner has delivered initialization data that cannot be transmitted to the OSI-TP partner (i.e. to the UTM application). Check the application program of the LU6.2 partner. If the initialization data is required for the application logic, then reasonable communication is not possible. *&TPNAME* designates the UTM transaction code.

304 Conversation rejected,
TP Name &TPNAME, job submitted by LU6.2 side:
Security data not supported by the OSITP partner or contain invalid characters.

An incoming conversation from the LU6.2 partner to the TP *&TPNAME* contains security data (user ID and possibly a password) and can only be transmitted to the OSITP partner (i.e. to UTM application) if this security check is also supported. This is only the case, however, if APPL-CONTEXT = UDTSEC is configured. openUTM-LU62 rejects the conversation immediately with the error code AP_SECURITY_PARAMS_INVALID for all other application contexts or when an error is found in the ASN1 code of the security data. *&TPNAME* designates the UTM transaction code.

305 Conversation rejected,
TP Name &TPNAME, job submitted by LU6.2 side:
The alias name &ALIAS of the partner LU is not identical with the configured name &CONFALIAS.

There is a problem with the names in the configuration.

306 Conversation rejected,
TP Name &TPNAME, job submitted by LU6.2 side:
Service TPs using mapped (!) conversations are not supported!

The LU6.2 partner has attempted to start a conversation with a non-printable TP name (i.e. a transaction code between X’00’ and X’3F’). openUTM-LU62 does not support these types of service TPs, except for the resync-TP X’06F2’. This message appears, for example, when the EXEC CICS START function that addresses the service TP X’02’ in openUTM-LU62 is used.

307 Conversation rejected,
TP Name &TPNAME, job submitted by LU6.2 side:
Configuration error: The net name of the local and/or of the partner LU has been changed!

An incoming conversation from the LU6.2 partner to the TP *&TPNAME* is rejected because an inconsistency has been detected while checking the network names of the local and the remote LU. This means that the alias names for the local and/or the remote LU (e.g. in the TRANSIT configuration) are not aliases of the same LUs (in the SNA network) as before. This message always precedes at least one of the messages 042 and 044.

308 Conversation rejected,
TP Name &TPNAME, job submitted by LU6.2 side:
Error in the exchange of the log names between the local and the remote LU!

The LU6.2 partner has opened a conversation to openUTM-LU62 with sync-level 2 although the log names have not yet been exchanged between the two LUs, or a fatal error occurred the last time the log names were exchanged. openUTM-LU62 reacts to this protocol violation by immediately closing the conversation.

309 RECEIVE_ALLOCATE rejected due to configuration error:
The local LU alias name &LUNAME is not configured!

The RECEIVE_ALLOCATE call to asynchronously accept incoming conversations is rejected by the LU6.2 basic software (e.g. by TRANSIT) because the alias name contained in this call of the local LU is not configured in the LU6.2 basic software.

310 RECEIVE_ALLOCATE failed:
LU6.2 base software is not running!

The RECEIVE_ALLOCATE call receives a negative replay because the LU6.2 basic software is not running. When TRANSIT is used as the LU6.2 basic software, then either the TRANSIT base system or the LU62Mgr has not been started. openUTM-LU62 re-initiates a RECEIVE_ALLOCATE call after a specific wait time but does not output this message again when an error occurs.

311 RECEIVE_ALLOCATE failed:
System error (errno = &ERRNO) occurred:
&ERRTXT

One of the system calls failed while the LU6.2 basic software (e.g. TRANSIT) was processing the RECEIVE_ALLOCATE call. The actual cause of the error should be returned by the LU6.2 basic software in the RECEIVE_ALLOCATE call in the `errno` variable. The value of `errno` *&ERRNO* together with a short interpretation in *&ERRTXT* is therefore output by openUTM-LU62. However, as TRANSIT does not return any text as the LU6.2 basic software, the following always appears as the error text:

The errno is not interpreted when TRANSIT is used as the LU6.2 software

312 Allocation of a conversation to the LU6.2 partner rejected,
TP Name &TPNAME, job submitted by OSI-TP side:
Configuration error:
The local LU alias name &ALIASNAME is not configured!

The LU6.2 basic software (e.g. TRANSIT) has rejected the attempt to open a conversation to the LU6.2 partner because the alias name *&ALIASNAME* of the local LU generated in openUTM-LU62 is not configured in the LU6.2 basic software.

313 Allocation of a conversation to the LU6.2 partner failed,
TP Name &TPNAME, job submitted by OSI-TP side:
The LU6.2 base software is not running!

The LU6.2 basic software is not running. when TRANSIT is being used. This means that the TRANSIT base system or the LU62Mgr has not been started. It is therefore not possible to communicate via LU6.2. *&TPNAME* designates the CICS transaction code, for example.

314 Allocation of a conversation to the LU6.2 partner failed,
TP Name &TPNAME, job submitted by OSI-TP side:
System error (errno = &ERRNO) occurred:
&ERRTXT

One of the system calls failed in the LU6.2 basic software (e.g. in TRANSIT) during the attempt to open a conversation to the LU6.2 partner. Similar to the RECEIVE_ALLOCATE call (see message 311!), TRANSIT does not provide any error text, and the following text is always output as the *&ERRTXT*:

No interpretation of errno with TRANSIT as the LU6.2 software

315 Actively allocated conversation to LU6.2 partner deallocated
(TP Name &TPNAME, job submitted by OSI-TP side)
again due to a configuration error:
The net name of the local and/or of the partner LU has been changed!

An already active, open conversation to the TP *&TPNAME* of the LU6.2 partner is closed immediately thereafter because an inconsistency was detected while checking the network names of the local and remote LU. This means that the alias names for the local and/or the remote LU (e.g. in the TRANSIT configuration) are not aliases of the same LUs (in the SNA network) as before. This message always precedes at least one of the messages 042 and 044.

316 Incoming conversation for TP &TPNAME not present any more:
Probably deallocated by issuing TP_ENDED in a collision
with RECEIVE_ALLOCATE!

A temporary error in the LU6.2 basic software led to an abort of an incoming conversation from the LU6.2 partner. *&TPNAME* designates the UTM transaction code.

320 Allocation of a conversation to LU6.2 partner failed,
TP Name &TPNAME, job submitted by OSI-TP side:
Allocation error (code = &ERRTXT)

No conversations to the TP *&TPNAME* of the LU6.2 partner can be actively opened at the moment. (*&TPNAME* is in this case the CICS transaction code, for example.) The error code *&ERRTXT* then contains the return code of the corresponding LU6.2 call in the plain text.

321 Allocation of a conversation to LU6.2 partner failed,
TP Name &TPNAME, job submitted by OSI-TP side:
Inconsistency in the configurations of openUTM-LU62 and the LU6.2 base software (MODENAME, LOC_LU_ALIAS, REM_LU_ALIAS)

Conversations to LU6.2 partners cannot be opened at all with the current generation of the LU6.2 basic software (e.g. TRANSIT) and openUTM-LU62 because the alias name of the local or remote LU or the name of the mode is not contained in the LU6.2 basic software configuration. This situation will exist until the LU6.2 basic software or the openUTM-LU62 entity is stopped, regenerated and restarted.

322 Allocation of a conversation to LU6.2 partner failed,
TP Name &TPNAME, job submitted by OSI-TP side:
Security data invalid

The LU6.2 basic software (e.g. TRANSIT) has rejected the request to open a conversation to the LU6.2 partner because the security data of the user ID and/or password contains a special character or because the partner has shown while opening the session that it does not support security data. *&TPNAME* is in this case the CICS transaction code, for example.

323 Allocation of a conversation to LU6.2 partner failed,
TP Name &TPNAME, job submitted by OSI-TP side:
The sync level &SYNCLEVEL is not supported by the LU6.2 base software.

The LU6.2 basic software (e.g. TRANSIT) has rejected the request to open a conversation to the LU6.2 partner because the requested sync-level *&SYNCLEVEL* SYNCPT (2) or CONFIRM (1) is not possible. If the LU6.2 partner supports the specified sync-level, then it is sufficient for TRANSIT to modify the PAIR_EXT generation parameter of the local LU in the TRANSIT configuration accordingly. *&TPNAME* is in this case the CICS transaction code, for example.

330 Conversation to the LU6.2 partner
(TP Name &TPNAME, job submitted by OSI-TP side)
terminated by the partner with ABEND!

The LU6.2 partner has closed the conversation abnormally with Deallocate(Abend). openUTM-LU62 therefore also terminates the OSI-TP dialog to the OSI-TP partner (i.e. to the UTM application) abnormally with U_ABORT_REQ. The actual initial error can generally be found in the application program of the LU6.2 partner.

331 Conversation to the LU6.2 partner
(TP Name &TPNAME, job submitted by LU6.2 side)
terminated by the partner with ABEND!

See comment on message 330.

332 Conversation to the LU6.2 partner
(TP Name &TPNAME, job submitted by OSI-TP side)
terminated by shutdown of the LU6.2 base software!

The LU6.2 basic software has terminated itself. When TRANSIT is used, this means that the TRANSIT base system or the LU62Mgr has been terminated. This is noted by openUTM-LU62 in the next LU6.2 basic software call and leads to the abnormal termination of the OSI-TP dialog with the OSI-TP partner (i.e. with the UTM application) using U_ABORT_REQ.

333 Conversation to the LU6.2 partner
(TP Name &TPNAME, job submitted by LU6.2 side)
terminated by shutdown of the LU6.2 base software!

See comment on message 332.

334 Conversation to the LU6.2 partner
(TP Name &TPNAME, job submitted by OSI-TP side)
terminated by conversation error!

A fatal error has occurred in the session that has opened the conversation to the LU6.2 partner, e.g. the link connection to the partner PU has failed or all sessions have been closed using the administration command `u62_adm -as`. openUTM-LU62 then closes the OSI-TP dialog to the OSI-TP partner (i.e. to the UTM application) abnormally.

335 Conversation to the LU6.2 partner
(TP Name &TPNAME, job submitted by LU6.2 side)
terminated by conversation error!

See comment on message 334.

336 User control data received on a conversation to the LU6.2 partner
(TP Name &TPNAME, job submitted by OSI TP side)
=> Termination of the conversation by openUTM-LU62!

The LU6.2 partner has sent openUTM-LU62 user control data. openUTM-LU62 closes the conversation to the LU6.2 partner and the OSI-TP dialog with the OSI-TP partner (i.e. with the UTM application) because this cannot be converted into the OSI-TP protocol.

337 User control data received on a conversation to the LU6.2 partner
(TP Name &TPNAME, job submitted by LU6.2 side)
=> Termination of the conversation by openUTM-LU62!

See comment on message 336.

338 Return code AP_STATE_CHECK received on a conversation to the LU6.2 partner (TP Name &TPNAME, job submitted by OSI TP side)
=> Termination of the conversation by openUTM-LU62!

The LU6.2 basic software has detected a formal encoding error in the data while receiving a data packet from the LU6.2 partner. The software then throws the data out, changes the direction of sending by itself and sends a negative acknowledge to the LU6.2 partner. openUTM-LU62 receives the error code AP_STATE_CHECK in this situation and closes the conversation and the associated OSI-TP dialog as a result.

339 Return code AP_STATE_CHECK received on a conversation to the LU6.2 partner (TP Name &TPNAME, job submitted by LU6.2 side)
=> Termination of the conversation by openUTM-LU62!

See comment on message 338.

340 Incoming conversation to the LU6.2 partner rejected,
TP Name &TPNAME, job submitted by LU6.2 side:
Expiration of the timer for the supervision of the association allocation

The LU6.2 partner has opened a conversation to the TP *&TPNAME*. (*&TPNAME* therefore designates the UTM transaction code.) openUTM-LU62 cannot open an OSI-TP dialog to the OSI-TP partner (i.e. to the UTM application) at the moment due to a lack of free parallel connections. If no parallel connection becomes free within a certain wait time, then openUTM-LU62 rejects the conversation. This maximum wait time is configurable in openUTM-LU62 using the parameter ALLOC-TIME, where the default value is 30 seconds. If this message appears often, then you should either lower the session limit (the generation parameter SESS-MAX for XMODE in TRANSIT) or increase the number of parallel connections for openUTM-LU62 (generation parameter ASSOCIATIONS).

341 Incoming dialogue to the OSI-TP partner rejected,
TP Name &TPNAME, job submitted by OSI-TP side:
Expiration of the timer for the supervision of the conversation allocation

The OSI-TP partner (i.e. the UTM application) has opened an OSI-TP dialog to the TP *&TPNAME*. (*&TPNAME* therefore designates the CICS transaction code, for example.) openUTM-LU62 cannot open a conversation to the LU6.2 partner at the moment due to a lack of free sessions. If no session becomes free within a specific wait time, then openUTM-LU62 rejects the OSI-TP dialog. This maximum wait time is configurable in openUTM-LU62 using the parameter ALLOC-TIME, where the default value is 30 seconds. If this message appears often, then you should either increase the session limit (in the generation parameter SESS-MAX for XMODE in TRANSIT) or decrease the number of parallel connections for openUTM-LU62 (generation parameter ASSOCIATIONS).

350 XAP-TP provider rejects association allocation
(TP &TPNAME): result = &RES, source = &SRC, reason = &RSN

openUTM-LU62 has a accepted an incoming conversation from the LU6.2 partner and attempts to reserve a parallel connection for an OSI-TP dialog to the TP *&TPNAME* of the OSI-TP partner (i.e. to the transaction code *&TPNAME* of the UTM application). However, the local XAP-TP provider has rejected the request. openUTM-LU62 closes the conversation to the LU6.2 partner as a result.

Meaning of *&RES*:

1	Association allocation request permanently rejected
2	Association allocation request temporarily rejected

Meaning of *&SRC*:

0	ACSE Service User
1	ACSE Service Provider
2	Presentation Service Provider
7	APM Service Provider (Association Pool Manager)

Meaning of *&RSN*:

-1	No reason given
0	CCR version 2 not available
1	Incompatible versions
2	Contention winner property rejected
3	Bid mandatory value rejected
6	Local or remote Application Entity Title (AET) invalid
7	No association pool found for the specified AET
58	All associations from the association pool are reserved although the pool limits allow the opening of additional associations. The XAP-TP provider has begun allocating additional associations.
59	All associations from the association pool are reserved and the pool limits do not allow the allocating of additional associations.
60	The association pool manager timer has run down. This timer specifies the maximum time to wait for the release of an association if all associations in the pool are reserved.
61	The association is to be reserved for a protected conversation and the current transaction is in the end phase. At this point in time, no more further OSI-TP dialogs can be added to the transaction tree.

351 XAP-TP provider reports loss of association
(TP &TPNAME): source = &SRC, reason = &RSN, event = &EVT

openUTM-LU62 has accepted an incoming conversation from the LU6.2 partner and has already successfully reserved a parallel connection for an OSI-TP dialog to the TP *&TPNAME* of the OSI-TP partner (i.e. to the transaction code *&TPNAME* of the UTM application). However, the parallel connection is lost before it has a chance to open the OSI-TP dialog. openUTM-LU62 closes the connection to the LU6.2 partner as a result.

Meaning of *&SRC*:

0	ACSE Service User
1	ACSE Service Provider
2	Presentation Service Provider
7	APM Service Provider (Association Pool Manager)

Meaning of *&RSN* when *&SRC* is equal to 0 or 1:

-1	A-RELEASE-IND: No reason given
0	A-RELEASE-IND: Normal release request
1	A-RELEASE-IND: Urgent release request
12	A-ABORT-IND
30	A-RELEASE-IND: User-defined release request

Meaning of *&RSN* when *&SRC* is equal to 2:

-1	No reason for abort given
0	Reason for abort unknown
1	Unknown Presentation Protocol Data Unit received
2	Unexpected Presentation Protocol Data Unit received
3	Unexpected Session Service Primitive received
4	Presentation Protocol Data Unit with unknown parameter received
5	Presentation Protocol Data Unit with unexpected parameter received
6	Presentation Protocol Data Unit with invalid parameter received

Meaning of *&RSN* when *&SRC* is equal to 7:

5	An OSI-TP dialog from the partner has reserved the association

352 OSI-TP partner (XAP-TP user) rejects begin dialogue request,
TP Name &TPNAME, job submitted by LU6.2 side

openUTM-LU62 has attempted to open a OSI-TP dialog to the TP *&TPNAME* of the OSI-TP partner (i.e. to the transaction code *&TPNAME* of the UTM application) due to an incoming conversation from the LU6.2 partner. The remote XAP-TP user (i.e. the UTM application program) has rejected the OSI-TP dialog, however, so that a TP_BEGIN_DIALOGUE confirmation with result = rejected has been received by openUTM-LU62. openUTM-LU62 terminates the conversation to the LU6.2 partner as a result.

353 OSI-TP partner (XAP-TP provider) rejects begin dialogue request,
TP Name &TPNAME, job submitted by LU6.2 side:
Reason = &RSN

openUTM-LU62 has attempted to open a OSI-TP dialog to the TP *&TPNAME* of the OSI-TP partner (i.e. to the transaction code *&TPNAME* of the UTM application) due to an incoming conversation from the LU6.2 partner. The remote XAP-TP provider (i.e. from openUTM itself) has rejected the OSI-TP dialog, however, so that a TP_BEGIN_DIALOGUE confirmation with result = rejected has been received by openUTM-LU62. openUTM-LU62 terminates the conversation to the LU6.2 partner as a result. The reason for the rejection can be found in the SYSLOG file of the UTM application.

Meaning of *&RSN*:

-1	No reason given
1	The TPSU title (transaction code) specified is not known to the partner
2	The TPSU title (transaction code) specified is permanently unavailable in the partner
3	The TPSU title (transaction code) specified is temporarily unavailable in the partner
4	The partner application has received a TP-BEGIN-DIALOGUE indication without the specification of a TPSU title
5	One or more of the desired functional units is not supported by the partner
6	The desired combination of functional units is not supported by the partner
7	The association that was allocated for the OSI-TP dialog is already being used by the partner application
8	The application entity parameter that was specified in the TP-BEGIN-DIALOGUE request does not identify any known application entity invocation

354 Diagnostic information in the initialization data of
AP_TP_BEGIN_DIALOGUE_CNF: 0xhhhh (d,d)

hhhh represents the hexadecimal value and d,d represent the according decimal values.

For UTM as OSI-TP partner, the initialization data contains further information on the reason, why the dialog has been rejected. The first value shows whether the fault is transient (1) or permanent (2).

The second value shows the detailed reason for rejection.

The meaning of these values can be taken from the manual
openUTM V5.3 messages, test and diagnosis in BS2000/OSD
(description DIA3 in message K119 for DIA1=2)

355 Protocol error:
AP_TP_ACCEPT received in AP_TP_BEGIN_DIALOGUE_CNF,
TP-Name &TPNAME, job submitted by LU6.2 side

The OSI-TP partner has acknowledged the incoming OSI-TP dialog positively although openUTM-LU62 has not requested an explicit acknowledge for opening the OSI-TP dialog. This protocol violation then leads to the OSI-TP dialog and the conversation to the LU6.2 partner being aborted. This message can only occur in an OSI-TP partner that is not a UTM application.

356 Protocol error:
AP_TP_BEGIN_TRANSACTION_IND received,
TP-Name &TPNAME, job submitted by LU6.2 side

openUTM-LU62 has received a BEGIN_TRANSACTION primitive from the OSI-TP partner. This is a serious violation of the OSI-TP protocol. As a result, openUTM-LU62 aborts the OSI-TP dialog and the conversation to the LU6.2 partner. This message can only occur in an OSI-TP partner that is not a UTM application.

357 Protocol error:
AP_TP_BEGIN_TRANSACTION_IND received,
TP Name &TPNAME, job submitted by OSI-TP side

See comment on message 356.

358 The length (&LEN) of the user data received from the OSI-TP partner exceeds the maximum value &MAXLEN:
TP-Name &TPNAME, job submitted by LU6.2 side

The OSI-TP partner (i.e. the UTM application) has sent data with a net length *&LEN* to openUTM-LU62. However, only *&MAXLEN* bytes - or 32763 bytes when TRANSIT is used as the LU6.2 basic software - can be transmitted in one data packet to the LU6.2 partner. openUTM-LU62 terminates the OSI-TP dialog and the conversation abnormally as a result. The UTM application program must therefore be changed.

359 The length (&LEN) of the user data received from the OSI-TP partner exceeds the maximum value &MAXLEN:
TP Name &TPNAME, job submitted by OSI-TP side

See comment on message 358.

360 Incoming dialogue rejected, job submitted by OSI-TP side:
No local TPSU title indicated

openUTM-LU62 has received a TP_BEGIN_DIALOGUE indication without the specification of a TPSU title, i.e. without the specification of the transaction code from the OSI-TP partner. The OSI-TP dialog request is rejected by openUTM-LU62. This message can only occur in an OSI-TP partner that is not a UTM application.

361 Incoming dialogue rejected, job submitted by OSI-TP side:
Cannot decode local TPSU title

An error has occurred while decoding the TPSU title contained in the TP_BEGIN_DIALOGUE indication (i.e. of the transaction code); the OSI-TP partner (i.e. the UTM application) has probably coded the TPSU title incorrectly. openUTM-LU62 therefore rejects the OSI-TP dialog request. This message can only occur in an OSI-TP partner that is not a UTM application.

362 Incoming dialogue rejected, job submitted by OSI-TP side:
Invalid type of local TPSU title

The type of TPSU title contained in the TP_BEGIN_DIALOGUE indication (i.e. of the transaction code) is not a PRINTABLE_STRING nor a T61_STRING. The OSI-TP dialog request is rejected because openUTM-LU62 only supports these two types.

363 Incoming dialogue rejected, job submitted by OSI-TP side:
the length of the local TPSU title (&LEN) exceeds the maximum value &MAXLEN

The decoded TPSU title contained in the TP_BEGIN_DIALOGUE indication (i.e. the CICS transaction code, for example) is *&LEN* characters long. However, only *&MAXLEN* characters (64 characters when TRANSIT is used as the LU6.2 basic software) are available when opening a conversation to the LU6.2 partner. openUTM-LU62 therefore rejects the OSI-TP dialog request.

364 Incoming dialogue rejected,
TP Name &TPNAME, job submitted by OSI-TP side:
Shared control requested

openUTM-LU62 has received a TP_BEGIN_DIALOGUE indication with the functional unit shared control from the OSI-TP partner. The OSI-TP dialog is aborted because this functionality is not supported by openUTM-LU62. This message can only occur in an OSI-TP partner that is not a UTM application.

365 Incoming dialogue rejected,
TP Name &TPNAME, job submitted by OSI-TP side:
Cannot decode remote application process title

openUTM-LU62 has received a TP_BEGIN_DIALOGUE indication with an incorrectly coded APT (application process title) or AEQ (application entity qualifier) from the OSI-TP partner. This message can only occur in an OSI-TP partner that is not a UTM application.

366 Incoming dialogue rejected,
TP Name &TPNAME, job submitted by OSI-TP side:
Invalid type of the remote Application Entity Qualifier

See comment on message 365.

369 Incoming dialogue rejected,
TP Name &TPNAME, job submitted by OSI-TP side:
CCR syntax not negotiated for this partner application

openUTM-LU62 has received a TP_BEGIN_DIALOGUE indication with the functional unit commit from the OSI-TP partner (i.e. from the UTM application) although a UDTAC or UDTDISAC application context was configured and therefore transaction management is not possible. openUTM-LU62 therefore rejects the OSI-TP dialog request.

If the functional unit commit is to be supported by openUTM-LU62, then a suitable application context that has been coordinated with the OSI-TP partner must be generated:

APPL-CONTEXT = UDTSEC (default) or
APPL-CONTEXT = UDTCCR

370 Incoming dialogue rejected,
TP Name &TPNAME, job submitted by OSI-TP side:
Explicit confirmation by DIALOGUE_RSP requested

The OSI-TP partner has requested an explicit confirmation while opening an OSI-TP dialog with a TP_BEGIN_DIALOGUE_RSP primitive. The OSI-TP dialog request will be rejected by openUTM-LU62 because the LU6.2 protocol does not have a direct equivalent for this. This message can only occur in an OSI-TP partner that is not a UTM application.

371 Incoming dialogue rejected,
TP Name &TPNAME, job submitted by OSI-TP side:
Unchained transactions not supported

openUTM-LU62 has received a TP_BEGIN_DIALOGUE indication with the functional unit commit_and_unchained from the OSI-TP partner. The LU6.2 protocol, however, does not provide an equivalent functionality, so openUTM-LU62 rejects the OSI-TP dialog request. This message can only occur in an OSI-TP partner that is not a UTM application.

373 Incoming dialogue rejected,
TP Name &TPNAME, job submitted by OSI-TP side:
Restart of the nominated control instance not completed yet

The OSI-TP partner (i.e. the UTM application) has opened a OSI-TP dialog to openUTM-LU62 with the functional unit commit at a time when the restart of the still existing transactions has not been completed after the start of openUTM-LU62. In this situation, the XAP-TP control entity is not yet able to support transaction management in new OSI-TP dialogs with the functional unit commit. openUTM-LU62 therefore rejects the OSI-TP dialog request. *&TPNAME* designates the TP name on the LU6.2 side, e.g. the CICS transaction code.

374 Incoming dialogue rejected,
TP Name &TPNAME, job submitted by OSI-TP side:
Remote application process title is not consistent with the configuration

375 Incoming dialogue rejected,
TP Name &TPNAME, job submitted by OSI-TP side:
Remote application entity qualifier is not consistent with the configuration

openUTM-LU62 has received a TP_BEGIN_DIALOGUE indication with an APT (application process title) or AEQ (application entity qualifier) from the OSI-TP partner (i.e. from the UTM application) that does not match the openUTM-LU62 generation. The corresponding parameters in the openUTM-LU62 generation that are to be modified are REM-APT and REM-AEQ.

376 Incoming dialogue rejected,
TP Name &TPNAME, job submitted by OSI-TP side:
Cannot decode application context name

openUTM-LU62 has received a TP_BEGIN_DIALOGUE indication with an incorrectly encoded application context name from the OSI-TP partner which has led to the rejection of the OSI-TP dialog request. This message can only occur in an OSITP partner that is not a UTM application.

377 Warning: Application context name is not configured.
Incoming dialogue to TP &TPNAME is accepted all the same!

openUTM-LU62 has received a TP_BEGIN_DIALOGUE indication with an application context name from the OSI-TP partner (i.e. from the UTM application) that does not match the openUTM-LU62 generation. This error is not viewed as a particularly serious error so that the conversation to the LU6.2 partner is still opened. However, the openUTM-LU62 generation parameter APPL-CONTEXT is to be modified accordingly!

378 Incoming dialogue rejected,
TP Name &TPNAME, job submitted by OSI-TP side:
Presentation context identifier not found

The presentation context identifier is a value that uniquely describes the (abstract syntax, transfer syntax) pair within the application context and that is required for the coding and decoding of the user and security data. This value is determined after a TP_BEGIN_DIALOGUE indication is received. If this is not possible, then a fatal error has occurred that will lead to the OSI-TP dialog being aborted.

379 Incoming dialogue rejected,
TP Name &TPNAME, job submitted by OSI-TP side:
Initialization data not supported

openUTM-LU62 has received a TP_BEGIN_DIALOGUE indication with appended initialization data from the OSI-TP partner. Such data is only allowed if the application context is UDTSEC. In this case, the data is the security data (user ID and password or the “Already verified“ indicator). For all other application contexts it is not clear how the initialization data is to be coded in order to transmit the data as PIP data in the ALLOCATE of the LU6.2 partner. For this reason, openUTM-LU62 rejects the OSI-TP dialog request here. This message can only occur in an OSI-TP partner that is not a UTM application.

380 Incoming dialogue rejected,
TP Name &TPNAME, job submitted by OSI-TP side:
Decoding security data failed

openUTM-LU62 has received a TP_BEGIN_DIALOGUE indication with initialization data from the OSI-TP partner. This data is interpreted as security data and decoded according to ASN.1 because the application context is UDTSEC. If the decoding fails, then this is a fatal error that will lead to the rejection of the OSI-TP dialog request. This message can only occur in an OSI-TP partner that is not a UTM application.

381 Incoming dialogue rejected,
TP Name &TPNAME, job submitted by OSI-TP side:
User-id or password too long:
Length of user-id = &ULEN, Length of password = &PLEN

openUTM-LU62 has received a TP_BEGIN_DIALOGUE indication with initialization data from the OSI-TP partner (i.e. from the UTM application) and has interpreted this data as security data (application context = UDTSEC). openUTM-LU62 has determined while decoding the data that the length of the specified user ID *&ULEN* or of the specified password *&PLEN* exceeds the maximum value of 10 that is defined for the LU6.2 interface. openUTM-LU62 rejects the OSI-TP dialog request because the security data of the LU6.2 partner cannot be transmitted in its entirety.

382 Incoming dialogue rejected,
TP Name &TPNAME, job submitted by OSI-TP side:
An empty user-id is not permitted

383 Incoming dialogue rejected,
TP Name &TPNAME, job submitted by OSI-TP side:
A user-id encoded as octet string is not permitted

384 Incoming dialogue rejected,
TP Name &TPNAME, job submitted by OSI-TP side:
An empty password is not permitted

385 Incoming dialogue rejected,
TP Name &TPNAME, job submitted by OSI-TP side:
A password encoded as octet string is not permitted

The four messages above show that the TP_BEGIN_DIALOGUE indication received from the OSI-TP partner (i.e. from the UTM application) contains an invalid user ID or an invalid password. Empty user ID's and passwords are not, however, supported by openUTM-LU62, just as user ID's and passwords that are encoded as octet strings are not supported because these cannot be converted to the LU6.2 protocol. Only user ID's and passwords that are encoded as a PRINTABLE_STRING or as a T61_STRING, are valid. If the “Already verified“ entry is specified instead of a password, then they are valid in this case, too.

386 Incoming dialogue rejected,
TP Name &TPNAME, job submitted by OSI-TP side:
Error in the exchange of the log names between the local and the remote LU!

The OSI-TP partner (i.e. the UTM application) has opened an OSI-TP dialog to openUTM-LU62 with the functional unit commit. However, before openUTM-LU62 can open a corresponding conversation with sync-level 2 to the LU6.2 partner and in case the number of sessions has dropped to 0 in the meantime, the log name must be exchanged again between the two LUs because the LU6.2 partner could have executed a cold start in the meantime. If an error occurs when the log names are exchanged, then the OSI-TP dialog request of the OSI-TP partner (i.e. the UTM application) is rejected. It does not matter in this case if the error that occurred was only temporary in nature (e.g. loss of the resync-TP session) or if the log names are actually inconsistent. An RS component (resync-TP) message that precedes this message informs you of the exact cause.

390 OSI-TP dialogue aborted by partner (XAP-TP provider)
TP Name &TPNAME, job submitted by LU6.2 side

391 OSI-TP dialogue aborted by partner (XAP-TP provider)
TP Name &TPNAME, job submitted by OSI-TP side

392 OSI-TP dialogue aborted by partner (XAP-TP user)
TP Name &TPNAME, job submitted by LU6.2 side

393 OSI-TP dialogue aborted by partner (XAP-TP user)
TP Name &TPNAME, job submitted by OSI-TP side

The OSI-TP dialog between openUTM-LU62 and the OSI-TP partner (i.e. the UTM application) has been terminated abnormally by the partner. If the cause is the remote XAP-TP provider (i.e. openUTM itself), then a TP_P_ABORT indication will have been received (messages 390 and 391). If the cause is the remote XAP-TP user (and therefore the application program), then openUTM-LU62 has received a TP_U_ABORT indication (messages 392 and 393). In any case, openUTM-LU62 closes the associated conversation to the LU6.2 partner using Deallocate(Abend) or Deallocate(Resource-Failure) as a result.

394 Received log data:

This message is output in addition to message 392 or 393 if the TP_U_ABORT indication contains log data. The data is logged in the hexadecimal format to the `prot.`*luname* file and then thrown out afterwards because this data cannot be transmitted to the LU6.2 partner.

399 Statistic:

```
Actively    established unprotected  conversations  = &NA1
Actively    established protected    conversations  = &NA2
Passively   established unprotected  conversations  = &NP1
Passively   established protected    conversations  = &NP2
-------------------------------------------------------------------------------------
Total number of all established      conversations  = &N3
```

This message is output every hour and when `u62_tp` is terminated. It informs the user of how many transactions have executed.

TM component messages

400 Initialization completed.

The `u62_tp` program has successfully completed the start phase. Transactions can only be started after this message is output.

401 Warm start.
Resynchronization of &NUMBER transactions required.

There were still *&NUMBER* transactions in an undefined state when openUTM-LU62 was terminated the last time. The open transactions must be resynchronized before new transactions can be started. The open transactions are described in more detail in the following message, message 402.

402 Information about open transaction:
TP Name &TPNAME, job submitted by &PARTNER side.
LUWID (LU6.2):
    LU name = &LUNAME
    instance = &INSTANCE
    sequence = &SEQUENCE
AAID (OSI-TP):
    size = &SIZE
    aaid = &AAID
Log records: &RECORDS
LU6.2 resync role: &ROLE

This message writes the log records found in the synclog file at start of the openUTM-LU62 entity.
&TPNAME designates the transaction code of the job-receiver that was called by the job-submitter.
&PARTNER can contain the value LU6.2 or OSI-TP. The UTM transaction code *&TPNAME* was called by the LU6.2 partner (e.g. by a CICS-program) for LU6.2. The transaction code *&TPNAME* was called by a UTM application program by the LU6.2 partner for OSI-TP (therefore the CICS transaction code *&TPNAME*, for example). The transaction can be identified uniquely by its LUWID and its AAID. The LU6.2 partner only knows the LUWID of the transaction, and it is output to the CICS log file when an error occurs, for example. The UTM application only knows the AAID of the transaction, and it is output to the SYSLOG of UTM when an error occurs.
&RECORDS records the status information of the transaction. The individual values are described below.

&ROLE describes the role of the openUTM-LU62 entity during a transaction restart to the LU6.2 partner. The individual values are described below.

RECORDS	**Description**
agent,committed	The LU6.2 partner has started the transaction, openUTM-LU62 has already committed the transaction, the LU6.2 partner may not be informed as yet of this fact.
agent,in-doubt	The LU6.2 partner has started the transaction, openUTM-LU62 cannot set forward or roll back the transaction at the moment.
agent,HM	The LU6.2 partner has started the transaction, the OSI-TP partner (i.e. the UTM application) has announced a heuristic mix, i.e. the participating databases are not consistent. The LU6.2 partner still needs to be informed.
agent,RIP	The LU6.2 partner has started the transaction, the OSI-TP partner (i.e. the UTM application) has announced a heuristic hazard, i.e. it is not sure if the participating databases are consistent. The LU6.2 partner still needs to be informed.
init,spm-pend	The OSI-TP partner (i.e. the UTM application) has started the transaction. No information has been received yet from the LU6.2 partner regarding whether it wants to roll the transaction forward or roll it back.
log-commit	openUTM-LU62 has already committed the transaction, the OSI-TP partner (i.e. the UTM application) may not be informed as yet of this fact.
log-ready	openUTM-LU62 cannot roll forward or roll back the transaction at the moment.
log-damage-mix	The LU6.2 partner has announced a heuristic mix, i.e. the participating databases are not consistent. The OSI-TP partner (i.e. the UTM application) still needs to be informed.
log-damage-haz	The LU6.2 partner has announced a “resync in progress“, i.e. it is not sure if the participating databases are consistent. The OSI-TP partner (i.e. the UTM application) still needs to be informed.

ROLE	**Description**
ACTIVE	openUTM-LU62 is responsible for the transaction restart for the LU6.2 partner.
PASSIVE	The LU6.2 partner is responsible for the transaction restart, i.e. it must send a “Compare States“.
NONE	A transaction restart for the LU6.2 partner is not needed from openUTM-LU62’s point of view.

403 Connection to LU6.2 partner lost.
Resynchronization of transaction required.
TP Name &TPNAME, job submitted by &PARTNER side.
LUWID (LU6.2):
    LU name = &LUNAME
    instance = &INSTANCE
    sequence = &SEQUENCE
AAID (OSI-TP):
    size = &SIZE
    aaid = &AAID

The connection to the LU6.2 partner was lost during the commit phase of a transaction. The transaction must be resynchronized after the connection is re-established.
See message 402 for the meanings of *&TPNAME*, *&PARTNER*, LUWID and AAID.

404 Next attempt to resynchronization in &TIME seconds.

This message is output when an error occurs during the resynchronization of a transaction, e.g. a connection is aborted and the resynchronization must be repeated again later.

405 Resynchronization of transaction successful.
TP Name &TPNAME, job submitted by &PARTNER side.
LUWID (LU6.2):
    LU name = &LUNAME
    instance = &INSTANCE
    sequence = &SEQUENCE
AAID (OSI-TP):
    size = &SIZE
    aaid = &AAID

A transaction aborted due to a loss of the connection or some other error has been successfully resynchronized.
See message 402 for the meanings of *&TPNAME*, *&PARTNER*, LUWID and AAID.

406 LU6.2 partner indicates heuristic mix.
TP Name &TPNAME, job submitted by &PARTNER side.
LUWID (LU6.2):
    LU name = &LUNAME
    instance = &INSTANCE
    sequence = &SEQUENCE
AAID (OSI-TP):
    size = &SIZE
    aaid = &AAID

The LU6.2 partner has terminated the transaction on its own, e.g. after a loss of the connection, without waiting for the UTM application. The transaction is therefore inconsistent. Changes to the database by UTM and LU6.2 that are to be coordinated by the transaction have been executed incorrectly. E.g., the requested database changes have been made on the LU6.2 side, but not on the UTM side.
See message 402 for the meanings of *&TPNAME*, *&PARTNER*, LUWID and AAID.
You usually need to change the participating database manually in this case to re-coordinate the inconsistent database records.

407 LU6.2 partner indicates resync in progress.
TP Name &TPNAME, job submitted by &PARTNER side.
LUWID (LU6.2):
    LU name = &LUNAME
    instance = &INSTANCE
    sequence = &SEQUENCE
AAID (OSI-TP):
    size = &SIZE
    aaid = &AAID

The LU6.2 partner has terminated the transaction on its own, e.g. after a loss of the connection, without waiting for the UTM application. It is unknown whether the transaction is inconsistent or not.
See message 402 for the meanings of &TPNAME, &PARTNER, LUWID and AAID.

The administrators of the databases participating in the transaction are to check to see if the databases are inconsistent.

408 OSI TP partner indicates heuristic mix.
TP Name &TPNAME, job submitted by &PARTNER side.
LUWID (LU6.2):
    LU name = &LUNAME
    instance = &INSTANCE
    sequence = &SEQUENCE
AAID (OSI-TP):
    size = &SIZE
    aaid = &AAID

The OSI-TP partner (i.e. the UTM application or the database on the UTM side) has terminated the transaction on its own, e.g. after a loss of the connection, without waiting for the LU6.2 partner. The transaction is therefore inconsistent. Changes to the database by UTM and LU6.2 that are to be coordinated by the transaction have been executed incorrectly. For example, the requested database changes have been made on the LU6.2 side, but not on the UTM side.
See message 402 for the meanings of &TPNAME, &PARTNER, LUWID and AAID.

You usually need to change the participating database manually in this case to recoordinate the inconsistent database records.

409 OSI TP partner indicates heuristic hazard.
TP Name &TPNAME, job submitted by &PARTNER side.
LUWID (LU6.2):
  LU name = &LUNAME
  instance = &INSTANCE
  sequence = &SEQUENCE
AAID (OSI-TP):
  size = &SIZE
  aaid = &AAID

It cannot be determined for sure whether the open transaction was executed or rolled back after a loss of the connection to the OSI-TP partner (i.e. to the UTM application).
See message 402 for the meanings of &TPNAME, &PARTNER, LUWID and AAID.

The administrators of the databases participating in the transaction are to check to see if the database is inconsistent.

410 Syncpoint request received from LU6.2 job-receiving service.
TP Name &TPNAME. Transaction is aborted.

A job-receiving program on the LU6.2 side has requested a syncpoint (e.g. EXEC CICS SYNCPOINT) without having been requested to do so by the UTM application program. This is prohibited in openUTM. The transaction is therefore aborted.
&TPNAME specifies the transaction code on the LU6.2 side.

The application program on the LU6.2 side, e.g. the CICS application program, must be changed.

420 Error during resynchronization of a transaction.
LU6.2 partner does not accept the state &STATE.
TP Name &TPNAME, job submitted by &PARTNER side.
LUWID (LU6.2):
  LU name = &LUNAME
  instance = &INSTANCE
  sequence = &SEQUENCE
AAID (OSI-TP):
  size = &SIZE
  aaid = &AAID
Possibly a cold start of openUTM-LU62 is needed.

A transaction has been interrupted in the commit phase due to a loss of the connection on the LU6.2 side. An error has occurred during resynchronization with the LU6.2 partner. The LU6.2 partner does not accept the transaction state suggested by openUTM-LU62.
See message 402 for the meanings of &TPNAME, &PARTNER, LUWID and AAID.

The transaction must be terminated manually in this case. On the openUTM-LU62 side, the only possibility to make a change manually is by terminating the affected openUTM-LU62 entity and then executing a cold start. If the LU6.2 partner is CICS, then the command “CEMT SET CONNECTION NOTPENDING“ must be entered in CICS. Note that database inconsistencies may arise due to this manual intervention. This possibility is to be examined by the administrators of the participating databases.

421 Log name error during resynchronization of a transaction.
TP Name &TPNAME, job submitted by &PARTNER side.
LUWID (LU6.2):
    LU name = &LUNAME
    instance = &INSTANCE
    sequence = &SEQUENCE
AAID (OSI-TP):
    size = &SIZE
    aaid = &AAID
Possibly a cold start of openUTM-LU62 is needed.

A transaction has been interrupted in the commit phase due to a loss of the connection on the LU6.2 side. An error has occurred during resynchronization with the LU6.2 partner. The LU6.2 partner does not accept the log name suggested by openUTM-LU62. The cause is probably a cold start of openUTM-LU62 or of the LU6.2 partner, or a “CEMT SET CONNECTION NOTPENDIG“ in CICS.
See message 402 for the meanings of &TPNAME, &PARTNER, LUWID and AAID.

If the cause is a cold start of the LU6.2 partner, then it should be checked whether it is possible to return to the previous application run, i.e. is the cold start can be reset. If this is not possible, then the openUTM-LU62 entity must be terminated and cold started. If the cause is a cold start of openUTM-LU62, then the LU6.2 partner must also execute a cold start. You do this in CICS using the “CEMT SET CONNECTION NOTPENDING“ command, for example.
Note that database inconsistencies may arise due to this manual intervention. This possibility is to be examined by the administrators of the participating databases.

422 Protocol error of LU6.2 partner:
&EVENT received in state &STATE1/&STATE2.
TP Name &TPNAME, job submitted by &PARTNER side.
LUWID (LU6.2):
    LU name = &LUNAME
    instance = &INSTANCE
    sequence = &SEQUENCE
AAID (OSI-TP):
    size = &SIZE
    aaid = &AAID
Transaction is aborted.

The LU6.2 partner has violated the LU6.2 protocol.
See message 402 for the meanings of &TPNAME, &PARTNER, LUWID and AAID.

If the protocol error is reproducible, then repeat the entire procedure with the trace activated and inform the systems support staff.

423 Protocol error of LU6.2 partner:
Compare States &LUWSTATE &RRI received in state &STATE1/&STATE2.
TP Name &TPNAME, job submitted by &PARTNER side.
LUWID (LU6.2):
    LU name = &LUNAME
    instance = &INSTANCE
    sequence = &SEQUENCE
AAID (OSI-TP):
    size = &SIZE
    aaid = &AAID

The LU6.2 partner has violated the LU6.2 protocol during the resynchronization of a transaction.
See message 402 for the meanings of &TPNAME, &PARTNER, LUWID and AAID.

If the protocol error is reproducible, then repeat the entire procedure with the trace activated and inform the systems support staff. The openUTM-LU62 entity and the LU6.2 partner may need to be cold started to clear up the situation. See message 421 for more information.

424 LU6.2 partner indicates a protocol error.
State: &STAT1/&STATE2,
TP Name &TPNAME, job submitted by &PARTNER side.
LUWID (LU6.2):
    LU name = &LUNAME
    instance = &INSTANCE
    sequence = &SEQUENCE
AAID (OSI-TP):
    size = &SIZE
    aaid = &AAID

The LU6.2 partner is of the opinion that openUTM-LU62 has violated the LU6.2 protocol. If the protocol violation is announced during normal operations, then the transaction is rolled back.
See message 402 for the meanings of &TPNAME, &PARTNER, LUWID and AAID.

If the protocol error is reproducible, then repeat the entire procedure with the trace activated and inform the systems support staff. It is possible that the database is now inconsistent. The openUTM-LU62 entity and the LU6.2 partner may need to be cold started to clear up the situation. See message 421 for more information.

425 Invalid log record (error type &ERROR).
Transaction will be removed.
LUWID (LU6.2):
    LU name = &LUNAME
    instance = &INSTANCE
    sequence = &SEQUENCE
AAID (OSI-TP):
    size = &SIZE
    aaid = &AAID

A faulty log record was read during a warm start of openUTM-LU62. The transaction recorded in this log record can therefore not be resynchronized.
See message 402 for the meanings of LUWID and AAID.

If there are still some diagnostics documents left over from the previous openUTM-LU62 run, then save these and give them to the systems support staff. It is possible that the database is now inconsistent. The openUTM-LU62 entity and the LU6.2 partner may need to be cold started to clear up the situation. See message 421 for more information.

426 Problem at the XAP-TP interface.
&EVENT received.

An error has occurred on the XAP-TP interface that is used for the communication between openUTM-LU62 and the UTM application.

Inform the systems support staff.

427 Error &ERRNR when issuing &CALL.

An error has occurred in a UNIX system call. This can lead to the abnormal termination of openUTM-LU62.

RS component messages

510 The LU (alias name = &ALIASNAME, net name = &NETNAME)
has returned an abnormal reply to the Exchange Logname command. This LU has detected a warm/cold mismatch or a logname mismatch.

The remote LU, identified by the alias name *&ALIASNAME* in the LU6.2 basic software (e.g. TRANSIT) or by the network name *&NETNAME* in the SNA network, has detected a fatal error while exchanging the log name. One possible cause is a cold start of openUTM-LU62 although the LU6.2 partner still needs to resynchronize transactions.

If some transactions still need to be resynchronized and the openUTM-LU62 synclog file has not been deleted yet, then you should attempt a warm start because otherwise you run the risk of introducing database inconsistencies. If this is not possible, then you must either cold start the partner LU or abnormally terminate any existing transactions (“logical units of work“) in the LU6.2 partner. No OSI-TP dialogs with the functional unit commit and no conversations with sync-level 2 are supported by openUTM-LU62 as long as the error has not been corrected.

511 A cold start has been attempted by LU
(alias name = &ALIASNAME, net name = &NETNAME),
but the local LU has logical units of work that are awaiting resynchronization from the previous activation.

The LU6.2 partner has executed a cold start although there are transactions (logical units of work) to openUTM-LU62 that still need to be resynchronized. If possible, the cold start of the LU6.2 partner is to be reset because otherwise you run the risk of introducing database inconsistencies. If this is not possible, then the situation can only be corrected by terminating the openUTM-LU62 entity and then executing a cold start.

512 The LU (alias name = &ALIASNAME, net name = &NETNAME)
does not have the same memory as does the local LU of the previous activation between them.

The LU6.2 partner has executed a warm start in which the values of the data it has stored (network name of its own LU, its own log name, the log name from openUTM-LU62) do not match the values that openUTM-LU62 has stored for this data.

This error situation is so serious that openUTM-LU62 cannot support any OSI-TP dialogs with the functional unit commit or any conversations with sync-level 2 until all sessions with the LU6.2 partner have been closed and the log names are successfully exchanged.

513 A format error has been detected in a sync point message received from LU (alias name = &ALIASNAME, net name = &NETNAME).

One of the syncpoint messages “Exchange log name“ or “Compare states“ that was received by the partner LU (alias name in the LU6.2 basic software generation and openUTM generation = *&ALIASNAME*, network name in the SNA network = *&NETNAME*) contained a format error. In this case, openUTM-LU62 rejects all incoming OSI-TP dialogs with the functional unit commit that wait for the result of the log names, but the error is not viewed as a permanent error. This means that in any case the exchange of the log names is to be initiated before the next conversation is opened with sync-level 2, regardless of whether or not the number of sessions to the LU6.2 partner has dropped down to 0 in the meantime.

If the format error keeps on occurring, then transaction-oriented communication with the LU6.2 partner is not possible.

515 The partner LU (alias name = &ALIASNAME, net name = &NETNAME) detected a protocol violation in a sync point message sent by the local LU.

The partner LU (alias name in the LU6.2 basic software generation and openUTM generation = *&ALIASNAME*, network name in the SNA network = *&NETNAME*) has detected a format error in a syncpoint message from openUTM-LU62. openUTM-LU62 reacts in the same manner as described in message 513:

Reject the OSI-TP dialogs using the functional unit commit that are still waiting at the next opportunity, but repeat the exchange of the log names.

If the format error keeps on occurring, then transaction-oriented communication with the LU6.2 partner is not possible.

516 A log name exchange with LU
(alias name = &ALIASNAME, net name = &NETNAME)
has failed. This LU issued a cold Exchange Logname. The failure may be caused by a warm/cold mismatch detected by the partner LU.

The partner LU (alias name in the LU6.2 basic software generation and openUTM-generation = *&ALIASNAME*, network name in the SNA network = *&NETNAME*) has detected a fatal error while exchanging the log names, probably a warm/cold error. Conversations with sync-level 2 cannot be opened any more after this point in time. Generally, the only recourse is to administratively terminate all existing transactions (logical units of work) in the LU6.2 partner or to execute a cold start of the partner directly. Note, however, that database inconsistencies may arise due to this.

520 Active allocation of a conversation to the resync TP X’06F2’ failed:
allocation error (code = &ERRTXT)

Conversations to the resync-TP of the LU6.2 partner cannot be actively opened at the moment so that the log names cannot be exchanged and it is not possible to compare transaction states. The error code *&ERRTXT* then contains the return code of the corresponding LU6.2 call in plain text. openUTM-LU62 rejects all OSI-TP dialogs using the functional unit commit that are still waiting, but repeats the exchange of the log names at the next opportunity.

521 Active allocation of a conversation to the resync TP X’06F2’ failed:
Inconsistency in the configurations of openUTM-LU62 and the LU6.2 base software (LOC_LU_ALIAS, REM_LU_ALIAS)

No conversations are currently possible between openUTM-LU62 and the LU6.2 partner due to the inconsistencies in the LU6.2 basic software (e.g. TRANSIT) and openUTM-LU62 configurations. As described for message 321, the error situation can only be corrected by stopping, regenerating and then restarting the LU6.2 basic software and/or openUTM-LU62.

523 Active allocation of a conversation to the resync TP X’06F2’ failed:
The sync level AP_CONFIRM_SYNC_LEVEL is not supported by the LU6.2 base software!

openUTM-LU62 has attempted to open a conversation to the resync-TP of the LU6.2 partner to exchange log names or to compare transaction states. This conversation always uses the CONFIRM sync-level (1). However, the LU6.2 basic software only allows conversations between the local and remote LU with sync-level NONE (0). The PAIR_EXT configuration parameter must be changed for this reason in TRANSIT.

The sync-level in the PAIR_EXT parameter should be set to the same value as in SYNCLEVEL - but naturally only if the LU6.2 partner also supports this level - because the resync-TP can only be initiated if the OSI-TP partner (i.e. the UTM application) desired transaction management.

524 Active allocation of a conversation to the resync TP X’06F2’ failed:
The LU6.2 base software is not running!

Communication via LU6.2 is currently not possible because the LU6.2 basic software (for TRANSIT the TRANSIT base system or the LU62Mgr) has not been started. All OSI-TP dialogs to the OSI-TP partner (to the UTM application) are therefore rejected at the moment.

525 Active allocation of a conversation to the resync TP X’06F2’ failed due to the following configuration error:
The local LU alias name &ALIASNAME is not configured!

This message is output if the first call to start the local resync-TP fails because the local LU alias name specified in *&ALIASNAME* is not configured in the LU6.2 basic software. The measures to take in this case are described in message 521.

526 Active allocation of a conversation to the resync TP X’06F2’ failed:
System error (errno = &ERRNO) occurred:
&ERRTXT

One of the system calls in the LU6.2 basic software (e.g. TRANSIT) failed while attempting to open a conversation to the LU6.2 partner. As described for messages 311 and 314, TRANSIT does not provide any error text and the following text always appears as the *&ERRTXT*:

No interpretation of errno with TRANSIT as LU6.2 software

This temporary error only leads to the rejection of all OSI-TP dialogs using the functional unit commit to the OSI-TP partner (i.e. to the UTM application) that wait for the end of the exchange of the log names. A failed “Compare States“ is repeated after a certain wait time for a specific LUW.

530 Incoming basic conversation to TP &TPNAME rejected:
The only TP that supports basic conversations is the resync TP X’06F2’.

The LU6.2 partner has opened a basic conversation to the TP *&TPNAME* of openUTM-LU62. openUTM-LU62 rejects the conversation with the TP_NOT_AVAIL_NO_RETRY code because basic conversations are reserved just for special transaction programs, of which openUTM-LU62 only supports the resync-TP X’06F2’.

531 Incoming conversation to the resync TP X'06F2' rejected:
The indicated sync level &SYNCLEVEL does not equal AP_CONFIRM_SYNC_LEVEL!

The LU6.2 partner has violated the protocol by starting the resync-TP with a sync-level not equal to CONFIRM (1). This message therefore points out a blatant violation committed by the LU6.2 partner.

532 Incoming conversation to the resync TP X'06F2' rejected:
The CCR syntax is not configured!

Conversations with sync-level 2 to the LU6.2 partner are not allowed any more because a UDTAC or UDTDISAC application context was configured, and transaction management is therefore not possible on the OSI-TP side. It is therefore not necessary to exchange the log names via the resync-TP. If you want to use conversations with sync-level 2, then you must use a UDTSEC or UDTCCR application context. If you do not want to use such conversations, then you can ignore this message.

533 Incoming conversation to the resync TP X'06F2' rejected:
Initialization data not supported

This message points out a protocol violation of the LU6.2 partner that was sent in the PIP data while opening the conversation to the resync-TP.

534 Incoming conversation to the resync TP X'06F2' rejected:
The alias name &ALSNAME of the partner LU is not identical with the configured name &CONFNAME.

The LU6.2 partner wants to open a conversation to the local resync-TP. The LU name of the LU6.2 partner, however, does not match the name generated in openUTM-LU62 using REM-LU-ALIAS. There is therefore an error in the overall configuration. An LU residing on the local computer that is to be used by openUTM-LU62 as a substitute LU for an UTM application can only communicate with a certain partner LU.

540 Conversation to the resync TP X'06F2' crashed down:
The LU6.2 base software has been stopped!

541 Conversation to the resync TP X'06F2' crashed down:
Conversation Failure!

This error message is output if the conversation to the resync-TP has been aborted due to a fatal error in the underlying session, possibly due to the breakdown of the link connection to the partner PU or because the administration has terminated all existing sessions to the partner LU (`u62_adm -as`). All OSI-TP dialogs using the functional unit commit that wait for the end of the exchange of the log names are rejected because this error is only temporary in nature. However, the exchanging of the log names is initiated again as soon as the next OSI-TP dialog using the functional unit commit arrives. You do not need to restart openUTM-LU62 after the problem with the connection has been corrected.

Other messages from the BD component on Windows systems

900 Listener socket cannot be opened, errno &ERRNO (&ERRTEXT)

On startup, each openUTM-LU62 entity creates a listener socket for accepting connection requests from the administration programs. If the socket() system function fails, this message is output and the entity shuts down.

901 Listener socket cannot be bound to port, errno &ERRNO (&ERRTEXT)

A free port >= 5000 is not available for openUTM-LU62. After this message is output, the entity shuts down.

902 Port file &PORTFILE cannot be created,
errno &ERRNO (&ERRTEXT)

903 Write to port file &PORTFILE failed,
errno &ERRNO (&ERRTEXT)

904 Write to port file &PORTFILE failed,
written: &NUM1 bytes, expected: &NUM2 bytes

An error occurred in the open() or write() system function while creating or writing to the &PORTFILE file, which should contain the port of the entity's listener socket in binary form. The entity then shuts down.

905 Accepting incoming TCP/IP connection failed, errno &ERRNO (&ERRTEXT)

The accept() system function for accepting an incoming connection request from an administration program failed with the error code ERRNO. This message is output for information purposes only, and does not cause the entity to shut down.

10.2 Messages from the XAP-TP provider

The messages from the XAP-TP provider all start with the letter “P”. The values for the inserts are either described following the message or (if the insert occurs a number of times) in the section “General inserts for the XAP-TP messages” on page 337.

Without using TNSX, the local Access Points and partner addresses are displayed in the messages as follows:
T'LOCTSEL'(26544)
T'REMTSEL'(host:5632)
The according inserts are marked with an asterisk (*).

P001 Error on OSS call (&XPFUNC):
&XPRET, &XPERR, &XP1INFO, &XP2INFOThis message is output if a call to an OSS function (*&XPFUNC*) returns an error. If the error has been reported by the transport system, message P012 is also output.

The inserts have the following meaning:

&XPFUNC	Name of the OSS function
&XPRET	See table on page 337
&XPERR	See table on page 337
&XP1INFO	Supplementary OSS information
&XP2INFO	Supplementary OSS information

P002 Error on association establishment (&XPFUNC)
&ACPNT, &OSLPAP, &XPRET, &XPERR, &XP1INFO, &XP2INFO

This message is issued if the call to an OSS function (*&XPFUNC*) required to establish an association returns an error. If the error has been reported by the transport system, message P012 is also output. If the error has not been reported by the transport system, the application is terminated with “Termapplication”.

The inserts have the following meaning:

&XPFUNC	Name of the OSS function
&ACPNT	Name of the local ACCESS-POINT (*)
&OSLPAP	Name of the partner in the local application (*)
&XPRET	See table on page 337
&XPERR	See table on page 337
&XP1INFO	Supplementary OSS information
&XP2INFO	Supplementary OSS information

P003 Association rejected (a_assin()):
&ACPNT, reason: &XPRJCT, length: &XPLTH

This message is issued if a request to establish an association was rejected from outside.

The inserts have the following meaning:

&ACPNT	Name of the local ACCESS-POINT (*)
&XPLTH	Incorrect length
&XPRJCT	See table on page 340

P004 Association rejected (a_assin()):
&ACPNT
&OSLPAP
Reason: &XPRJCT

This message is issued if a request to establish an association was rejected from outside.

The inserts have the following meaning:

&ACPNT	Name of the local ACCESS-POINT (*)
&OSLPAP	Name of the partner in the local application (*)
&XPRJCT	See table on page 340

P005 Association rejected (a_assin()):
&ACPNT, reason: unknown partner,
partner address (*)

This message is issued if a request to establish an association was rejected from outside because the remote partner is not known to the local application.

The inserts have the following meanings:

&ACPNT	Name of the local ACCESS-POINT (*)

P006 Association rejected (a_assin()):
&ACPNT, &OSLPAP, reason: wrong application context name
(&XP0OBID, &XP1OBID, &XP2OBID, &XP3OBID, &XP4OBID,
&XP5OBID, &XP6OBID, &XP7OBID, &XP8OBID, XP9OBID)

This message is issued if a request to establish an association was rejected from outside. The application context name for the remote partner does not match the application context name generated for this partner in the local application.

The inserts have the following meaning:

&ACPNT	Name of the local ACCESS-POINT (*)

&OSLPAP	Name of the partner in the local application (*)
&XP0OBID - *&XP9OBID*	These are (up to) ten elements of the object identifier which form the application context name of the remote partner. *-1* is output for elements which do not have a value assigned.

P007 Error on association establishment (a_assrs()):
&ACPNT, &OSLPAP, &XPRET, &XPERR, &XP1INFO, &XP2INFO This message is output when a call to the OSS function *a_assrs()* to respond to a request to establish an association from outside returns an error. If the error has been reported by the transport system, message P012 is also output.

The inserts have the following meaning:

&ACPNT	Name of the local ACCESS-POINT (*)
&OSLPAP	Name of the partner in the local application (*)
&XPRET	See table on page 337
&XPERR	See table on page 337
&XP1INFO	Supplementary OSS information
&XP2INFO	Supplementary OSS information

P008 Association established: &ACPNT, &OSLPAP

This message is issued when an association has been established.

The inserts have the following meaning:

&ACPNT	Name of the local ACCESS-POINT (*)
&OSLPAP	Name of the partner in the local application (*)

P009 Association rejected (a_asscf()):
&ACPNT, &OSLPAP, reason: &XPRJCT, length: &XPLTH

This message is issued when active establishment of an association is rejected because the confirmation from the partner cannot be accepted.

The inserts have the following meaning:

&ACPNT	Name of the local ACCESS-POINT (*)
&OSLPAP	Name of the partner in the local application (*)
&XPRJCT	See table on page 340
&XPLTH	Possible incorrect length

P010 Association rejected (a_asscf()):
&ACPNT, &OSLPAP, reason: unknown partner
Partner address: &PARTADDR (*)

(&XASSREF)
This message is issued when active establishment of an association is rejected, because the remote partner confirms establishment of an association with an address (*&XPADDR*) which is unknown to the local application.

The inserts have the following meaning:

&ACPNT	Name of the local ACCESS-POINT (*)
&OSLPAP	Name of the partner in the local application (*)
&PARTADDR	Real partner address
&XASSREF	XAPTP-internal association reference

P011 Association rejected (a_asscf()):
&ACPNT, &OSLPAP, reason: wrong application context name
(&XP0OBID, &XP1OBID, &XP2OBID, &XP3OBID, &XP4OBID,
&XP5OBID, &XP6OBID, &XP7OBID, &XP8OBID, XP9OBID)

This message is issued when active establishment of an association is rejected, because the remote partner confirms establishment of an association with an application context name other than the one configured for this partner in the local application.

The inserts have the following meanings:

&ACPNT	Name of the local ACCESS-POINT (*)
&OSLPAP	Name of the partner in the local application (*)
&XP0OBID - *&XP9OBID*	These are (up to) ten elements of the object identifier which form the application context name of the remote partner. *-1* is output for elements which do not have a value assigned.

P012 CMX diagnostic information:
&XPCTYPE, &XPCCLS, &XPCVAL

This message is issued if a preceding message is issued as a result of an error reported by the transport system. The diagnostic code of the transport system is print-edited. The following table describes a number of values for *&XPCTYPE*, *&XPCCLS* and *&XPCVAL*. The CMX header file *cmx.h* contains a complete list.

XPCTYPE	Meaning (CMX error type)
0	T_CMXTYPE: CMX error detected by the CMX library
2	T_DSTEMPERR: Temporary TNS error
3	T_DSCALL_ERR: TNS call error
4	T_DSPERM_ERR: Permanent TNS error
5	T_DSWARNING: TNS warning
>15	CMX error on the basis of error codes from the transport system

XPCCLS	Meaning (CMX error class, valid for &XPCTYPE < 15)
0	T_CMXCLASS: CMX class
2	T_DSNOT_SPEC: TNS class not specified
3	T_DSPAR_ERR: TNS parameter error
4	T_DSILL_VERS: Invalid TNS version
5	T_DSSYS_ERR: TNS system error
6	T_DSINT_ERR: Internal TNS error
7	T_DSMESSAGE: TNS note

XPCVAL	Meaning (CMX error value)
0	T_NOERROR: No error
5	T_EIO: Temporary bottleneck or error in the transport system
14	T_EFAULT: IO_Area not allocated
100	T_UNSPECIFIED: Unspecified error, generally a system call error
101	T_WSEQUENCE: Invalid call sequence
103	T_WPARAMETER: Invalid parameter
104	T_WAPPLICATION: The application is not known to TNS of the task is not authorized to sign on to the application or the application has already been opened by this task.
105	T_WAPP_LIMIT: The limit for the number of simultaneously active applications has already been reached.
106	T_WCONN_LIMIT: The limit for the number of simultaneously active connections has already been reached.
107	T_WTREF: Invalid transport reference or the transport connection has already been established.
111	T_NOCCP: The transport system does not support the requested application or connection.
114	T_CCP_END: The transport system has been terminated or the application was closed by the administrator.
255	T_WLIBVERSION: No connection to the CMX subsystem possible.
-100	T_INVREF: Invalid evid. CMX cannot assign the call to a wait point.

P013 Association rejected (a_asscf()):
&ACPNT, &OSLPAP, reason: &XPCRES, &XPSRC, &XPNDIA
CCR V2 = &XP1BOOL, Version Incompatibility = &XP2BOOL,
ContWin Assignment rejected = &XP3BOOL,
Bid mandatory rejected = &XP4BOOL, No reason = &XP5BOOL

This message is issued when active establishment of an association is rejected by the remote partner.

The inserts have the following meaning:

&ACPNT	Name of the local ACCESS-POINT (*)
&OSLPAP	Name of the partner in the local application (*)
&XPCRES	Specifies whether the rejection is temporary or permanent: 0= permanent reject 1= transient reject
&XPCSRC	Specifies who has rejected establishment of the association: 0 = ACSE service user 1 = ACSE service provider 2 = Presentation service provider
&XPNDIA	See table on page 344

&XP1BOOL - *&XP5BOOL*

These inserts can take the values *TRUE* or *FALSE*. Values of *TRUE* indicate the reasons the partner reported for rejecting the request to establish an association:

&XP1BOOL: CCR Version 2 is not available
&XP2BOOL: The TP protocol versions are not compatible
&XP3BOOL: The contention winner assignment has been rejected
&XP4BOOL: The specification “Bidding is mandatory” or “Bidding is not mandatory” has been rejected
&XP5BOOL: No reason is specified

P014 Error on association establishment (&XPFUNC)
&ACPNT, &OSLPAP, &XPRET, &XPERR, &XP1INFO, &XP2INFOThis message is issued if the call to an OSS function (*&XPFUNC*) required to establish an association returns an error. If the error has been reported by the transport system, message P012 is also output. If the error has not been reported by the transport system, the application is terminated with “Termapplication”.

The inserts have the following meanings:

&XPFUNC	Name of the OSS function
&ACPNT	Name of the local ACCESS-POINT (*)
&OSLPAP	Name of the partner in the local application (*)
&XPRET	See table on page 337
&XPRER	See table on page 337
&XP1INFO	Supplementary OSS information
&XP2INFO	Supplementary OSS information, currently always set to zero.

P015 Association disconnected (&XPFUNC)
&ACPNT, &OSLPAP, &XPLNK, &XPSRC, &XPNDIA,
&XPINI, &XP1INFO, &XP2INFOThis message is issued when an association is cleared.
The inserts have the following meaning:

&XPFUNC	Name of the OSS function
&ACPNT	Name of the local ACCESS-POINT (*)
&OSLPAP	Name of the partner in the local application (*)
&XPLNK	Represents the internal status of the association 0 = Association not linked 1 = Association linked to channel 2 = Association linked to instance
&XPCSRC	Originator of clear-down 0 = ACSE service user 1 = ACSE service provider 2 = Presentation service provider
&XPNDIA	See table on page 344
&XP1INFO	Supplementary OSS information
&XP2INFO	Supplementary OSS information
&XPINI	See table below:

XPINI	Meaning
0	Association was cleared internally.
401	O_LOC_TRAN The originator is the local transport system. &XP1INFO contains the CMX return code. This is output in detail in the subsequent message P012.
402	O_REM_TRAN The originator is the remote transport system. &XP1INFO contains the reason for the CMX event t_disin. The values are defined in *cmx.h*. Some of the possible values for &XP1INFO are contained in the list below: 0 (T_USER) The communication partner cleared the association, possibly as a result of a user error on the partner side. 1 (T_RTIMEOUT) The connection was cleared locally by CMX because the connection had been inactive for too long according to the t_timeout parameter. 2 (T_RADMIN) The connection was cleared locally by CMX because the administrator closed down CCP. 3 (R_CCPEND) The connection was cleared locally by CMX because CCP failed. 256 (T_RUNKOWN) Either the partner or CCP cleared the connection. No reason was given. 257 (T_RSAP_CONGEST) The partner CCP cleared the connection because of a TSAP-specific bottleneck. 258 (T_RSAP_NOTATT) The partner CCP cleared the connection because the addressed TSAP was not registered there. 259 (T_RUNSAP) The partner CCP cleared the connection because the addressed TSAP was not known there. 261 (T_RPERMLOST) The connection was cleared by the network administrator or the partner CCP administrator. 262 (T_RSYSERR) Error in the network 385 (T_RCONGEST) The partner CCP cleared the connection as a result of a resource bottleneck.

XPINI	Meaning
	386 (T_RCONNFAIL) No connection could be established. The partner CCP aborted the attempt to do so.
	387 (T_RDUPREF) The partner CCP cleared the connection because a second connection reference was assigned for an NSAP pair (system error).
	388 (T_RMISREF) The partner CCP cleared the connection because a connection reference could not be assigned (system error).
	389 (T_PROTERR) The partner CCP cleared the connection because of a protocol error (system error).
	391 (T_PREFOFLOW) The partner CCP cleared the connection because of a connection reference overflow.
	392 (T_RNOCONN) The partner CCP rejected the request to establish a network connection.
	394 (T_RINLNG) The partner CCP cleared the connection because of an incorrect length header or parameter (system error).
	448 (T_RLCONGEST) The local CCP cleared the connection because of a resource bottleneck.
	449 (T_RLNOQOS) The local CCP cleared the connection because the “quality of service” could not be maintained.
	451 (T_RILLPWD) Invalid connection password.
	452 (RNETACC) Access to the network was refused.
	464 (T_RLPROTERR) The local CCP cleared the connection because of a transport protocol error (system error).
	465 (T_RLINTIDU) The local CCP cleared the connection because it received an interface data unit which was too long (system error).
	466 (T_RLNORMFLOW) The local CCP cleared the connection because of an infringement of the flow control rules for normal data (system error).

XPINI	Meaning
	467 (T_RLEXFLOW) The local CCP cleared the connection because of an infringement of the flow control rules for expedited data (system error).
	468 (T_RLINSAPID) The local CCP cleared the connection because it received an invalid TSAP identification (system error).
	469 (T_RLINCEPID) The local CCP cleared the connection because it received an invalid TCEP identification (system error).
	470 (T_RLINPAR) The local CCP cleared the connection because of an invalid parameter value (e.g. user data too long or expedited data not permitted).
	480 (T_RLNOPERM) The administrator of the local CCP prevented establishment of a connection.
	481 (T_RLPERMLOST) The administrator of the local CCP cleared the connection.
	482 (T_RLNOCONN) The local CCP could not establish the connection because no network connection is available.
	483 (T_RLCONNLOST) The local CCP cleared the connection because the network connection was lost. Most common cause: generation error on the CCP and PDN side, e.g. incorrect link addresses. Other possible causes: partner is not available, modem is faulty or has been set incorrectly, data transfer connection not plugged in, data transfer card faulty.
	484 (T_RLNORESP) The local CCP cannot establish the connection, because the partner did not respond to the CONRQ.
	485 (T_RLIDLETRAF) The local CCP cleared the connection because the connection was lost (Idle Traffic Timeout).
	486 (T_RLRESYNC) The local CCP cleared the connection because it was not possible to resynchronize (more than ten attempts were made).
	487 (T_RLEXLOST) The local CCP cleared the connection because the expedited data channel is faulty (more than three attempts were made).

XPINI	Meaning
403	O_LOC_SESS The originator is the local session provider. &XP1INFO can take the following values: 4 (S_PROTERROR) Protocol error: Incorrect establishment of the session PDU or incorrect SPDU parameter 16 (S_PICSREST) Violation of implementation-specific restrictions.
404	O_REM_SESS The originator is the remote session provider. &XP1INFO can take the following values: 1 (S_TCDISCON) transport disconnect 4 (S_PROTERROR) protocol error 8 (S_UNDEFINED) undefined 16 (S_PICSREST) violation against restriction stated in PICS
405	O_LOC_PRES The originator is the local presentation provider. &XP1INFO can take the following values: 0 (P_ARRNO) reason not specified A decoding buffer requested internally cannot be provided due to a lack of memory. Overflow of the internal data buffer when reassembling fragmented messages An unknown session event was reported. System bottleneck or system error. 1 (P_ARNRPDU) unrecognized PPDU No session user data is available or the presentation part of the session user data cannot be decoded (system error). 4 (P_ARNRPAR) unrecognized PPDU parameter Error on decoding the ACSE, presentation or user syntax. 5 (P_ARNEPAR) unexpected PPDU parameter PPDU parameter not in normal mode.

XPINI	Meaning
	6 (P_ARNIPAR) invalid PPDU parameter Invalid context identifier on decoding. Invalid PPDU parameter, e.g. incorrect length. This “abort” can be triggered by the UTM user by specifying invalid presentation or session selectors.
406	O_REM_PRES The originator is the remote presentation provider. &XP1INFO can take the following values: -1 (O_NOVALUE) Optional parameter is not present 0 (P_ARNNO) Reason not specified 1 (P_ARNRPDU) Unrecognized PPDU 2 (P_ARNEPDU) Unexpected PPDU 3 (P_ARNESSP) Unexpected session service primitive 4 (P_ARNRPAR) Unrecognized PPDU parameter 5 (P_ARNEPAR) Unexpected PPDU parameter 6 (P_ARNIPAR) Invalid PPDU parameter value
407	O_LOC_ACSE The originator is the local ACSE provider &XP1INFO always has the following value: 1 (A_ABSASP) ACSE service provider initiated the abort The instance is specified which initiated the abort (“abort source”) from the point of view of ACSE.
408	O_REM_ACSE The originator is the ACSE service provider. &XP1INFO can take the following values: 0 (A_ABSASU) ACSE service user initiated the abort 1 (A_ABSASP) ACSE service provider initiated the abort

P016 Association disconnected (a_relin()):
&ACPNT, &OSLPAP, &XPLNK, &XPNDIA

This message is issued if an association is cleared because a “release indication” was received.

The inserts have the following meaning:

&ACPNT	Name of the local ACCESS-POINT (*)
&OSLPAP	Name of the partner in the local application (*)
&XPLNK	Represents the internal status of the association 0 = Association not linked 1 = Association linked to channel 2 = Association linked to instance
&XPNDIA	See table on page 344

P017 OSS decoding error: &XPPDU, &XP1DIA, &XP2DIA, &XP3DIA

This message is issued if OSS detects an error on decoding a TP PDU, CCR PDU or user data PDU.

The inserts have the following meaning:

XPPDU	Meaning
0	PDU_UNKNOWN
1	TP_BEGIN_DIALOGUE_RI
2	TP_BEGIN_DIALOGUE_RC
3	TP_BID_RI
4	TP_BID_RC
5	TP_EBD_DIALOGUE_RI
6	TP_END_DIALOGUE_RC
7	TP_U_ERROR_RI
8	TP_U_ERROR_RC
9	TP_ABORT_RI
10	TP_GRANT_CONTROL_RI
11	TP_REQUEST_CONTROL_RI
12	TP_HANDSHAKE_RI
13	TP_HANDSHAKE_RC
14	TP_HSK_AND_GRT_CTRL_RI
15	TP_HSK_AND_GRT_CTRL_RC

XPPDU	Meaning
16	TP_DEFER_RI
17	TP_PREPARE_RI
18	TP_HEURISTIC_REPORT_RI
19	TP_TOKEN_GIVE_RI
20	TP_TOKEN_PLEASE_RI
21	TP_RECOVER_RI
22	TP_INITIALIZE_RI
23	TP_INITIALIZE_RC
24	CCR_INITIALIZE_RI
25	CCR_INITIALIZE_RC
26	CCR_BEGIN_RI
27	CCR_BEGIN_RC
28	CCR_PREPARE_RI
29	CCR_READY_RI
30	CCR_COMMIT_RI
31	CCR_COMMIT_RC
32	CCR_ROLLBACK_RI
33	CCR_ROLLBACK_RC
34	CCR_RECOVER_RI
35	CCR_RECOVER_RC
50	PDU_ANY
51	PDU_UASE_RI

XP1DIA XP2DIA	**Meaning**
1	not supported parameter was received and skipped
2	received data truncated
4	required transfer syntax name missing in user data or not specified in AVX list, error codes in &XP2DIA
6	no transfer syntax name in user data though presentation negotiation was not completed
7	transfer syntax name encoded in user data not found in AVX list
10	invalid value in data structure
11	invalid object identifier in data structure
12	invalid length or count in data structure
13	invalid index in data structure (EXTERNAL, CHOICE)
14	invalid value of ax_typtag in corresponding syntax table

&XP3DIA	Corresponding index in the syntax table

P018 FSM protocol error: &ACPNT, &OSLPAP, &XPPTYP, &XPFSMN

This message is issued when the finite state machine reports an error.

The inserts have the following meanings:

&ACPNT	Name of the local ACCESS-POINT (*)
&OSLPAP	Name of the partner in the local application (*)
&XPPTYP	Type of the service protocol element
&XPFSMN	Name of the finite state machine

P019 APDU contains invalid value:
&ACPNT, &OSLPAP, &XPAPDU, &XP3INFO

This message is issued if an invalid APDU is received.

The inserts have the following meanings:

&ACPNT	Name of the local ACCESS-POINT (*)
&OSLPAP	Name of the partner in the local application (*)
&XPAPDU	Type of the APDU
&XP3INFO	Supplementary information on the error

P020 OTRACE implicitly switched off. Reason: &XPTRFAIL

This message is issued when an attempt to write a trace record fails. The OSS trace is deactivated implicitly as a result of the error. After the error has been corrected, the administrator can reactivate the OSS trace.

The inserts have the following meaning:

XPTRFAIL	Meaning
1	The OSS function o_wutr() issued the return code O_ERROR. The preceding P001 message provides further information on the error.
2	The OSS function o_wutr() issued the return code O_INVEREF.
3	The OSS function o_wutr() issued an unknown return code.

P021 The unexpected event &XPEVT occurred for associations. The event is ignored: &ACPNT, &OSLPAP, &XPOSAS, &XPASST

This message is output when an event occurs that does not fit with the current status of the association. XAP-TP ignores this event.

The message inserts are as follows:

&XPEVT	Type of the event that has occurred
&ACPNT	Name of the local ACCESS-POINT (*)
&OSLPAP	Name of the partner in the local application (*)
&XPOSAS	Index of the relevant association
*&XPASS*T	Status of the relevant association

P100 Instance allocation timeout:
&ACPNT,
&OSLPAP,
&XPOSAS

This message is output when the attempt to allocate an XAP-TP instance to a connection fails within a specified period.

&ACPNT	Name of the local ACCESS-POINT (*)
&OSLPAP	Name of the partner in the local application (*)
&XPOSAS	Index of the relevant association

P101 CMX Error:
&ACPNT,
&OSLPAP

This message is output when a CMX error occurs. Message P012 is output as well.

&ACPNT	Name of the local ACCESS-POINT (*)
&OSLPAP	Name of the partner in the local application (*)

One possible cause may be that the entries for PSEL and SSEL are missing from the TNSX generation.

10.2.1 General inserts for the XAP-TP messages

XPRET	Meaning
2	Not first process of application
-1	Function call not successful due to permanent error.
-2	Function call not successful due to transient error. Retry the call later
-3	Function call not successful, data flow stopped Continue after event GO
-4	Session call: Expedited function call stopped due to expedited data flow control shortage Continue after event S_XGO/S_GO Presentation call: Function call not successful, apref invalid Local function call: Invalid connection reference ACSE call: Function call not successful, apref resp. aref invalid
-5	Invalid waiting point reference
-6	Invalid application reference
-7	Waiting period to obtain a lock on a shared association expired

XPERR	Meaning
1	No memory available (temporary)
100	Call sequence error
101	Application not attached
102	Sending of data not allowed; wait for GO event
103	Internal error
104	Shared association is not locked
200	Missing ACSE/presentation reference
201	Invalid ACSE/presentation reference
202	Presentation call: missing AVX list (o_attach) ACSE call: missing application reference
203	Presentation call: invalid AVX list ACSE call: invalid application reference

XPERR	Meaning
204	Presentation call: invalid abstract syntax name in AVX ACSE call: missing ACSE parameters
205	Presentation call: invalid decoding mode in AVX ACSE call: missing presentation parameters
206	Presentation call: invalid user data length ACSE call: missing session parameters
207	Presentation call: invalid context id in p_udl ACSE call: missing application context name
208	Presentation call: invalid next parameter in p_udl ACSE call: invalid application context name
209	Presentation call: invalid pdv parameter in p_udl ACSE call: invalid calling AP Title
210	Presentation call: invalid chaining parameter ACSE call: invalid calling AE Qualifier
211	Presentation call: missing token parameter ACSE call: invalid called AP Title
212	Presentation call: invalid token parameter ACSE call: invalid called AE Qualifier
213	Presentation call: missing rtype parameter ACSE call: invalid responding AP Title
214	Presentation call: invalid rtype parameter ACSE call: invalid responding AE Qualifier
215	Presentation call: missing type parameter ACSE call: missing called p_address
216	Presentation call: invalid type parameter ACSE call: invalid called p_address
217	Presentation call: invalid syncp parameter ACSE call: missing calling p_address
218	Presentation call: missing syncp parameter ACSE call: missing responding p_address
219	Presentation call: invalid ctxlst parameter ACSE call: no mode parameter
220	Presentation call: invalid number of abstract syntaxes passed to OSS ACSE call: invalid mode parameter
221	Presentation call: invalid transfer syntax name ACSE call: missing result
222	Presentation call: invalid number of transfer syntaxes ACSE call: invalid result

XPERR	Meaning
223	Presentation call: invalid number of abstract syntaxes ACSE call: missing result source
224	Presentation call: same abstract syntax occurred already in transparent or non-transparent mode ACSE call: invalid result source
225	Presentation call: invalid data separation parameter ACSE call: invalid diagnostic
226	ACSE call: missing reason
227	ACSE call: invalid reason
228	ACSE call: missing provider reason
229	ACSE call: invalid provider reason
230	ACSE call: missing abort source
231	ACSE call: invalid p-requirements
232	ACSE call: invalid s-requirements
233	ACSE call: invalid syntax identifier
234	ACSE call: invalid p-context identifier
235	ACSE call: invalid p-context definition list
236	ACSE call: invalid p-context definition result list
237	ACSE call: invalid result in p-context definition result list
238	ACSE call: invalid default p-context result
239	ACSE call: invalid default p-context name
240	ACSE call: invalid user data length
241	ACSE call: invalid quality of service
242	ACSE call: invalid sync point serial number
243	ACSE call: invalid tokens
244	ACSE call: invalid SS-user reference
245	ACSE call: invalid SS-common reference
246	ACSE call: invalid SS-additional reference
250	Presentation call: ASN encoding error ACSE call: ASN encoding error
251	Presentation call: ASN decoding error ACSE call: ASN decoding error
252	Presentation call: ASN: invalid value in data struct ACSE call: ASN: invalid value in data struct

XPERR	Meaning
253	Presentation call: ASN: invalid object id in data struct ACSE call: ASN: invalid object id in data struct
254	Presentation call: ASN: invalid length in data struct ACSE call: ASN: invalid length in data struct
255	Presentation call: ASN: invalid index in data struct ACSE call: ASN: invalid index in data struct
256	Presentation call: ASN: invalid tag in syntax table ACSE call: ASN: invalid tag in syntax table
300	Presentation call: invalid protocol state ACSE call: invalid protocol state Local function call: error on system call
301	Presentation call: protocol error ACSE call: protocol error Local function call: error on transport system call
302	Local function call: error on local function call
305	Local function call: error on session call
306	Local function call: error on presentation call
307	Local function call: error on ACSE call

XPRJCT	Meaning
0	NO_REJECT
1	APPLICATION_CONTEXT_NAME_TOO_LONG The object identifier received from the partner, which forms the application context name, contains more elements than supported by openUTM.
2	CALLING_APT_TOO_LONG A length was specified for the application process title in the association indication which is not supported by openUTM.
3	CALLING_AEQ_TOO_LONG A length was specified for the application entity qualifier in the association indication which is not supported by openUTM.
4	CALLED_APT_TOO_LONG The application process title which was called is longer than that supported by openUTM.
5	CALLED_AEQ_TOO_LONG The application entity qualifier which was called is longer than that supported by openUTM.

XPRJCT	Meaning
6	CONTEXT_DEFINITION_LIST_TOO_LONG More abstract syntaxes were passed for the association indication than are supported by openUTM.
7	CONTEXT_RESULT_LIST_TOO_LONG The list of supported abstract syntaxes passed when establishing an association (association indication or confirmation) contains more elements than are supported by openUTM.
9	ADDRESS_NO_PSAP_INFO The address passed for association indication or confirmation does not contain any information on PSAP.
10	ADDRESS_NO_INFO_VERS_0_PSAP The address passed for association indication or confirmation contains an incorrect version for the PSAP information.
11	ADDRESS_INVALID_P_SEL_LENGTH The address passed for association indication or confirmation contains an invalid length for the presentation selector.
12	ADDRESS_NO_SSAPINFO The address passed for association indication or confirmation does not contain any information on SSAP.
13	ADDRESS_NO_INFOVERS_0_SSAP The address passed for association indication or confirmation contains an incorrect version for the SSAP information.
14	ADDRESS_INVALID_S_SEL_LENGTH The address passed for association indication or confirmation does not contain a valid part for the session selector.
15	ADDRESS_NO_PARTNER_MODE The address passed for association indication or confirmation does not contain a valid part for the network and transport selector.
16	ADDRESS_TNSX_ERROR The address passed for association indication or confirmation has been rejected by TNS.
17	UNKNOWN_PARTNER The address passed for association indication or confirmation is not known in the local application.
18	WRONG_APPLICATION_CONTEXT_NAME The application context passed for association indication or confirmation does not correspond to the application context name generated in the local application.
19	ABSTRACT_SYNTAX_MISSING The association indication or confirmation supports less abstract syntaxes than are generated in the local application.

XPRJCT	Meaning
20	OSITP_SYNTAX_MISSING The association indication or confirmation does not support the abstract syntax for OSI-TP.
21	NO_TP_INITIALIZE No TP-INITIALIZE-RI/RC PDU was passed with the association indication or confirmation.
22	OSITP_NO_VERSION_1 The partner does not support Version 1 of the OSI-TP protocol.
23	OSITP_RCH_WRONG_LENGTH The recovery context handle passed with the TP-INITIALIZE-indication or TP-INITIALIZE-confirmation is of a length not supported by openUTM.
24	NO_CCR_INITIALIZE The CCR-INITIALIZE-RI PDU is missing.
25	CCR_NOT_VERSION_2 The partner does not support Version 2 of the CCR protocol.
26	SESSION_NO_FDX Session functionality “full duplex” has not been set.
27	SESSION_NO_DATA_SEPARATION Session functionality “data separation” has not been set although CCR is in the context.
28	SESSION_NO_TYPED_DATA Session functionality “typed data” has not been set although CCR is in the context.
29	SESSION_NO_MINOR_SYNCHRONIZE Session functionality “minor synchronize” has not been set although CCR is in the context.
30	SESSION_NO_RESYNCHRONIZE Session functionality “resynchronize” has not been set although CCR is in the context.
31	TOKEN_CONTENTION_WINNER_AND_NO_TOKEN The local application is the contention winner, but does not possess the “token” (only if CCR is in the context).
32	TOKEN_CONTENTION_LOSER_AND_TOKEN The local application is the contention loser, but possesses the “token” (only if CCR is in the context).
33	INITIAL_SYNC_POINT_SERIAL_NUMBER_NOT_SET The initial syncpoint serial number is not set, although CCR is in the context.
34	NO_MORE_CONTENTION_LOSER_ASSOCIATIONS The request to establish an association from outside is rejected because all the contention loser associations have already been established in the local application.
35	NO_MORE_CONTENTION_WINNER_ASSOCIATIONS Request to establish an association from outside is rejected because all the contention winner associations have already been established in the local application.

XPRJCT	Meaning
36	CCR_BUT_NO_PARTNER_AET Partner did not specify an application entity title, although CCR is in the context.
37	CCR_BUT_NO_OWN_AET No application entity title is specified in the local application, although CCR is in the context.
38	RESPONDING_APT_TOO_LONG The application process title specified in the association confirmation is longer than that supported by openUTM.
39	RESPONDING_AEQ_TOO_LONG The application entity qualifier specified in the association confirmation is longer than that supported by openUTM.
40	ASS_ESTABLISHMENT_TIMEOUT The establishment of an association started by the local application cannot be completed in the specified time.
41	PARTNER_IS_IN_QUIET_STATE The request for establishment of an association will be rejected because the partner in the local application has been set to Quiet.
42	NO_SPACE_FOR_RCH The PutElement call for storing the recovery context handle returned a bad value.
43	REMOTE_AET_2_BIG The application entity title of the partner is longer than that supported by openUTM.
44	REMOTE_AET_CHANGED When establishing parallel associations to a partner the partner did not provide the same application entity title as for the first association established.
45	NO_SPACE_FOR_REMOTE_AET The PutElement call for storing the application entity title of the partner returned a bad value.
46	PARTNER_HAS_STATUS_OFF The establishment of the association is rejected because the partner is locked in the local UTM application (STATUS=OFF is set).
47	ADDRESS_PRES_ERROR The address delivered with the Association Indication or Confirmation couldn’t be evaluated.

XPNDIA	Meaning
0	NO_REASON_GIVEN
1	NO_COMMON_ACSE_VERSION The partner rejected the request to establish an association because there is no common ACSE version.
2	APPL_CONTXT_NAM_NOT_SUPPORTD The partner rejected the request to establish an association because it does not support the application context name.
3	CALLING_NSEL_NOT_RECON The partner rejected the request to establish an association because the sender is not generated correctly at the partner (e.g. incorrect N-SEL). or (for heterogeneous connections only): CALLING_AP_TITLE_NOT_RECON The partner rejected the request to establish an association because it does not know the calling application process title.
4	CALLING_AE_QUALI_NOT_RECON The partner rejected the request to establish an association because it does not know the calling application entity qualifier.
5	CALLING_AP_INVOC_ID_NOT_RECON The partner rejected the request to establish an association because it does not know the calling application process invocation identifier.
6	CALLING_AE_INVOC_ID_NOT_RECON The partner rejected the request to establish an association because it does not know the calling application entity invocation identifier.
7	CALLED_AP_TITLE_NOT_RECON The partner rejected the request to establish an association because it does not know the called application process title.
8	CALLED_AE_QUALI_NOT_RECON The partner rejected the request to establish an association because it does not know the called application entity qualifier.
9	CALLED_AP_INVOC_ID_NOT_RECON The partner rejected the request to establish an association because it does not know the called application process invocation identifier.
10	CALLED_AE_INVOC_ID_NOT_RECON The partner rejected the request to establish an association because it does not know the called application entity invocation identifier.
11	PERMANENT_FAILURE The partner cleared the association because a permanent error occurred.

XPNDIA	Meaning
12	BEGIN_TRANSACTION_REJECT The partner cleared the association because it rejected the start of a trans-action.
13	TRANSIENT_FAILURE The partner cleared the association because a temporary error occurred.
14	PROTOCOL_ERROR The partner cleared the association because a protocol error occurred.
15	UNRECOGNIZED_PDU The association was cleared from outside with P-ABORT because the presentation layer received an unknown presentation PDU.
16	UNEXPECTED_PDU The association was cleared from outside with P-ABORT because the presentation layer received an unexpected presentation PDU.
17	UNEXPECTED_SESSION_SERVICE_PRIMITIVE The association was cleared from outside with P-ABORT because the session layer received an unexpected session service primitive.
18	UNRECOGNIZED_PDU_PARAMETER The association was cleared from outside with P-ABORT because the presentation layer received an unknown PPDU parameter.
19	UNEXPECTED_PPDU_PARAMETER The association was cleared from outside with P-ABORT because the presentation layer received an unexpected PPDU parameter.
20	INVALID_PPDU_PARAMETER_VALUE The association was cleared from outside with P-ABORT because the presentation layer received an invalid PPDU parameter value.
21	RELEASE_NORMAL The association was cleared by the partner with release. The partner specified release normal as the reason.
22	RELEASE_URGENT The association was cleared by the partner with release. The partner specified release urgent as the reason.
23	RELEASE_USER_DEFINED The association was cleared by the partner with release. The partner specified user defined as the reason
24	IDLE_TIMEOUT_ABORT The association was cleared by the local application because the association was not used in the time generated with IDLETIME.

10.3 Messages from the utilities

10.3.1 Messages from u62_start

All messages from `u62_start, with the exception of` message 18, are output to `stdout`. Messages 17 and 18 are output to the file

`/opt/lib/utmlu62/PROT/prot.`***luname***

on UNIX systems, and to the file

`Programs\utmlu62\PROT\prot.`***luname***`.txt`

on Windows systems.

02 Usage:
u62_start [-l <LU name>] [-c|-k] [-t on[,<trace options>]]
-l LU name: Name of the local LU of the openUTM-LU62 instance
-c: cold start openUTM-LU62
-k: lukewarm start openUTM-LU62
t on: explicit specification of the trace by
additional options possible (separated by comma):
IN[=<level>]: with instance trace
XAP: with XAP-TP provider trace

`u62_start` has been called with incorrect switches or parameters. openUTM-LU62 is not started.

03 Entering the base directory &DIRNAME denied,
errno &ERRNO (&ERRTEXT)

04 The directory &DIRNAME cannot be created, errno &ERRNO (&ERRTEXT)

If the subdirectories `PROT` and `.pipes` on UNIX systems do not exist under `/opt/lib/utmlu62` (or `Programs\utmlu62`) when `u62_start` is called (e.g. during the first start after openUTM-LU62 has been installed), then `u62_start` creates these subdirectories before the `u62_tp` call. If the corresponding system call (mkdir) fails, then this message informs the user in more detail about the cause of the error:
&ERRNO = Contents of the system variable `errno`
&ERRTXT = Short explanation of *&ERRNO*

05 The directory &DIRNAME cannot be read,
errno &ERRNO (&ERRTEXT)

06 The configuration file &FILENAME cannot be opened,
errno &ERRNO (&ERRTEXT)

07 Error in reading the configuration file &FILENAME,
errno &ERRNO (&ERRTEXT)

08 The version of u62_start does not match with the configuration file &FILENAME (&VERS1 instead of &VERS2)

The *&FILENAME* configuration file created by `u62_gen` contains an incorrect version number. *&FILENAME* is either not a configuration file or an incompatible `u62_gen` version was used for the generation.

09 The local LU name &LUNAME is not configured.

10 The instance for the local LU name &LUNAME is already running.

11 Not enough memory available

12 Internal error occurred

13 System call fork() failed, errno &ERRNO (&ERRTEXT)

`u62_start` uses the system call fork() to execute in the background so that the call does not block the system. If this call fails, then `u62_start` outputs the cause of the error with this message. Generally, the maximum number of processes has been exceeded (in the system or for the user). This can be corrected by changing the system settings.

On Windows systems, the system call CreateProcess() is used instead of fork().

14 System call fork() for starting the instance with the local LU name &LUNAME failed, errno &ERRNO (&ERRTEXT)

There is a lack of system resources, just like as described for message 13.

15 System call execv() for starting the instance with the local LU name &LUNAME failed, errno &ERRNO (&ERRTEXT)

Error in the system call execv(). On Windows systems, the system call CreateProcess() is used instead of execv().

16 The protocol file &FILENAME cannot be opened,
errno &ERRNO (&ERRTEXT)

The system resources have probably all been used up (file system full, maximum number of I-nodes has been reached, ...).

17 * * *   openUTM-LU62 started: * * *
local LU alias name = &LUNAME,
PID = &INSTPID,
Start Type = c|w|k

This message is always output when an entity is started.

18 * * * End of openUTM-LU62 * * *

`u62_start` outputs a start message and an end message in the log file of a started openUTM-LU62 entity. Both messages contain the exact time. *&INSTPID* in start message 17 specifies the process ID of the `u62_tp` program that carried out the protocol conversion for the local LU *&LUNAME*.

19 The current user is not authorized to start openUTM-LU62.

Users of the UNIX system (other than root) not listed in the configuration file `u62_users` are not authorized to start openUTM-LU62 instances.

20 Variable U62_INST_DIR (installation directory) not set!

The variable U62_INST_DIR is entered in the Windows systems registry during installation, and must be set. If this entry is deleted from the registry, `u62_start` aborts immediately.

21 UTM-LU62 runs as service.

This message is only output to the file `stderr.txt`.

22 stdout cannot be redirected to &FILENAME,
errno &ERRNO (&ERRTEXT)

23 stderr cannot be redirected to &FILENAME,
errno &ERRNO (&ERRTEXT)

If openUTM-LU62 is started as a service, a connection to a terminal is not available for outputting messages. stdout and stderr are therefore rerouted to a file. If this fails, openUTM-LU62 shuts down immediately with one of the messages shown above.

24 Base thread id = %ID.

25 Service thread id = %ID.

If openUTM-LU62 is started as a service, the thread IDs of the basic and actual service threads are output to the file `stdout.txt` for information purposes only.

26 Service control dispatcher returned successfully.

This message is output together with a time stamp if the service dispatcher is successful, i.e. the system function that started the actual service is output to the file `stdout.txt`.

27 Service control dispatcher failed,
errno &ERRNO (&ERRTEXT)

This message is output together with a time stamp if the service dispatcher is not successful, i.e. the system function that started the actual service is output to the file `stderr.txt`.

28 Waiting for service thread ...

When openUTM-LU62 is terminated as a service, you must wait until the service has ended before exiting the program. To indicate the duration of this process, message 28 is repeated every 100 milliseconds.

29 Service thread has terminated

This message is output together with a time stamp to indicate when the service thread has ended.

30 Error installing the service:
Function &FUNC,
errno &ERRNO (&ERRTEXT)

31 Service &SERVICENAME installed successfully.

Messages 30 and 31 appear in response to the command `u62_start -r`.

32 Stopping service &SERVICENAME.

This message appears in response to the command `u62_start -r` if openUTM-LU62 is still running. A period is then output every second until the service has ended.

33 Service &SERVICENAME is stopped now.

34 Service &SERVICENAME cannot be stopped,
errno &ERRNO (&ERRTEXT)

This message appears in response to the command `u62_start -u` to indicate whether or not openUTM-LU62 was stopped successfully. In the event of an error, the error code &ERRNO from the system function QueryServiceStatus() is output.

35 Service &SERVICENAME removed successfully!

36 Error removing the service %s:
Function &FUNC,
errno &ERRNO (&ERRTEXT)

This message indicates whether or not openUTM-LU62 was successfully removed as a service. In the event of an error, the name of the system function that caused the error is output (OpenSCManager, OpenService or DeleteService) together with the error code &ERRNO.

37 Error indicating service handler:
Function &FUNC,
errno &ERRNO (&ERRTEXT)

If openUTM-LU62 was started as a service, the system must be informed of the function responsible for terminating the service. If this fails, error message 37 is output. The error code ERRNO of the system function FUNKNAME provides information on the precise cause of the error. openUTM-LU62 then shuts down.

38 Error setting service status:
Function &FUNC,
errno &ERRNO (&ERRTEXT)

At different points in time, the openUTM-LU62 service informs the system of the current service state. If the corresponding system function &FUNKNAME fails, message 38 is output. openUTM-LU62 then shuts down.

39 Creating new thread failed,
errno &ERRNO (&ERRTEXT)

As soon as all openUTM-LU62 entities have been started successfully, `u62_start` creates threads whose sole purpose is to wait for entities to terminate and then perform the appropriate clean-up activities. If the corresponding function call CreateThread() fails, this message is output and openUTM-LU62 terminates.

40 Usage:
In order to start openUTM-LU62 instances:
u62_start [-l <LU name>] [-c|-k] [-t on[,<trace options>]]
-l LU name: name of the local LU of the openUTM-LU62 instance
-c: cold start of openUTM-LU62
-k: lukewarm start of openUTM-LU62
-t on: explicit specification of the trace by additional options possible (separated by comma):
IN=[<Level>]: with instance trace
XAP: with XAP-TP provider trace
In order to (de)register openUTM-LU62 as service:
u62_start -r [-l <LU-Name>] to register as service
u62_start -u [-l <LU-Name>] to deregister as service

42 The service &SERVICENAME started successfully.

This message is output at startup of openUTM-LU62 on Windows systems.

43 Error at startup of the service &SERVICENAME:
Function &FUNC,
errno &ERRNO (&ERRTEXT)

If one or more openUTM-LU62 instances are started directly from the command line on Windows systems, a system service with the name &SERVICENAME is set up and started for each instance that is started. If the system service cannot be started, message 43 appears, where &FUNKNAME is the name of the failed system function (OpenSCManager, OpenService, StartService) and &ERRNO and &ERRTEXT specify the error code.

10.3.2 Messages from u62_sta

02 Usage:
u62_sta [-l <LU-Name>] [-b]
u62_sta -c [-b]

03 Error creating input pipe &PIPEFILE:
errno &ERRNO (&ERRTEXT)

04 Error opening input pipe &PIPEFILE:
errno &ERRNO (&ERRTEXT)

05 Instance &INST:
Error opening output pipe &PIPEFILE:
errno &ERRNO (&ERRTEXT)

There is probably a temporary shortage of system resources.

06 Instance &INST:
Instance is still initializing or terminating.

07 Instance &INST:
Error writing to output pipe &PIPEFILE:
errno &ERRNO (&ERRTEXT)

This message should only appear in exceptional situations, such as when there is a shortage of system resources or when a collision occurs when the openUTM-LU62 entity *&INST* is terminated.

08 Instance &INST:
Error writing to output pipe &PIPEFILE:
Number of written bytes erroneous.

09 Instance &INST:
Instance &PID (&PID) does not respond in time.

The `u62_tp` program has not responded within a specified period after `u62_sta` has sent its request to the $INST entity of openUTM-LU62. This maximum wait time is 5 seconds long. `u62_sta` then continues with the status request of the next openUTM-LU62 entity. Any acknowledgments from the openUTM-LU62 entity *&INST* that arrive too late are thrown out without the user being informed. Such drastic delays can occur when the load is high in large configurations.

10 Instance &INST:
Error reading from input pipe &PIPEFILE:
errno &ERRNO (&ERRTEXT)

11 Instance &INST:
Error reading from input pipe &PIPEFILE:
Number of read bytes erroneous.

12 No instance process active.

13 The given instance is not active.

16 Instance &INST:
Command not recognized.

Probable cause: `u62_sta` and `u62_tp` belong to different versions due to a faulty openUTM-LU62 installation.

17 Instance &INST:
Unexpected return code &RETCODE received.

Probable cause: `u62_sta` and `u62_tp` belong to different versions due to a faulty openUTM-LU62 installation.

80 Instance &INST:
Error connecting to the gateway via sockets:
errno &ERRNO (&ERRTEXT)

`u62_sta` cannot establish a socket connection to the entity &INST. The error code ERRNO originates from the system function connect().

81 Instance &INST:
Error issuing message to the gateway:
errno &ERRNO (&ERRTEXT)

82 Instance &INST:
Error issuing message to the gateway:
Number of written bytes erroneous.

The system function send() for sending requests to the entity has not returned the expected result, i.e. the message length in bytes. This error does not cause the administration program to terminate.

83 Instance &INST:
Error receiving the response from the gateway:
errno &ERRNO (&ERRTEXT)

84 Instance &INST:\
Error receiving the response from the gateway:
Number of read bytes erroneous.

The system function recv() for retrieving acknowledgments from the entity &INST has not returned the expected result, which causes the program to abort. If ??? is output for the entity, there are at least two acknowledgments from different entities.

10.3.3 Messages from u62_adm

Messages 03 through 13 and messages 16 and 17 each mean the same as the corresponding messages of the `u62_sta` program. These messages point to an error in communication between the administration program (`u62_sta` or `u62_adm`) and the `u62_tp` program.

02 Usage:
u62_adm [-l <LU name>] -ton [,<trace options]
u62_adm [-l <LU name>] -tof [,<trace options>]
u62_adm [-l <LU name>] -tfl [,<trace options>]
u62_adm [-l <LU name>] -co
u62_adm [-l <LU name>] -cs
u62_adm [-l <LU name>] -do
u62_adm [-l <LU name>] -ds
u62_adm [-l <LU name>] -ao
u62_adm [-l <LU name>] -as
u62_adm [-l <LU name>] -e
u62_adm [-l <LU name>] -v
u62_adm -f [-p] -o <output> <trace 1> [<trace 2> ...]

14 Instance &INST:
Configuration of LU6.2 software and openUTM-LU62 inconsistent.

This error message is only possible in conjunction with the switches -cs, -ds and -as, and therefore with the commands in `u62_tp` that affect the LU6.2 interface to the LU6.2 basic software (e.g. TRANSIT). In the case of TRANSIT, the message informs the user that the TRANSIT configuration does not contain the local LU of the openUTM-LU62 entity (LOC-LU-ALIAS generation parameter) or the remote LU (REM-LU-ALIAS generation parameter), or that there is no suitable PAIR statement in the TRANSIT generation so that communication between the two LUs is not possible.

15 Instance &INST: System error occurred.

An administration command could not be executed for some reason which is not described further. If the administration command affects the LU6.2 interface of openUTM-LU62 (-cs, -ds, -as), a possible cause for this error message is that the LU6.2 basic software (i.e. for TRANSIT the TRANSIT base system or the LU62Mgr) is not active. In the case of the -cs command, it is also possible that the partner has rejected the request to open a session.

20 Instance &INST terminates.

21 Instance &INST: Traces activated successfully.

22 Instance &INST: Traces already active.

23 Instance &INST: Instance trace activated successfully.

24	Instance &INST: Instance trace is already active.
25	Instance &INST: XAP-TP trace activated successfully.
26	Instance &INST: XAP-TP trace is already active.
27	Instance &INST: Traces deactivated successfully.
28	Instance &INST: Traces already inactive.
29	Instance &INST: Instance trace deactivated successfully.
30	Instance &INST: Instance trace is already inactive.
31	Instance &INST: XAP-TP trace deactivated successfully.
32	Instance &INST: XAP-TP trace is already inactive.
33	Instance &INST: Flush of traces successful.
34	Instance &INST: Flush of instance trace successful.
35	Instance &INST: Flush of XAP-TP trace successful.
36	Instance &INST: Activation of new association initiated.
37	Instance &INST: Maximum number of associations exceeded.
38	Instance &INST: Activation of new session initiated.
39	Instance &INST: Maximum number of sessions exceeded.
40	Instance &INST: Deactivation of free associations initiated.
41	Instance &INST: No free associations exist currently.
42	Instance &INST: Deactivation of all sessions initiated.
43	Instance &INST: No sessions exist currently.
44	Instance &INST: Deactivation of all associations initiated.
45	Instance &INST: No associations exist currently.
50	Error opening output file &FILENAME, errno &ERRNO (&ERRTXT)

51 Error opening message catalogue &CATNAME,
errno &ERRNO (&ERRTXT)

The message catalog *&CATNAME* is required for the evaluation of the binary trace file initiated with the `command u62_adm -f`. This catalog can be found in the directory `/opt/lib/nls/msg/En` after openUTM-LU62 has been installed successfully. If the catalog cannot be opened by `u62_adm`, then the catalog has generally been deleted manually or the access rights have been changed manually. The contents of *&ERRNO* for the variable `errno` contains more detailed information as to the cause in any case.

52 Error opening trace file &FILENAME,
errno &ERRNO (&ERRTXT)

53 Error reading from trace file &FILENAME,
errno &ERRNO (&ERRTXT)

A trace record in the binary trace file *&FILENAME* could not be read; the evaluation of the file is aborted. This message appears if *&FILENAME* is not a trace file for an openUTM-LU62 entity or if the trace file was not completely written. The file is written completely when the user enters `u62_adm -tfl [,IN]` or `u62_adm -tof [,IN]`. In addition, the trace is also written completely when openUTM-LU62 is terminated (only exception: kill -9 <PID of `u62_tp`>).

54 Not enough memory available

This error message only appears in connection with the -f switch used to evaluate trace files. The program has not received enough memory space from the operating system. This error message can appear when the trace level is greater than or equal to 4, meaning that the `u62_tp` program has also written trace entries while interrupts were being handled. When this happens just as another trace entry is being written, then the internal structure of the trace file is destroyed.

55 Internal error occurred

56 Error closing trace file &FILENAME,
errno &ERRNO (&ERRTXT)

This message only serves to inform the caller and does not have any further meaning; the evaluation has already been completed.

57 Error closing message catalogue &CATNAME,
errno &ERRNO (&ERRTXT)

This message only serves to inform the caller and does not have any other effect because the evaluation of all specified trace files has already been completed.

58 Error extracting message flow by calling &FUNC
errno &ERRNO (&ERRTEXT)

59 Extracting the protocol trace may take some time. Please wait ...

This message always appears when `u62_adm -f -p` is called to extract the protocol between openUTM-LU62 and its communication partners (UTM application and LU6.2 partner, e.g. CICS) from the trace file. The performance is not as good as it could be because it is implemented as a shell script. The caller therefore needs to be patient (do not press the DEL key!).

60 The current user is not authorized to administer openUTM-LU62.

Users of the UNIX system (other than root) not listed in the configuration file `u62_users` are not authorized to execute administration commands - with the exception of a trace evaluation.

80 Instance &INST:
Error connecting to the gateway via sockets:
errno &ERRNO (&ERRTEXT)

`u62_adm` cannot establish a socket connection to the entity &INST. The error code ERRNO originates from the system function connect().

81 Instance &INST:
Error issuing message to the gateway:
errno &ERRNO (&ERRTEXT)

82 Instance &INST:
Error issuing message to the gateway:
Number of written bytes erroneous.

The system function send() for sending requests to the entity has not returned the expected result, i.e. the message length in bytes. This error does not cause the administration program to terminate.

83 Instance &INST:
Error receiving the response from the gateway:
errno &ERRNO (&ERRTEXT)

84 Instance &INST:\
Error receiving the response from the gateway:
Number of read bytes erroneous.

The system function recv() for retrieving acknowledgments from the entity &INST has not returned the expected result, which causes the program to abort. If ??? is output for the entity, there are at least two acknowledgments from different entities.

10.3.4 Messages from u62_gen

03 Unknown text in line &LINENO - not read from &INVTXT onwards

04 No comma permitted at the beginning of the line (error in line &LINENO)

05 No comma permitted at the end of a statement (error in line &LINENO)

06 Comma is expected in line &LINENO1 or &LINENO2

07 Comma at the wrong place in line &LINENO

08 Number expected (error in line &LINENO)

09 Invalid name (error in line &LINENO)

10 INSTANCE must be the only parameter in one line.

20 Multiple parameter definition (error in line &LINENO)

21 Invalid number of elements in the array OBJECT-IDENTIFIER
(error in line &LINENO)

An object identifier consists of at least 2 and at most 10 whole numbers. If these limits are exceeded, then `u62_gen` outputs this error message.

22 Syntax error in line &LINENO

23 Remainder not read up to line &LINENO

This message is always output together with message 22 and shows the line that parser has resynchronized and in which processing will continue after a syntax error in the input file has been detected.

24 Name &NAME too long (error in line &LINENO)

25 Value &VAL not valid

26 Value for CONNECT too large (error in statement before line &LINENO)

27 Value for CONTWIN too large (error in statement before line &LINENO)

The value of CONNECT or CONTWIN exceeds the maximum number of parallel connections specified by the generation parameter ASSOCIATIONS.

28 OSITP-CODE and LU62-CODE inconsistent
(only one = *NO) (error in statement before line &LINENO)

29 Invalid TNS name &NAME (error in line &LINENO)

30 Name &NAME already allocated (error in line &LINENO)

31 The local AE Title built from APT and AEQ is already allocated
(error in statement before line &LINENO)

33 Invalid mode name &MODENAME (error in line &LINENO)

SNASVCMG is a reserved name and may not be specified as a common mode.

34 Parameter LOC-LU-ALIAS is mandatory

35 Parameter REM-LU-ALIAS is mandatory

36 Parameter MODENAME is mandatory

37 Parameter LOC-APT is mandatory

38 Parameter REM-APT is mandatory

39 Parameter LOC-AEQ is mandatory

40 Parameter REM-AEQ is mandatory

41 The configuration of the local OSI address is incomplete:
The parameter LOC-AE resp. one of the parameters
LOC-TSEL, LOC-LISTENER-PORT is missing

42 The configuration of the remote OSI address is incomplete:
The parameter REM-AE resp. one of the parameters
REM-NSEL, REM-TSEL, REM-LISTENER-PORT is missing

43 Parameter CONTWIN is mandatory

47 TNS entries and parameters for TNS-less use
must not be mixed up

48 Invalid port number
Only the ports 102 and 1025 - 65535 may be used

50 Usage:
Generation: u62_gen [-f <output file>] [<input file>]
Reverse : u62_gen -r [<conffile>]

51 File &FILENAME cannot be opened,
errno &ERRNO (&ERRTXT)

52 The version of u62_gen does not match with the file &FILENAME
(&VERS1 instead of &VERS2), or the file was not generated with u62_gen.

This message only appears when the -r switch is used. `u62_gen` has determined that the binary configuration file *&FILENAME* contains the wrong version code *&VERS1*. This must match the version *&VERS2* of `u62_gen`. A new version of openUTM-LU62 has probably been installed, and an attempt was made to convert the existing binary configuration file to a readable format using `u62_gen -r`.

53 INSTANCE statement missing

54 Application context &CONTEXT is not defined.
(error in line &LINENO)

The specified application context *&CONTEXT* does not correspond to any of the allowable values (UDTSEC, UDTAC, UDTDISAC or UDTCCR).

55 STOP (error)

`u62_gen` has detected a fatal error and has therefore aborted the processing of the input file. A configuration file was not created.

56 Not enough memory available

57 Internal error occurred

58 Error writing to file &FILENAME,
errno &ERRNO (&ERRTXT)

The most common cause is incorrect access rights for the directory or for the existing configuration file, but the cause may also be a full file system.

59 Error reading from file &FILENAME,
errno &ERRNO (&ERRTXT)

60 Generation in file &FILENAME

`u62_gen` has not detected an error in the input file and has created the binary configuration file *&FILENAME*. openUTM-LU62 can now be started.

62 The current user is not authorized to generate the configuration.

Users of the UNIX system (other than root) not listed in the configuration file `u62_users` are not authorized to (re)configure openUTM-LU62.

Glossary

addressable SNA entity (network addressable unit)

Generic term for *LU*, *PU*, *SSCP*.

Advanced Program-to-Program Communication (APPC)

Another name for the *LU6.2 protocol* and the underlying architecture. This term is also sometimes used for a program interface used to access the *LU6.2 protocol* (APPC application program interface).

agent

The communication partner that receives the *sync-point* (end of transaction) request in distributed transaction processing with *LU6.2*.

alternate facility

The name of a communication partner for a *CICS* program that has opened communication by itself using Allocate. Antonym: *principal facility*. The term job-receiver is used instead in openUTM.

Anynet

Name of a family of IBM products that implement MPTN protocols. MPTN stands for Multiprotocol Transport Networking and allows you to run applications written for a specific transport protocol on other transport protocols. An important example of “APPC over TCP/IP“, which is sometimes also referred to as “SNA over TCP/IP” or “SNA/IP”. This allows you to run two *APPC* applications over a TCP/IP network. For this purpose, an Anynet product must be installed on both end systems. On Windows systems, the IBM eNetwork Communications Server for Windows systems offers the same functionality.

association

In the OSI world, a communication relationship between two application entities. The term parallel connection is usually used instead of association in openUTM. An association corresponds to a *session* for *SNA* .

attach header

See *function management header*.

back-end transaction

In the *CICS* language, an application program that is started by another application program using *LU6.1* or *LU6.2*. The term job-receiver service is used instead in openUTM.

backout

Roll back a transaction.

basic conversation

A type of *LU6.2* conversation in which the application program must assemble the data packets itself using *GDS*. Antonym: *mapped conversation*. Basic conversations are usually not used by normal application programs, but by system programs only. The restart (*resync*) in the *LU6.2 protocol*, for example, is implemented with basic conversation.

bidder

Synonym for *contention loser*. Antonym: *first speaker*.

BIND

An *SNA* message that is sent from the *primary LU* to the *secondary LU* when the session is set up. The *secondary LU* replies with a BIND response. All properties of the *session* are specified with these two messages, including the type of the *session* (e.g. *LU6.1* or *LU6.2*), pacing, *RU sizes*, *sync level*, *mode name* and *contention winner/contention loser*, amongst others.

bracket protocol

The bracket protocol in *SNA* is a definition of how to combine several messages within the data flow of an LU-LU *session* into a processing unit. The bracket protocol supports the exchange of transaction-oriented messages with changes of direction in a job-submitter/job-receiver relationship.

chain of RUs

Specification within the *LU6.1 protocol* as to how user data is to be transferred. Chain of RUs specifies that user data is to be split into blocks with a maximum size equal to *RU size*. The first block receives the “begin of chain“ indicator, the last block the “end of chain“ indicator and all other blocks the “middle of chain“ indicator. Alternatively, the user data can also be transferred as *VLVB*. These two alternatives are not visible in the UTM application program, but they are visible in the *CICS* application program.

cluster controller

Synonym for *type 2.0 node*, although more from a hardware point of view. A typical example of a cluster controller is an IBM 3174.

Common Programming Interface for Communication (CPI-C)

A program interface defined by IBM and X/Open for *LU6.2* and OSI-TP communication.

Common Programming Interface for Resource Recovery (CPI-RR)

A program interface defined by IBM for transaction security. CPI-RR is usually used together with the CPI-C program interface. X/Open has defined the TX program interface instead of the CPI-RR.

communication controller

Synonym for *type 4 node*, although more from a hardware point of view. Corresponds approximately to a BS2000/OSD front-end processor. A typical example of a communication controller is an IBM 3745.

compare states

A protocol element used in *LU6.2* that synchronizes the common transaction when the connection is interrupted or a computer crashes.

contention winner / contention loser

One of the two communication partners of a *session* is declared the contention winner, the other is declared the contention loser. The contention winner is allowed to open *brackets* autonomously. The contention loser must obtain permission from the contention winner to open a *bracket*. The contention winner is also sometimes called the *first speaker*, and the contention loser is sometimes called the *bidder*.

conversation

A logical connection between two transaction programs via an *LU6.2 session*. A conversation begins with Allocate (corresponding to APRO in openUTM) and ends with Deallocate. A conversation allocates a *session* for its entire life and locks out all other users. The *bracket protocol* is used to reserve the *session*.

Customer Information Control System (CICS)

The IBM transaction monitor. It's functionality is comparable to that of openUTM. CICS is available for various platforms, e.g. CICS Transaction Server for z/OS (for the operation system z/OS), CICS for iSeries (for the operation system i5/OS, prior OS/400), TXSeries for Multiplatforms (for the operation systems AIX, Solaris, HP-UX, Windows).

Data Language/1 (DL/I)

Program interface defined by IBM for database accesses and data communication. DL/I is used exclusively in IMS. DL/I permits communication between IMS and openUTM.

dependent LU

See *independent LU*.

dialog

In the OSI terminology world, a communication relationship that exists between 2 services. A dialog begins with a TP-BEGIN-DIALOGUE (corresponding to APRO in openUTM) and ends with a TP-END-DIALOGUE or TP-ABORT. A dialog allocates an *association* for its entire life and locks out all other users. The term *conversation* is used instead in the *SNA* terminology world. The word “dialog” is often used in other contexts in the openUTM manuals. For this reason, this manual uses the OSI-TP dialog when in doubt.

Distributed program link (DPL)

Program communication variant that can be used by CICS programs. Communication between CICS and openUTM is not possible.

Distributed transaction processing (DTP)

Program-to-program communication variant that can be used by CICS programs. Distributed transaction processing permits communication between CICS and openUTM.

Enterprise Extender

Protocol defined by IBM that allows SNA protocols to be transported in IP networks. Enterprise Extender is regarded by IBM as, among other things, a successor to *Anynet*. Enterprise Extender does not use TCP. Instead, it uses the connectionless User Datagram Protocol (UDP). The products that support Enterprise Extender include IBM Communications Server (for AIX, Linux and Windows), SNAP-IX and IBM Communications Server for z/OS. TRANSIT does not support Enterprise Extender.

first speaker session

Synonym for *contention-winner*. Antonym: *bidder*.

front-end transaction

In the *CICS* language, an application program that calls another application program using *LU6.1* or *LU6.2*. The term job-submitter service is used instead in openUTM.

function management header (FMH)

The function management headers contain information used for communication and are sent as a header before the net data in a *request unit* (RU). The following FMHs exist for *LU6.1*:
* FMH4: FMH4 can be sent at the beginning of a message and contains additional information for the following message. openUTM sends an FMH4 if KCDF was specified as anything other than 0 or KCMF was specified as anything other than a space character in MPUT.
* FMH5 (*attach header*): FMH5 is sent at the beginning of a *bracket* and begins a job-submitter/job-receiver relationship. It contains the transaction code and an indicator for the data format (*VLVB* or *chain of RUs*). It can also be placed at the beginning of a response message, and it only contains the indicator for the data format in this case.
* FMH6 (scheduler FMH): FMH6 is sent when a queued job is transmitted.
* FMH7 (error description): FMH7 sends the partner an error message.

The following FMHs exist for *LU6.2*:
* FMH5 (*attach header*): FMH5 is sent at the beginning of a *conversation* and begins a job-submitter/job-receiver relationship. It contains the transaction code, user ID, password, already verified indicator, indicators for *basic/mapped conversation*, *sync-level* and *LUWID,* amongst others.
* FMH7 (error description): FMH7 sends the partner an error message.

functional unit commit

A function group in the OSI-TP protocol that is required to create distributed transactions. Whether or not the functional unit commit may be used is negotiated when an *association* is set up between the two partners. OSI-TP *dialogs* can run with or without the functional unit commit in an association in which the functional unit commit was agreed to. Whether or not the functional unit commit is used by an OSI-TP dialog is decided upon by the UTM program by the KCOF parameter in the APRO call or by the LU6.2 partner program. In *SNA,* a dialog with functional unit commit corresponds to a *conversation* with *sync-level* 2.

General Data Stream (GDS)

A specification within the *LU6.2 protocol* as to how user data is to be transferred. GDS specifies that user data is to be split into blocks with a maximum size of 32763 bytes, with each block containing a 2 byte long length field and a 2 byte field used to describe the data type.

half-session

In *LU6.1*, each *LU* administers one half of a *session*, the half-session. The half-session is identified by a name up to 8 characters long, the half-session qualifier. This name is specified in the generation of the corresponding applications and is specified in LSES and RSES in openUTM. A *session* is uniquely identified by the two half-session qualifiers. There are no half-session qualifiers in *LU6.2*.

host

A synonym for *type 5 node* in the *SNA* language.

independent LU / dependent LU

A *logical unit* (LU) in a *type 2.0 node* is always a dependent LU. It depends on the *SSCP* (System Service Control Point) in the associated *type 5 node*. A dependent LU can set up at most one *session* to an LU in the associated *type 5 node*. It is always the *secondary LU* in this *session*. There are dependent LUs for all LU types, and therefore also for *LU6.1* and *LU6.2,* amongst others. In contrast, a *type 2.1 node* can contain dependent and independent LUs. An independent LU can only utilize the *LU6.2 protocol*. However, it can maintain *sessions* with several partners at the same time and can maintain up to 254 *parallel sessions* with one partner.

Information Management System (IMS)

The IBM transaction monitor and database system. IMS only runs on the IBM mainframe operating system z/OS. IMS consists of the following two central components:

- IMS Database Management System (IMS DB)
- IMS Transaction Manager (IMS TM)

IMS TM is a comparable transaction monitor to openUTM.

initiator / sync point initiator

The communication partner that has prompted the other partner to execute a s*ync-point* (end of transaction) in distributed transaction processing with *LU6.2*.

cold start

A type of start for transaction-oriented applications in which all *log records* are deleted, meaning that the memory of any incomplete transactions is lost.

logical unit (LU)

A logical unit is understood to be an addressable communication partner in an *SNA* network. An LU administers one or more communication relationships (*sessions*) to other LUs (terminals or openUTM, *CICS* and *IMS* applications). Whether only one or more communication relationships are possible depends on the type of *physical unit* and on the type of LU (*independent/dependent LU*). An LU in a *type 2.0 node* can only set up a *session* to another LU, and only using the same *SNA* protocol. Depending on which protocol is being used (e.g. LU2, *LU6.1* or *LU6.2*), the LUs are also designated as LUs of type LU2, *LU6.1* or *LU6.2*. Every LU is known in the *SNA* network under a unique name with a maximum of 8 characters, the LU name. An 8-character network identification is placed in front of the 8-character LU name in an *independent LU6.2*. Together the two build the so-called fully qualified network name of the LU. Addressing is not done using the LU name in *dependent LUs* in *type 2.0 nodes*, rather using a so-called loc-address (a number between 1 and 255). An LU corresponds approximately to a BCAMAPPL or a ACCESS-POINT in openUTM.

LU6.1 protocol

The LU6.1 protocol is a component of the IBM *SNA* network architecture and defines methods for distributed transaction processing. LU6.1 was developed by IBM for distributed processing between *CICS* and *IMS* applications on *host systems*. LU6.1 can also be used for distributed processing between openUTM and *CICS* and between openUTM and *IMS* because it is also implemented in openUTM. IBM views *LU6.2* as the successor to LU6.1.

LU6.2 protocol

The LU6.2 protocol is a component of the IBM network architecture and is the newest and most modern IBM network protocol. LU6.2 defines methods for program-program communication between applications on different computers. LU6.2 can be used with and without transaction security. It provides many different security mechanisms.

logical unit of work (LUW)

This term is used in the IBM literature instead of the term transaction usually used in openUTM. A logical unit of work can be local to a computer or distributed amongst several computers.

logical unit of work identifier (LUWID)

In *LU6.2*, every distributed transaction is assigned a network-wide and long-term unique name, the LUWID. The *initiator* of the transaction assigns this name. All of the systems participating in the transaction can negotiate the outcome of the transaction when a connection is lost or a computer crashes using this name.

log name

The name used in the *LU6.2 protocol* that characterizes an application (e.g. a UTM or *CICS* application) for its entire life, i.e. between two *cold starts*. The log names must be exchanged in accordance with *LU6.2* between the two *resync TPs* before the restart (*compare states*) is initiated after a connection has been interrupted or the system has crashed. The exchange of the log names serves to identify a one-sided *cold start*.

log record

Information on the status of an open transaction that is written to a secure storage media (generally disk storage). The log records are used to restart the transaction after a computer crash. The log records are deleted after the transaction has terminated.

mapped conversation

A type of *LU6.2* conversation in which the application program does not have to take care of assembling the data into packets using *GDS*. Antonym: *basic conversation*. Normal application programs usually use mapped conversation.

mode

Describes the properties of sessions such as the maximum number of parallel sessions, RU sizes and pacing values. A mode has a *mode name*.

mode name

A name up to 8 characters long that is defined when the session is set up. A mode name is a symbolic name for a list of *session* properties, such as the maximum number of *parallel sessions*, *RU sizes* and pacing values, for example. The properties belonging to a mode name must usually be specified in at least one system per generation. The mode name is only generated on the *host* for *LU6.1*. The mode name must be generated in all participating systems for *LU6.2* and *independent LUs*.

Network Control Program (NCP)

The operating system of the *communication controller*.

network name

See *logical unit*.

pacing

A mechanism to control the flow of data in *SNA*. The receiver only gives the sender permission to send a certain number of *RUs* when pacing is used.

parallel session

When there is more than one *session* between two *LUs*, then these sessions are designated as parallel sessions. Parallel sessions are only possible between two *type 5 nodes* in *LU6.1* and only between two *type 2.1 nodes* in *LU6.2*.

physical unit (PU)

Every node in a *SNA* network contains a physical unit (PU) as an *addressable SNA entity*. Before two *logical units* (LUs) can open a communication relationship in the *SNA* network, a communication relationship must first be opened between the corresponding PUs.
There are various types of PUs in *SNA* networks. The associated systems are correspondingly designated as *type 5 nodes*, *type 4 nodes*, *type 2.0 nodes* or *type 2.1 nodes*.

primary logical unit (PLU) / secondary logical unit (SLU)

The *logical units* (LUs) can be separated into primary logical units (PLU) and secondary logical units (SLU). The PLU or SLU status of two LUs is specified during the SNA session setup (*BIND*). A preset value for the primary/secondary property is specified with the help of generation parameters. This is done in openUTM using the SESCHA generation parameter in *LU6.1* connections. A preset value is usually not specified in the generation for *LU6.2* and *independent LUs*. The primary logical unit sends proposals for the session parameters when the session is opened. The secondary logical unit accepts these proposals or sends back its own proposals (*BIND* response).

principal facility

Designation for the standard communication partner of a *CICS* program, i.e. the partner that the program has called. Antonym: *alternate facility*. The term job-submitter is used instead in openUTM.

RACF (Resource Access Control Facility)

RACF is an IBM product that offers access protection to all applications and resources in a mainframe environment. RACF is thus responsible for:

- Identification and verification of users by checking user IDs and passwords (authentication)
- Protection of resources through the administration of access rights (authorization)
- Logging of accesses to protected resources (auditing).

request unit (RU)

Net data portion of an *SNA* protocol element. See also *RU size*.

RU size

Maximum length of a *request unit*. These lengths must usually be set on both systems. The value is then negotiated during the session setup.

resync (resynchronization)

A procedure specified in the *LU6.1* and *LU6.2* protocol definition that forces open distributed transactions into a consistent state after a connection has been interrupted or a computer has crashed.

Resync service Transaction Program (Resync TP)

A part of the *LU6.2 protocol* that is required for *conversations* with *sync-level* 2. The LU6.2 protocol specifies that both resync TPs of the corresponding LUs must exchange their *log names* before a conversation with sync-level 2 is set up. The resync TP is also required to resynchronize transactions aborted when the connection has been interrupted or the system has crashed. The resync TP has the TP name X’06F2’ specified by LU6.2.

secondary logical unit (SLU)

See *primary logical unit*.

session

A session is understood to be a communication relationship between two *LUs*, more generally between two *addressable SNA entities*. Only one session can be set up between an LU on a system with *PU* type 2 and an LU on a *host system* (PU type 5). This is also called a dependent session because the *host* is the dominant partner in the session. Several sessions (*parallel sessions*) can be set up simultaneously between two LUs on host systems or type 2.1 *PUs*. These sessions are also called independent sessions.
Note that the term session has a different meaning in the OSI protocol world than the meaning described here.

side information

Side information describes the configuration required for the execution of CPI-C programs at the CPI-C program interface. In openUTM the side information is described by means of the control statements for KDCDEF. For CPI-C programs that run under IMS, the system administrator must make available a separate side information file.

symbolic destination name

For the purposes of the CPI-C program interface, a symbolic destination name is a symbolic name that allows you to address a remote transaction from within the program. The side information is used to assign this symbolic name a concrete transaction code in a concrete remote application. The concept is comparable to the LTAC used by openUTM.

sync-level (synchronization level)

Designation in *LU6.2* that characterizes the transaction security for distributed processing:
For sync-level 0 (none), only net data and error messages may be sent. Acknowledgments are not allowed. This corresponds to KCOF=B for APRO in openUTM.
For sync-level 1 (confirm), simple acknowledgments may also be sent in addition to net data and error messages. This corresponds to KCOF=H for APRO in openUTM.
For sync-level 2 (syncpoint), full transaction security is activated for distributed transactions. This corresponds to KCOF=C for APRO in openUTM.

sync point (synchronization point)

A location within the flow of a distributed process at which the common resources are brought to a defined state. The term “end of transaction” or “synchronization point” is used instead in openUTM.

Systems Network Architecture (SNA)

SNA is the designation for a series of communication protocols defined by IBM.

System Services Control Point (SSCP)

Addressable SNA entities in a *type 5 node*. The SSCP administers the connections to all *type 4* and *type 2.0 nodes* in its network and all *sessions* between the *type 5 node* and its lower-ranking node.

transaction program (TP)

In the IBM language, every program that carries out program-program communication to other programs in the network is designated as a transaction program, regardless of whether real distributed transaction processing occurs or not. If the program-program communication is done via *LU6.2*, then every transaction program is identified by a TP name. This TP name corresponds to the transaction code from openUTM or *CICS*.

transmission header (TH)

The SNA layer 3 protocol header. openUTM uses the FID1 format. The transmission header is 10 bytes long in this format. The FID2 format is used for the transmission header between TRANSIT and the IBM partner system. The transmission header is 6 bytes long in this case.

type 2.0 node

Designation for a node in the *SNA* network that can only set up *sessions* to its higher-ranking *type 4* or *type 5 node*. An *LU* in a type 2.0 node is always a *dependent LU*.

type 2.1 node

Designation for a node in the *SNA* network that contains the functions of a *type 2.0 node* on the one hand, but that can also open *sessions* to other type 2.1 nodes. Only *independent LUs* and the *LU6.2* protocol can be used in the *sessions* to other type 2.1 nodes. The TRANSIT-SERVER product emulates a type 2.1 node.

type 4 node

Designation for a node that assumes the routing functions in a conventional *SNA* network. It maintains connections to one or more *type 5 node*s and one or more *type 2.0/2.1 nodes*.

type 5 node

Designation for a node in the *SNA* network that assumes *host* functions, i.e. that contains an *SSCP*. A type 5 node can also simultaneously cover the functionality of a *type 2.1 node*. The prior TRANSIT-CD product emulates a *type 4* and a *type 5 node*.

UDP (User Datagram Protocol)

UDP is a transport protocol that supports connectionless data exchange between computers. UDP was defined in order to allow application processes to send datagrams directly and thus meet the requirements of transaction-oriented traffic. UDP builds directly on the underlying IP protocol. UDP has a minimal protocol mechanism and neither guarantees the delivery of a datagram to the destination partner nor includes any measures to prevent duplication or changes in sequence.

variable length variable blocked (VLVB)

A specification in the *LU6.1 protocol* as to how user data is to be transferred. VLVB specifies that user data is to be split into blocks with a maximum size of 32765 bytes with each block containing a 2 byte long length field in front. The user data can also be transferred as a *chain of RUs* as an alternative. These two alternatives are not visible in the UTM application program, but they are visible in the *CICS* application program.

virtual telecommunications access method (VTAM)

The component in an IBM *host system* that is responsible for remote data processing. The term ACF/VTAM is also used sometimes instead.

Abbreviations

AAID	Atomic Action IDentifier
ACF	Advanced Communications Function
ACSE	Association Control Service Element
AE	Application Entity
AEQ	Application Entity Qualifier
APDU	Application Protocol Data Unit
APPC	Advanced Program to Program Communication
APT	Application Process Title
ASCII	American Standard Code for Information Interchange
BB	Begin Bracket
BCAM	Basic Communication Access Method
CCR	Commitment, Concurrency and Recovery
CD	Change Direction
CEB	Conditional End Bracket
CICS	Customer Information Control System
CMX	Communication Manager in UNIX
CPI-C	Common Programming Interface for Communication
CPI-RR	Common Programming Interface for Resource Recovery
DPL	Distributed Program Link
DPN	Destination Process Name
DRI	Define Response Indicator
DTP	Distributed Transaction Processing
EB	End Bracket
EBCDIC	Extended Binary-Coded Decimal Interchange Code
ECI	External Call Interface
EDI	Enciphered Data Indicator
EIB	Exec Interface Block

ERI	Exception Response Indicator
ESA	Enterprise System Architecture
ET	End of Transaction
FDDI	Fiber Distributed Data Interface
FID	Format Identifier
FMH	Function Management Header
FSM	Finite State Machine
GDS	General Data Stream
IMS	Information Management System
ISC	InterSystem Communication
ISO	international organization for standardization
KDCS	Compatible data communication interface
LU	Logical Unit
LUW	Logical Unit of Work
LUWID	Logical Unit of Word Identifier
MFS	Message Format Service
MPTN	Multiprotocol Transport Networking
MVS	Multiple Virtual Storage System
NCP	Network Control Program
OSI	Open Systems Interconnection
OSI-TP	Open Systems Interconnection - Distributed Transaction Processing
OSS	Open Systems Interconnection Services
PCMX	Portable Communication Manager in UNIX
PDI	Padded Data Indicator
PDN	Program system for remote data processing and network control
PET	Preliminary End of Transaction
PID	Process Identifier
PIP	Program Initialization Parameter
PLU	Primary Logical Unit
PRN	Primary Resource Name
QRI	Queued Response Indicator
RACF	Resource Access Control Facility
RDO	Resource Definition Online

RDPN	Return Destination Process Name
RPRN	Return Primary Resource Name
RLU	Remote Logical Unit
RTI	Response Type Indicator
RTR	Ready To Receive
RU	Request Unit
SDI	Sense Data Indicator
SDLC	Synchronous Data Link Control
SSCP	System Services Control Point
SIT	System Initialization Table
SLU	Secondary Logical Unit
SNA	System Network Architecture
SNI	SNA Network Interconnection
TNSX	Transport Name Service UNIX
TP	Transaction Processing, Transaction Program
TPSU	Transaction Processing Service User
TX	Transaction Demarcation (X/Open)
UDT	Unstructured Data Transfer
UTM	Universal Transaction Monitor
VLVB	Variable Length Variable Blocked
VM	Virtual Machine
VTAM	Virtual Telecommunications Access Method
XAP	X/Open ACSE/Presentation Services
XAP-TP	X/Open ACSE/Presentation Services - Transaction Processing
XLN	Exchange Log Name

Related publications

i openUTM is accompanied by a CD-ROM. This contains PDF files of all the openUTM manuals.
The manuals are available as online manuals, see *http://manuals.ts.fujitsu.com*, or in printed form which must be paid and ordered separately at *http://manualshop.ts.fujitsu.com*.

openUTM documentation

openUTM
Concepts and Functions
User Guide

Target group
Anyone who wants information about the functionality and performance capability of openUTM.

openUTM
Programming Applications with KDCS for COBOL, C and C++
Core Manual

Target group
Programmers who wish to use the KDCS program interface for programming UTM applications.

openUTM
Generating Applications
User Guide

Target group
Application planners, application developers and UTM application support staff.

openUTM
Using openUTM Applications under BS2000/OSD
User Guide

Target group
Application planners, application developers, users and UTM application support staff.

openUTM
Using openUTM Applications under UNIX Systems and Windows Systems
User Guide

Target group
Application planners, application developers, users and UTM application support staff.

openUTM
Administering Applications
User Guide

Target group
Administrators and programmers of administration programs.

openUTM
Messages, Debugging and Diagnostics in BS2000/OSD
User Guide

Target group
Users, administrators and programmers of UTM applications.

openUTM
Messages, Debugging and Diagnostics in UNIX Systems and Windows Systems
User Guide

Target group
Users, administrators and programmers of UTM applications.

openUTM (BS2000/OSD, UNIX systems, Windows NT)
Creating Applications with X/Open Interfaces
Core Manual

Target group
Programmers and generators of UTM applications which make use of the X/Open interfaces CPI-C, XATMI and TX.

openUTM
openUTM Data Marshalling with XML
User Guide
(Available in electronic form only: internet)

Target group
Programmers of UTM applications.

openUTM Client (UNIX systems)
for the OpenCPIC Carrier System
Client-Server Communication with openUTM
User Guide

Target group
Organizers, application planners and programmers of CPI-C and XATMI programs.

openUTM Client
for the UPIC Carrier System
Client-Server Communication with openUTM
User Guide

Target group
Organizers, application planners and programmers of CPI-C and XATMI programs.

openUTM WinAdmin
Graphical Administration Workstation for openUTM
Online description (available on the internet) and online help system

Target group
Organizers and administrators of UTM applications.

openUTM (BS2000/OSD)
Programming Applications with KDCS for Assembler
Supplement to Core Manual
(Available in electronic form only: internet)

Target group
Programmers of UTM Assembler applications.

openUTM (BS2000/OSD)
Programming Applications with KDCS for Fortran
Supplement to Core Manual
(Available in electronic form only: internet)

Target group
Programmers of UTM Fortran applications.

openUTM (BS2000/OSD)
Programming Applications with KDCS for Pascal-XT
Supplement to Core Manual
(Available in electronic form only: internet)

Target group
Programmers of UTM Pascal-XT applications.

openUTM (BS2000/OSD)
Programming Applications with KDCS for PL/I
Supplement to Core Manual
(Available in electronic form only: internet)

Target group
Programmers of UTM PL/I applications.

WS4UTM (UNIX systems and Windows systems)
WebServices for openUTM
(Available in electronic form only: internet)

Target group
Application planners, application developers and UTM application support staff.

openUTM
Master Index
(Available in electronic form only: internet)

Target group
Anyone who wishes to work with the openUTM documentation.

Documentation for the openSEAS product environment

BeanConnect for openUTM
User Guide

Target group
BeanConnect administrators and designers, administrators of application servers, e.g. OracleAS/OC4J, Deployer, EJB developer and openUTM administrators.

BeanConnect for CICS
User Guide

Target group
BeanConnect administrators and designers, administrators of application servers, e.g. OracleAS/OC4J, Deployer, EJB developer and CICS administrators.

Biz*Transactions*
Application Integration with Business Objects
Online description (available in the Internet)

Target group
Anyone wishing to integrate openUTM, MVS, UTM applications into Windows, Java and openSEAS via business objects.

JConnect V2.0
Connecting Java Clients to openUTM
User documentation and Java docs
(Available in electronic form only: internet)

Target group
Programmers of Java applications.

Web*Transactions*
Concepts and Functions
Online description (available in the Internet)

Target group
Anyone wishing to connect UTM, MVS or OSD applications to the web.

Web*Transactions*
Template Language
Online description (available in the Internet)

Target group
Anyone wishing to connect UTM, MVS or OSD applications to the web.

Web*Transactions*
Web Access to openUTM Applications via UPIC
Online description (available in the Internet)

Target group
Anyone who wishes to use Web*Transactions* to connect UTM applications to the web via UPIC.

Web*Transactions*
Web Access to MVS Applications
Online description (available in the Internet)

Target group
Anyone who wishes to use Web*Transactions* to connect MVS applications to the web.

Web*Transactions*
Web Access to OSD Applications
Online description (available in the Internet)

Target group
Anyone who wishes to use Web*Transactions* to connect OSD applications to the web.

Documentation for the BS2000/OSD environment

AID (BS2000/OSD)
Advanced Interactive Debugger
Core Manual
User Guide

Target group
Programmers in BS2000/OSD.

BCAM (BS2000/OSD)
BCAM Volume 1/2
User Guide

Target group
Network planners, generators and administrators who run BCAM in BS2000/OSD.

BINDER (BS2000/OSD)
User Guide

Target group
Software developers.

BINDER (BS2000/OSD)
Ready Reference

Target group
Software developers with experience in the use of BINDER.

BS2000/OSD
Executive Macros
User Guide

Target group
BS2000/OSD assembly language programmers.

BS2000/OSD-BC
BLSSERV
Dynamic Binder Loader / Starter
User Guide

Target group
Software developers and experienced BS2000/OSD users.

DCAM (BS2000/OSD, TRANSDATA)
COBOL Calls
User Guide

Target group
Programmers of DCAM COBOL programs.

DCAM (BS2000/OSD, TRANSDATA)
Macros
User Guide

Target group
Programmers of DCAM ASSEMBLER programs.

DCAM (BS2000/OSD, TRANSDATA)
Program Interfaces
Description

Target group
Organizers, application planners, programmers, system and network administrators.

FHS (BS2000/OSD, TRANSDATA)
Format Handling System for openUTM, TIAM, DCAM
User Guide

Target group
Programmers.

IFG for FHS (TRANSDATA)
User Guide

Target group
Terminal users, application engineers and programmers.

FHS-DOORS (BS2000/OSD,MS-Windows)
Graphical Interface for BS2000/OSD Applications
User Guide

Target group
End users of FHS-DOORS and administrators responsible for mask conversion and provision.

HIPLEX AF (BS2000/OSD)
High-Availability of Applications in BS2000/OSD
Product Manual

Target group
System administrators and organizers in computer centers.

IMON (BS2000/OSD)
Installation Monitor
User Guide

Target group
Systems support staff of the BS2000/OSD operating system.

MT9750 (MS Windows)
9750 Emulation under Windows
Product Manual

Target group
PC users who want to use the product MT9750 to communicate with applications on a BS2000 host.

OMNIS/OMNIS-MENU (TRANSDATA, BS2000/OSD)
Functions and Commands
User Guide

Target group
OMNIS administrators and OMNIS users.

OMNIS/OMNIS-MENU (TRANSDATA, BS2000)
Administration and Programming
User Guide

Target group
OMNIS administrators and programmers.

OMNIS-MENU (TRANSDATA, BS2000/OSD)
User Guide

Target group
OMNIS-MENU users and OMNIS-MENU administrators.

OSS (BS2000/OSD)
OSI Session Service
User Guide

Target group
OSI TP users.

RSO (BS2000/OSD)
Remote SPOOL Output
User Guide

Target group
Nonprivileged users, RSO device administrators, SPOOL administrators and systems support of BS2000/OSD.

SECOS (BS2000/OSD)
Security Control System
User Guide

Target group
BS2000/OSD system administrators and BS2000/OSD users working with extended access protection for files.

SECOS (BS2000/OSD)
Security Control System
Ready Reference

Target group
Experienced SECOS users.

openSM2 (BS2000/OSD)
Software Monitor
Volume 1: Administration and Operation

Target group
Users and systems support staff.

TIAM (BS2000/OSD, TRANSDATA)
User Guide

Target group
BS2000/OSD users (non privileged) and programmers.

Unicode in BS2000/OSD
Introduction

Target group
Application programmers and system administrators, who want to get an overview, to which extent the Unicode support is provided in BS2000/OSD, and which BS2000/OSD components you need for the Unicode support.

VTSU (BS2000/OSD, TRANSDATA)
Virtual Terminal Support
User Guide

Target group
Users of the DCAM and TIAM access methods and of UTM, and also system and network administrators.

XHCS (BS2000/OSD)
8-Bit Code and Unicode Support in BS2000/OSD
User Guide

Target group
Users of the DCAM and TIAM access methods and of UTM, and also system administrators and users migrating from EHCS to XHCS.

Documentation for the UNIX system environment

CMX V6.0 (Solaris)
Operation and Administration
User Guide

Target group
System administrators for Solaris.

CMX V6.0 (UNIX systems)
Operation and Administration
User Guide

Target group
System administrators for UNIX systems.

CMX V6.0
Programming CMX Applications
Programming Guide

Target group
Programmers of communication applications on UNIX systems.

OSS (SINIX)
OSI Session Service
User Guide

Target group
OSI TP users

PRIMECLUSTER^{TM}
Concepts Guide (Solaris, Linux)

Target group
Users, system administrators, and support personnel.

Documentation for TRANSIT

The manuals are available as online manuals
http://manuals.fujitsu-siemens.com/softbooks/software/de/prevtsit.htm.

TRANSIT SERVER
Administration of TRANSIT
Core Manual Volume 1 and 2.

Target group
For openUTM-LU6.2 interconnection and openUTM-LU6.1 interconnection, System administrators, which are responsible for the usage of TRANSIT under UNIX systems.

TRANSIT CLIENT
Programming interface LU0, LU0-API, FTX-LU, UTMGW1
Programming Guide

Target group
For openUTM-LU6.1 interconnection, application programmer and TRANSIT adminstrators.

Documentation for CICS and IMS

This list of manuals provides just a general overview of the IBM publications available for CICS and IMS. The titles may vary slightly for the different CICS Transaction Server for z/OS V3.1 and IMS Version 9.

CICS
Intercommunication Guide
This document describes the different ways in CICS of communicating with other applications. Among other things, it describes the fundamentals of distributed transaction processing with regard to installation, configuration and programming.

CICS
Distributed Transaction Programming Guide
Description programming when Distributed Transaction Programming is used

CICS
Application Programming Guide
This document describes the fundamentals of CPI-C application programming, including distributed transaction processing.

CICS
Application Programming Reference
Description of the CICS program interface arranged in the alphabetical order of the commands

CICS
Supplied Transactions
Description of the CICS administration commands

CICS
Resource Definition Guide
Description of the CICS definitions, e.g. Connection and Session definition

IMS Version 9
Application Programming: Design Guide
Chapter 7 describes the design of LU6.2 application programs.

IMS Version 9
Application Programming: Transaction Manager
This document contains detailed information on application programming with DL/I.

IMS Version 9
Installation Volume 2: System Definition and Tailoring
This document describes IMS system definition (e.g. the macros for defining transaction codes).

IMS Version 9
Administration Guide: Transaction Manager
Chapters 19 and 20 describe how to set up LU6.2 interconnections.

IMS Version 9
Command Reference
Description of IMS administration commands

Documentation for SNAP-IX and IBM Communications Server

This list of manuals provides just a general overview of the Data Connection Limited documentation available for SNAP-IX and IBM documentation available for IBM Communications Server products. The list of documentation is related to the following product versions:

– SNAP-IX Version 7
– IBM Communications Server for Windows Version 6.1.2
– IBM Communications Server for Linux Version 6.2.2
– IBM Communications Server for AIX Version 6.3.

SNAP-IX
General Information
This document contains general information on SNAP-IX.

SNAP-IX
Installation
This document contains instructions for installing SNAP-IX on Solaris systems.

SNAP-IX
Administration Guide
This document describes the administration of SNAP-IX. It is subdivided on the basis of the different kinds of tasks involved.

SNAP-IX
Administration Command Reference
This document describes the administration commands of SNAP-IX. The commands are arranged alphabetically. It also describes the generation parameters.

SNAP-IX
Diagnostic Guide
This document describes the diagnostic options available in SNAP-IX.

IBM Communications Server for Windows
Quick Beginnings
This document contains an introduction to the product and installation instructions.

IBM Communcations Server for Windows
Network Administration Guide
This document describes network administration. It is subdivided on the basis of the different kinds of tasks involved.

IBM Communications Server for Windows
Configuration File Reference
This document describes the generation parameters.

IBM Communications Server for Linux
Quick Beginnings
This document contains an introduction to the product and installation instructions.

IBM Communications Server for Linux
Administration Guide
This document describes administration. It is subdivided on the basis of the different kinds of tasks involved.

IBM Communications Server for Linux
Administration Command Reference
This document describes the administration commands. The commands are arranged alphabetically. It also describes the generation parameters.

IBM Communications Server for Linux
Diagnostics Guide
This document describes the diagnostic options that are available.

IBM Communications Server for AIX
Quick Beginnings
This document contains an introduction to the product and installation instructions.

IBM Communications Server for AIX
Administration Guide
This document describes administration. It is subdivided on the basis of the different kinds of tasks involved.

IBM Communications Server for AIX
Administration Command Reference
This document describes the administration commands. The commands are arranged alphabetically. It also describes the generation parameters.

IBM Communications Server for AIX
Diagnostics Guide
This document describes the diagnostic options that are available.

Other publications

CPI-C (X/Open)
Distributed Transaction Processing
X/Open CAE Specification, Version 2
ISBN 1 85912 135 7

Reference Model Version 2 (X/Open)
Distributed Transaction Processing
X/Open Guide
ISBN 1 85912 019 9

TX (Transaction Demarcation) (X/Open)
Distributed Transaction Processing
X/Open CAE Specification
ISBN 1 85912 094 6

XTAMI (X/Open)
Distributed Transaction Processing
X/Open CAE Specification
ISBN 1 85912 130 6

XML
W3C specification (www consortium)
Web page: http://www.w3.org/XML

Index

A
ACCESS-POINT statement 49
activating
  trace 64
administration
  openUTM-LU62 54
  openUTM-LU62 under Windows systems 54
agent 361
AIX systems
  IBM CS gateway for LU6.2
    interconnection 31
  LU6.2 interconnection via IBM CS 27
alias name 47, 50, 55
ALLOCATE
  IMS command 213
ALLOC-TIME 45
ALREADY_VERIFIED 187
alternate facility 361
analyzing traces 66
Anynet 28, 361
APPC 37, 361
APPL definition
  IMS example for LU6.2 190
application context 45
application entity 36, 49
application entity qualifier 46, 49
application process title 47, 50
APPLID
  IMS 182
APRO see KDCS call
ASCII 48, 49, 240
association 361
asynchronous processing 110, 224
ATTACH header 227
attach header 234, 361

B
back-end transaction 362
backout 362
basic conversation 122, 362
BCAM generation 94
BCIN 94
bidder 362
BIND 362
bracket protocol 362
brackets 21
busy parallel connections 61
busy session 61

C
CEMT SET CONNECTION NOTPENDING 40
Chain of RUs 227, 231, 362
change direction 113, 225
CHNG call see IMS calls
CICS 363
CICS client 148
CICS commands 238
CICS definitions 77
  for LU6.1 connection 215
  for LU6.1 transactions 222
  for LU6.2 connection 86
  for LU6.2 transactions 84
CICS macros 215
CICS programming
  commands for LU6.1 226
  commands for LU6.2 110
  commands see EXEC CICS
  dummy dialog 256, 262
  example for LU6.2 86, 125
  job recipient for LU6.2 118
  job submitter for LU6.2 111

- job-receiver for LU6.1 233
- job-submitter for LU6.1 226
- LU6.1 example 237, 239
- rules for LU6.1 224
- rules for LU6.2 122
- see also KDCS 123

clear connection 63
cluster controller 362
cold start 40, 56, 366
- for CICS 40

commit 365
Committed 125
communication controller 363
compare states 363
comparing KDCS calls 236, 251
conffile 52
configuration file 42, 52
connection loss 40
CONNECTION name 78, 80, 216, 217
contention loser 363
contention winner 46, 81, 363
control entity 67
conversation 363
CPI-C 153, 363
CPI-C call
- CMACCP 154
- CMALLC 153
- CMCFMD 176, 178
- CMDEAL 154
- CMINIT 153
- CMRCV 154
- CMSDT 155
- CMSEND 154
- CMSERR 155, 177
- CMSPTR 155
- CMSSL 153
- CMSST 154

CPI-C program
- see also KDCS 153

CPI-RR 153, 363
CTRL see KDCS call

D

data protection 260
DEFINE CONNECTION 78, 216
DEFINE SESSIONS 80, 217
DEFINE TRANSACTION 84, 222, 223
dependent LU 366
Destination Process Name 255
DFSLUEE0 201
dialog 364
dialog with IMS 250
DISPLAY
- IMS command 213

Distributed Program Link 148, 364
distributed transaction processing 364
Distributed Transaction Processing (DTP) 110, 224
DL/I 198
DL/I-COBOL program
- example 200

documentation
- summary 14

DPN 244, 255, 261
dummy dialog 255
dump
- creating 67

dump files 67

E

EBCDIC 48, 49, 240
EDTNAME 244
end of service 113
Enterprise Extender 364
entity 36
entity trace 55, 64, 66
error diagnosis 263
establishing connections 62
example for generating
- openUTM-CICS via LU6.1 and UNIX systems 221
- openUTM-IMS via LU6.1 and UNIX systems 248

examples 237
- CPI-C programs (LU6.2) 208
- DL/I program (LU6.2) 202

exchange log name 60
EXEC CICS
  ALLOCATE 86, 111, 226
  BUILD ATTACH 227, 234
  CONNECT PROCESS 86, 112
  CONVERSE 114, 119, 230, 235
  EXTRACT ATTACH 231, 233
  EXTRACT PROCESS 118
  FREE 117, 120
  ISSUE ABEND 115, 119
  ISSUE CONFIRMATION 115, 119
  ISSUE ERROR 115, 119
  ISSUE PREPARE 116, 120, 129
  ISSUE SIGNAL 117, 120
  LINK 148
  RECEIVE 114, 118, 229, 233
  RETRIEVE 239, 256, 262
  RETURN 121, 235
  SEND 113, 118, 228, 234
  START 238, 256, 262
  SYNCPOINT 116, 120, 235
  SYNCPOINT ROLLBACK 116, 120, 141, 142, 172
  WAIT CONVID 117, 120
External Call Interface 148

F
first speaker session 364
FMH 365
FMH4 240, 365
FMH5 227, 231, 234, 365
FMH6 241, 255, 261, 365
FMH7 365
format names
  IMS with LU6.2 201
front-end transaction 364
function management header 365
functional unit commit 365

G
gateway
  Linux/AIX systems with IBM CS 31
  Solaris with SNAP-IX 30
gateway computer 29, 32, 34
GDS 365
general data stream 365
generating IMS transactions 249
generation 243
  of openUTM-LU62 42
generation file 42, 52, 53
  display name 53
generation program 52
GN call see IMS calls
GU call see IMS calls

H
half-session 366
host 366

I
IBM Communications Server 37
  as gateway on Linux/AIX 31
  generating LU6.2 interconnection 82, 95
  IMS example for LU6.2 191
  interconnection example 28, 32
  Solaris 27
IMS 366
  LU name in the case of LU6.2 182
  LU6.2 TP name 183
IMS administration
  LU6.2 213
IMS calls
  AUTH (LU6.2) 198
  CHNG 251
  CHNG (LU6.2) 198
  CMD (LU6.2) 198
  GCMD (LU6.2) 198
  GN 251
  GN (LU6.2) 198
  GU 251
  GU (LU6.2) 198
  INOY (LU6.2) 199
  ISRT 251
  ISRT (LU6.2) 198
  PURG (LU6.2) 199
  RETURN 251
  ROLB (LU6.2) 199
  ROLL 251

- ROLL (LU6.2) 199
- ROLLB 251
- ROLS (LU6.2) 199
- SETO (LU6.2) 199
- SETS (LU6.2) 199
- SETU (LU6.2) 199
- SYNC (LU6.2) 199

IMS definitions in the case of LU6.2 188
IMS programming 250
- CPI-C for LU6.2 201
- dummy dialog 262
- with DL/I for LU6.2 198

IMS startup parameters
- LU6.2 181

IMS transactions 183
independent LU 366
INFO see KDCS call
INIT see KDCS call
initiator 366
installation directory of openUTM-LU62 39
INSTANCE statement 43
ISC edit 244, 262
ISRT call see IMS calls

J
job receiver 224
job submitter 228

K
KCDF 240
KCDPN 257, 258, 259
KCMF 240
KCPRN 257, 258, 259
KCPSWORD 85
KCPWDTYP 85
KCRDPN 257
KCRPRN 257
KCSECTYP 85, 152
KCTAST 125
KCUIDTYP 85
KCUSERID 85
KCVGST 125
KDCS call
- APRO AM 79, 123, 146, 153, 178, 239
- APRO DM 79, 93, 123, 153, 236
- APRO IN 257, 259
- CTRL AB 124, 155
- CTRL PE 123, 136, 152, 154
- CTRL PR 123, 139, 152, 154, 169
- INFO GN 257, 258
- INIT PU 125
- MPUT EM 124, 132, 155, 162
- MPUT HM 124, 145, 155, 176
- MPUT RM 130
- PEND ER 124, 129, 155, 159
- PEND FR 124, 129, 155, 159
- PEND RE 138, 167
- PEND RS 124, 130, 154
- see also CICS 123
- see also CPI-C 153

L
language 269
Linux systems
- IBM CS gateway for LU6.2
  - interconnection 31
- LU6.2 interconnection via IBM CS 27

LNETNAME 245
LOC-LISTENER-PORT 47
LOC-TSEL 47
log name 368
log record 40, 368
logical unit 367
logical unit of work 367
LPAP statement 215, 244, 245
LSES statement 217, 246
LTAC statement 152, 249
LTERM definition
- IMS example 189

LU 367
LU6.1 interconnection
- direct with TRANSIT on Solaris 33
- to openUTM under BS2000/OSD 34
- to openUTM under UNIX systems 34
- to openUTM under Windwos systems 34

LU6.1 protocol 367
LU6.2 application 35
LU6.2 Edit Exit routine 201

LU6.2 interconnection 35
  direct via IBM CS on Solaris 27
  direct via SNAP-IX on Solaris 26
  direct via TRANSIT on Solaris 25
  to openUTM under BS2000/OSD 29, 32
  to openUTM under UNIX systems 29, 32
  to openUTM under Windows systems 29, 32
  with IBM CS gateway on Linux/AIX 31
  with SNAP-IX gateway on Solaris 30
LU6.2 protocol 367
LU6.2 TP name
  IMS 183
lukewarm 56
LUWID 367

M
macros 244, 249
major node definition
  for Enterprise Extender 190
mapped conversation 368
Message Format Service 245
message segments 240
messages 269
  u62_tp 270
  XAP-TP provider 319
mode 368
MODE definition
  IMS example for LU6.2 190
mode name 48, 122, 368
MPTN 361
MPUT see KDCS call

N
NCP 368
NETNAME 78, 216
NETNAMEQ 217
network name 368
notes regarding UTM-CICS programming 240
number of sessions 81

O
openUTM generation
  openUTM-CICS LU6.2 interconnection 83, 92, 94
openUTM programming
  dummy dialog 257
  for LU6.2 152
  IMS interconnection 254
openUTM-CICS interconnection 77, 255
openUTM-IMS interconnection 255
openUTM-LU62 35
  entity 36
  interconnection example 28, 29, 32
openUTM-LU62 generation with TNSX 91
  IMS example for LU6.2 195
openUTM-LU62 generation without TNSX 91
openUTM-LU62 generation without TNSX TNSX
  IMS-Beispiel für LU6.2 195
openUTM-LU62 generation without TSNX 93
openUTM-LU62 installation directory 39
openUTM-LU62 process 38
OSI-LPAP statement 45, 47
OSI-TP dialog 364
OSS 37

P
parallel connection 45
  busy 61
parallel sessions 78, 218, 246, 369
Partner-UTM-Anwendung
  TCP-Portnummer 50
password 85
  IMS 187
PEND see KDCS call
physical unit 369
PLU 369
primary LU 218, 369
Primary Resource Name 255
principal facility 369
PRN 255
processing acknowledge 113, 115, 119, 131, 145, 161, 176
programming example
  openUTM-IMS 252
protocol trace 67
PU 369

Q
queued job
    dummy dialog 255
    in CICS 133, 146
    in CPI-C 163, 178
    with and without transaction
        management 134, 147
queued jobs with IMS 250

R
RACF 85
    IMS 187
RDPN 255
README files 19
registration editor 57
REM-LISTENER-PORT 50
REM-NSEL 50
remote LU 50
REM-TSEL 50
request unit 369
RequestCommit 125
Resource Definition Online 215
restart 40, 240
resync 370
RETURN call see IMS calls
Return Destination Process Name 255
Return Primary Resource Name 255
RNETNAME 215, 244
ROLL call see IMS calls
ROLLB call see IMS calls
RPRN 255
RSES 217, 246
RTAC 233, 249
RU 369
RU size 81, 218, 370

S
sample generation
    openUTM-CICS via LU6.2 86
    openUTM-CICS via TRANSIT-CD 219
    openUTM-IMS transactions 249
sample program
    asynchronous openUTM-CICS 239
    openUTM-CICS via LU6.2 125
secondary LU 218, 369
security 79
security variants
    IMS 187
send authorization 113, 225, 228
sense data 265
service restart 240
service status 125
service under Windows systems 57
SESCHA statement 218, 240, 244, 261
session 370
    busy 61
    release 224
    setup 248
SESSNAME 217
side information
    IMS 184
    IMS example 189
SIDEINFO 184
SLU 369
SNA 371
SNAP-IX
    as gateway on Solaris 30
    Solaris 26
Solaris
    direct LU6.1 interconnection 33
    LU6.2 interconnection via SNAP-IX 26
    LU6.2 interconnection via TRANSIT 25
    SNAP-IX gateway for LU6.2
        interconnection 30
SRRBACK 153, 171
SRRCMIT 153
SSCP 371
standard IMS transactions
    LU6.2 209
starting openUTM-LU62 55
status information 59
Stellvertreter-AE
    Transportselektor 47
Stellvertreter-LU
    TCP-Portnummer 47
step 66
substitute AE 46
substitute concept 35

substitute LU 47
sync point 232, 371
sync-level 112, 118, 371
synclog file 40
sync-point initiator 366
SYSID 78, 111, 216, 226, 238
system initialization table 77, 215
system startup 56

T
TAC statement 223
TCP-Portnummer 47, 50
terminating openUTM-LU62 58
TH 371
time measurement 55, 64
TNSX 46, 49, 83, 92, 94
TNSX generation
  IMS example for LU6.2 196
TP 371
TP name 371
TP_Profile definition 188
trace level 55, 64
transaction code
  IMS 183
transaction program 371
transaction restart 40
transaction status 125
TRANSIT
  LU6.1 interconnection 33
TRANSIT-CD generation 249
TRANSIT-CLIENT
  generating openUTM-CICS
    interconnection 220
  interconnection example 33, 34
TRANSIT-CPIC 37
  interconnection example 25
TRANSIT-LU0 generation 249
TRANSIT-SERVER 25, 37
  generating LU6.2 interconnection 82, 90
transmission header 220, 371
Transportselektor 47
TRMSGLTH 244, 246
TX 153, 363

type 2.0 node 371
type 2.1 node 29, 372
type 4 node 372
type 5 node 372
typewriter font 21

U
u62_adm 53, 58, 62, 63, 64, 66
u62_gen 52, 53
u62_sta 59
u62_start 55
u62_svc 39
u62_tp 39
  messages 270
U62_TRC_FILES
  trace files, change quantity 65
UDP 372
UDT 152
UDTAC 45
UDTCCR 45
UDTDISAC 45
UDTSEC 45
underscore 21
user ID 85, 152
  IMS 187
USER.IMS.CNTL 188
userid see user ID
UTM generation
  IMS example for LU6.2 196
UTM-ASYNC 51
UTM-CICS interconnection 215
UTMD statement 50
UTMGW1 220
UTM-IMS interconnection 243

V
VLVB 216, 227, 245, 372
VLVB format 231
VTAM 372
VTAM generation 82, 89
  IMS example (LU6.2) 189
  IMS LU6.2 interconnection 186

W
warm start 40
    Lukewarm 56
Write Log daemon 39
Write Log program 39

X
XAP-TP 37
XAP-TP provider 37
    messages 319
XAP-TP trace 55, 64, 66
XMODE statement 45