Transaction Component Interface (TCI) Specification

Transaction Component Interface (TCI) Specification

Version 1.1 Edition 7.20141001
Updated October 25, 2014
Distributed with Package openss7-1.1.7.20141001

Abstract:

This document is a Specification containing technical details concerning the implementation of the Transaction Component Interface (TCI) for OpenSS7. It contains recommendations on software architecture as well as platform and system applicability of the Transaction Component Interface (TCI). It provides abstraction of the Transaction Component Handling (TC) interface to these components as well as providing a basis for Transaction Component Handling control for other Transaction Component Handling protocols.

Brian Bidulock <bidulock@openss7.org> **for**
The OpenSS7 Project <http://www.openss7.org/>

Published by:

OpenSS7 Corporation
1469 Jefferys Crescent
Edmonton, Alberta T6L 6T1
Canada

Notice:

Short Contents

Table of Contents

List of Figures

List of Tables

Preface

Notice

Software in this document and related software is released under the AGPL (see [GNU Affero General Public License], page 95). Please note, however, that there are different licensing terms for some of the manual package and some of the documentation. Consult permission notices contained in the documentation of those components for more information.
This document is released under the FDL (see [GNU Free Documentation License], page 105) with no invariant sections, no front-cover texts and no back-cover texts.

Abstract

This document is a Specification containing technical details concerning the implementation of the Transaction Component Interface (TCI) for OpenSS7. It contains recommendations on software architecture as well as platform and system applicability of the Transaction Component Interface (TCI).
This document specifies a Transaction Component Interface (TCI) Specification in support of the OpenSS7 Transaction Component Handling (TC) protocol stacks. It provides abstraction of the Transaction Component interface to these components as well as providing a basis for Transaction Component control for other Transaction Component protocols.

Purpose

The purpose of this document is to provide technical documentation of the Transaction Component Interface (TCI). This document is intended to be included with the OpenSS7 STREAMS software package released by *OpenSS7 Corporation*. It is intended to assist software developers, maintainers and users of the Transaction Component Interface (TCI) with understanding the software architecture and technical interfaces that are made available in the software package.

Intent

It is the intent of this document that it act as the primary source of information concerning the Transaction Component Interface (TCI). This document is intended to provide information for writers of OpenSS7 Transaction Component Interface (TCI) applications as well as writers of OpenSS7 Transaction Component Interface (TCI) Users.

Audience

The audience for this document is software developers, maintainers and users and integrators of the Transaction Component Interface (TCI). The target audience is developers and users of the OpenSS7 SS7 stack.

Revision History

Take care that you are working with a current version of this documentation: you will not be notified of updates. To ensure that you are working with a current version, check the OpenSS7 Project website for a current version.
A current version of this specification is normally distributed with the *OpenSS7* package, `openss7-1.1.7.20141001`.[1]

[1] `http://www.openss7.org/repos/tarballs/openss7-1.1.7.20141001.tar.bz2`

Version Control

Although the author has attempted to ensure that the information in this document is complete and correct, neither the Author nor OpenSS7 Corporation will take any responsibility in it. *OpenSS7 Corporation* is making this documentation available as a reference point for the industry. While *OpenSS7 Corporation* believes that these interfaces are well defined in this release of the document, minor changes may be made prior to products conforming to the interfaces being made available. *OpenSS7 Corporation* reserves the right to revise this software and documentation for any reason, including but not limited to, conformity with standards promulgated by various agencies, utilization of advances in the state of the technical arts, or the reflection of changes in the design of any techniques, or procedures embodied, described, or referred to herein. *OpenSS7 Corporation* is under no obligation to provide any feature listed herein.

```
$Log: tci.texi,v $
Revision 1.1.2.2  2011-02-07 02:21:46  brian
- updated manuals

Revision 1.1.2.1  2009-06-21 10:57:04  brian
- added files to new distro
```

ISO 9000 Compliance

Only the TEX, texinfo, or roff source for this maual is controlled. An opaque (printed, postscript or portable document format) version of this manual is a **UNCONTROLLED VERSION**.

Disclaimer

OpenSS7 Corporation **disclaims all warranties with regard to this documentation including all implied warranties of merchantability, fitness for a particular purpose, non-infrincement, or title; that the contents of the manual are suitable for any purpose, or that the implementation of such contents will not infringe on any third party patents, copyrights, trademarks or other rights. In no event shall** *OpenSS7 Corporation* **be liable for any direct, indirect, special or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action or contract, negligence or other tortious action, arising out of or in connection with any use of this documentation or the performance or implementation of the contents thereof.**

U.S. Government Restricted Rights

If you are licensing this Software on behalf of the U.S. Government ("Government"), the following provisions apply to you. If the Software is supplied by the Department of Defense ("DoD"), it is classified as "Commercial Computer Software" under paragraph 252.227-7014 of the DoD Supplement to the Federal Aquisition Regulations ("DFARS") (or any successor regulations) and the Government is acquiring only the license rights granded herein (the license rights customarily provided to non-Government users). If the Software is supplied to any unit or agency of the Government other than DoD, it is classified as "Restricted Computer Software" and the Government's rights in the Software are defined in paragraph 52.227-19 of the Federal Acquisition Regulations ("FAR") (or any successor regulations) or, in the cases of NASA, in paragraph 18.52.227-86 of the NASA Supplerment to the FAR (or any successor regulations).

Acknowledgements

The OpenSS7 Project was funded in part by:

- Monavacon Limited
- OpenSS7 Corporation

Thanks to the subscribers to and sponsors of The OpenSS7 Project. Without their support, open software like this would not be possible.

As with most open source projects, this project would not have been possible without the valiant efforts and productive software of the Free Software Foundation, the Linux Kernel Community, and the open source software movement at large.

1 Introduction

This document specifies a STREAMS-based kernel-level instantiation of the ITU-T Transaction Capabilities Application Part (TCAP) Component (TC) Sub-Layer. The Transaction Component Interface (TCI) enables the user of a component sub-layer service to access and use any of a variety of conforming transaction providers without specific knowledge of the provider's protocol. The service interface is designed to support any transaction protocol but is intended for the ITU-T Recommendation Q.771 Transaction Capabilities Application Part (TCAP) Component (TC) Sub-Layer. This interface only specifies access to transaction component sub-layer services providers, and does not address issues concerning transaction or component sub-layer management, protocol performance, and performance analysis tools.

The specification assumes that the reader is familiar with the ISO reference model terminology, ISO/ITU-T transaction service definitions (ROSE, ACSE, TCAP), and STREAMS.

1.1 Related Documentation

— ITU-T Recommendation X.200 (White Book) — ISO/IEC 7498-1:1994
— ITU-T Recommendation X.219 (White Book) — ISO/IEC
— ITU-T Recommendation X.229 (White Book) — ISO/IEC
— ITU-T Recommendation X.217 (White Book) — ISO/IEC 8649 : 1996
— ITU-T Recommendation X.227 (White Book) — ISO/IEC 8650-1 : 1995
— ITU-T Recommendation X.237 (White Book) — ISO/IEC 10035-1 : 1995
— ITU-T Recommendation Q.771 (White Book)
— System V Interface Definition, Issue 2 - Volume 3

1.1.1 Role

This document specifies an interface that supports the Transaction Component (TC) Sub-Layer services provided by the Transaction Capabilities Application Part (TCAP) as specified in ITU-T Recommendation Q.771. It may also be capable of supporting the transaction component capabilities of the Remote Operations Service Execution (ROSE) for Open Systems Interconnect for CCITT Applications as specified in ITU-T Recommendation X.219 and ISO ????. These specifications are targeted for use by developers and testers of protocol modules that require transaction component sub-layer service.[1]

1.2 Definitions, Acronyms, and Abbreviations

Originating TC User
: A TC-User that initiates a transaction.

Destination TC User
: A TC-User with whom an originating TC user wishes to establish a transaction dialogue.

ISO
: International Organization for Standardization

[1] An example of a protocol module that requires transaction component sub-layer services is the 3GPP TS 29.002 Mobile Application Part (MAP).

TC User Kernel level protocol or user level application that is accessing the services of the transaction component sub-layer.

TC Provider
Transaction sub-layer entity/entities that provide/s the services of the transaction component interface.

TCI Transaction Component Interface

TIDU Transaction Interface Data Unit

TSDU Transaction Service Data Unit

OSI Open Systems Interconnection

QOS Quality of Service

STREAMS A communication services development facility first available with UNIX System V Release 3

2 The Transaction Component Sub-Layer

The Transaction Component Sub-Layer provides the means to manage the dialogue of TC-Users into transaction components and dialogues. It is responsible for the routing and management of transaction component exchange within dialogues between TC-user entities.

2.1 Model of the TCI

The TCI defines the services provided by the transaction component sub-layer to the transaction component-user at the boundary between the Transaction Capabilities Application Part (TCAP) user and the Transaction Component (TC) Sub-Layer in the model presented in ITU-T Recommendation Q.771. The interface consists of a set of primitives defined as STREAMS messages that provide access to the component sub-layer services, and are transferred between the TC user entity and the TC provider. These primitives are of two types: ones that originate from the TC user, and others that originate from the TC provider, or respond to an event of the TC provider. The primitives that originate from the TC provider are either confirmations of a request or are indications to the TC user that the event has occurred. Figure 2.1 shows the model of the TCI.

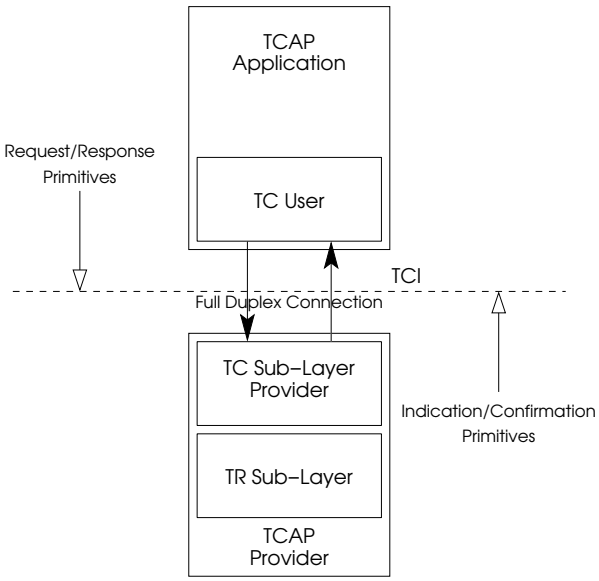

Figure 2.1: *Model of the TCI*

The TCI allows the TC provider to be configured with any component sub-layer user (such as the Mobile Application Part whose upper layer interface is described in *Mobile Application Part Interface*), that also conforms to the TCI. A transaction component sub-layer user can also be a user program that conforms to the TCI and accesses the TC provider via `putmsg(2s)` and `getmsg(2s)` system calls.

STREAMS messages that are used to communicate transaction component service primitives between the transaction component user and the transaction component provider may have one of the following formats:

1. A `M_PROTO` message block followed by zero or more `M_DATA` message blocks. The `M_PROTO` message block contains the type of service primitive and all relevant arguments associated with the primitive. The `M_DATA` blocks contain user data associated with the service primitive.
2. One `M_PCPROTO` message block containing the type of service primitive and all the relevant arguments associated with the primitive.
3. One or more `M_DATA` message blocks containing user data.

The following sections describe the service primitives which define all operation classes of service.

For all operation classes of service, two types of primitives exist: primitives that originate from the service user and primitives that originate from the service provider. The primitives that originate from the service user make requests to the service provider or response to an event of the service provider. The primitive that originate from the service provider are either confirmations of a request or are indications to the service user that an event has occurred. The primitive types along with the mapping of those primitives to the STREAMS message types and the service primitives of the ISO/IEC xxxxx and service definitions are listed in Chapter 4 [TCI Primitives], page 23. The format of these primitives and the rules governing the use of them are described in Section 4.1 [Management Primitives], page 24, Section 4.2 [Operation Class 1, 2 and 3 Primitives], page 41, and Section 4.3 [Operation Class 4 Primitives], page 58.

2.2 TCI Services

The features of the TCI are defined in terms of the services provided by the TC provider, and the individual primitives that may flow between the TC user and the TC provider.

The services supported by the TCI are based on four distinct classes of transaction, operation classes 1, 2, 3 and 4. In addition, the TCI supports services for local management.

2.2.1 Operation Class 1

The main features of operation class 1 transactions are:

- Operation success is reported.
- Operation failure is reported.

There are three phases to each transaction: Transaction Initiation, Transaction Data Transfer, and Transaction Termination.[1] Transaction components arrive at their destination in the same order as they departed their source and the data is protected against duplication or loss of data units within some specified quality of service.

2.2.2 Operation Class 2

The main features of operation class 2 transactions are:

- Operation success is *not* reported.
- Operation failure is reported.

There are three phases to each transaction: Transaction Initiation, Transaction Data Transfer, and Transaction Termination.[2] Transaction components arrive at their destination in the same order as they departed their source and the data is protected against duplication or loss of data units within some specified quality of service.

[1] All three phases in operation class 1 can be combined into a single exchange of primitives.
[2] All three phases in operation class 2 can be combined into a single exchange of primitives.

2.2.3 Operation Class 3

The main features of operation class 3 transactions are:

- Operation success is reported.
- Operation failure is *not* reported.

There are three phases to each transaction: Transaction Initiation, Transaction Data Transfer, and Transaction Termination.[3] Transaction components arrive at their destination in the same order as they departed their source and the data is protected against duplication or loss of data units within some specified quality of service.

2.2.4 Operation Class 4

The main features of operation class 4 transactions are:

- Operation success is *not* reported.
- Operation failure is *not* reported.

Operation class 4 has no structure to the transaction and has no separate phases. Each transaction component is transmitted from source to destination independently, appropriate addressing information is included with each component sequence. As the components are transmitted independently from source to destination, there are, in general, no guarantees of proper sequence and completeness of the data transmission.

2.2.5 Component Handling

TC-Invoke	1	2	3	4
TC-Result	1	–	3	–
TC-Error	1	2	–	–
TC-Cancel	1	2	3	–
TC-Reject	1	2	–	4

2.2.6 Local Management

The TCI specifications also define a set of local management functions that apply to all operation classes. These services have local significance only.
Table 1 and Table 2 summarize the TCI service primitives by their state and service.

Table 1. *Service Primitives for Operation Classes 1, 2 and 3*

STATE	**SERVICE**	**PRIMITIVES**
Local Management	Information Reporting	TC_INFO_REQ, TC_INFO_ACK, TC_ERROR_ACK
	Bind	TC_BIND_REQ, TC_BIND_ACK, TC_UNBIND_REQ, TC_OK_ACK, TC_ERROR_ACK
	Options Management	TC_OPTMGMT_REQ, TC_OK_ACK, TC_ERROR_ACK
Transaction Initiation	Transaction Begin	TC_BEGIN_REQ, TC_BEGIN_IND, TC_BEGIN_RES, TC_BEGIN_CON, TC_TOKEN_REQ, TC_TOKEN_ACK, TC_OK_ACK, TC_ERROR_ACK

[3] All three phases in operation class 3 can be combined into a single exchange of primitives.

Transaction Data Transfer	Transaction Continue	TC_CONT_REQ, TC_CONT_IND
Transaction Release	Transaction End	TC_END_REQ, TC_END_IND
	Transaction Abort	TC_ABORT_REQ, TC_ABORT_IND

Table 2. *Service Primitives for Operation Class 4*

STATE	**SERVICE**	**PRIMITIVES**
Local Management	Information Reporting	TC_INFO_REQ, TC_INFO_ACK, TC_ERROR_ACK
	Bind	TC_BIND_REQ, TC_BIND_ACK, TC_UNBIND_REQ, TC_OK_ACK, TC_ERROR_ACK
	Options Management	TC_OPTMGMT_REQ, TC_OK_ACK, TC_ERROR_ACK
Transaction Unitdata	Transaction Unidirectional	TC_UNI_REQ, TC_UNI_IND

3 TCI Services Definition

This section describes the services of the TCI primitives. Time-sequence diagrams[1] that illustrate the sequence of primitives are used. The format of the primitives will be defined later in this document.

3.1 Local Management Services Definition

The services defined in this section are outside the scope of the international standards. These services apply to all operation classes. They are involved for the initialization/de-initialization of a *Stream* connected to the TC provider. They are also used to manage options supported by the TC provider and to report information on the supported parameter values.

3.1.1 Transaction Information Reporting Service

This service provides information on the options supported by the TC provider.

- `TC_INFO_REQ`: This primitive request that the TC provider returns the values of all the supported protocol parameters. This request may be invoked during any phase.
- `TC_INFO_ACK`: This primitive is in response to the *TC_INFO_REQ* primitive and returns the values of the supported protocol parameters to the TC user.

The sequence of primitives for transaction information management is shown in Figure 3.1.

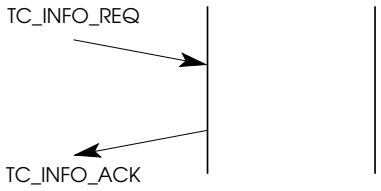

Figure 3.1: *Sequence of Primitives – Transaction Information Reporting Service*

3.1.2 TC User Bind Service

This service allows an originating address to be associated with a *Stream*. It allows the TC user to negotiate the number of transaction begin indications that can remain unacknowledged for that TC user (a transaction begin indication is considered unacknowledged while it is awaiting a corresponding transaction response or abort request from the TC user). This service also defines a mechanism that allows a *Stream* (bound to the address of the TC user) to be reserved to handle incoming transactions only. This *Stream* is referred to as the listener *Stream*.

- `TC_BIND_REQ`: This primitive request that the TC user be bound to a particular originating address, and negotiate the number of allowable outstanding transaction indications for that address.
- `TC_BIND_ACK`: This primitive is in response to the *TC_BIND_REQ* primitive and indicates to the user that the specified TC user has been bound to an originating address.

[1] Conventions for the time-sequence diagrams are defined in ITU-T X.210, ISO/IEC 10731:1994.

The sequence of primitives for the TC user bind service is shown in Figure 3.2.

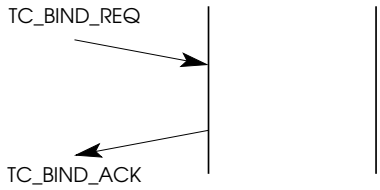

Figure 3.2: *Sequence of Primitives – TC User Bind Service*

3.1.3 TC User Unbind Service

This service allows the TC user to be unbound from a network address.

- `TC_UNBIND_REQ`: This primitive requests that the TC user be unbound from the network address it had previously been bound to.

The sequence of primitives for the TC user unbind service is shown in Figure 3.3.

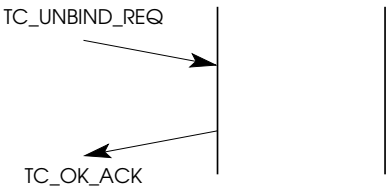

Figure 3.3: *Sequence of Primitives – TC User Unbind Receipt Acknowledgement Services*

3.1.4 Receipt Acknowledgement Service

- `TC_OK_ACK`: This primitive indicates to the TC user that the previous TC user originated primitive was received successfully by the TC provider.

An example showing the sequence of primitives for successful receive acknowledgement is depicted in Figure 3.3.

3.1.5 Options Mangement Service

This service allows the TC user to manage the QOS parameter values associated with the TC provider.

- `TC_OPTMGMT_REQ`: This primitive allows the TC user to select default values for QOS parameters within the range supported by the TC provider, and to indicate the default selection of return option.
- `TC_OPTMGMT_ACK`:

Figure 3.4 shows the sequence of primitives for transaction options management.

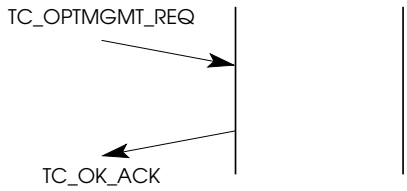

Figure 3.4: *Sequence of Primitives – Options Management Service*

3.1.6 Error Acknowledgement Service

- `TC_ERROR_ACK`: This primitive indicates to the TC user that a non-fatal error has occurred in the last TC user originated request or response primitive (listed in Figure 3.5) on the *Stream*.

Figure 3.5 shows the sequence of primitives for the error management primitive.

Figure 3.5: *Sequence of Primitives – Error Acknowledgement Service*

3.2 Operation Class 1, 2 and 3 Transaction Services Definition

This section describes the required transaction service primitives that define the operation class 1, 2 and 3, structured transaction interface.

The queue model for operation classes 1, 2 and 3 are discussed in more detail in ITU-T X.219 and ITU-T Q.771.

The queue model represents the operation of a transaction dialogue in the abstract by a pair of queues linking two transaction users. There is one queue for each direction of component flow. Each queue represents a flow control function in one direction of transfer. The ability of a user to add objects to a queue will be determined by the behaviour of the user removing objects from that queue, and the state of the queue. The pair of queues is considered to be available for each potential transaction dialogue. Objects that are entered or removed from the queue are either as a result of interactions at the two transaction addresses, or as the result of TC provider initiatives.

- A queue is empty until a transaction object has been entered and can be returned to this state, with loss of its contents, by the TC provider.
- Objects may be entered into a queue as a result of the actions of the source TC user, subject to control by the TC provider.
- Objects may also be entered into a queue by the TC provider.
- Objects are removed from the queue under the control of the TC user in the same order as they were entered except:

1. If the object is of type defined to be able to advance ahead of the preceding object (however, no object is defined to be able to advance ahead of another object of the same type), or
2. If the following object is defined to be destructive with respect to the preceding object on the queue. If necessary, the last object on the queue will be deleted to allow a destructive object to be entered - they will therefore always be added to the queue. For example, "abort" objects are defined to be destructive with respect to all other objects.

Table 3 shows the ordering relationships among the queue model objects.

Table 3. *Ordering Relationships Between Queue Model Objects*

Object X / **Object Y**	**BEGIN**	**CONT**	**END**	**ABORT**
BEGIN	N/A	-	-	DES
CONT	N/A	-	-	DES
END	N/A	N/A	-	-

AA	Indicates that Object X is defined to be able to advance ahead of preceding Object Y
DES	Indicates that Object X is defined to be destructive with respect to the preceding Object Y.
-	Indicates that Object X is neither destructive with respect to Object Y, nor able to advance ahead of Object Y
N/A	Indicates that Object X will not occur in a position succeeding Object Y in a valid state of a queue.

3.2.1 Transaction Initiation

A pair of queues is associated with a transaction dialogue between two transaction users when the TC provider receives a `TC_BEGIN_REQ` primitive at one of the TC users resulting in a begin object being entered into the queue. The queues will remain associated with the transaction until a `TC_END_REQ` or `TC_ABORT_REQ` primitive (resulting in an end or abort object) is either entered or removed from a queue. Similarly, in the queue from the destination TC user, objects can be entered into the queue only after the begin object associated with the `TC_BEGIN_RES` has been entered into the queue. Alternatively, the destination TC user can enter an end or abort object into the queue instead of the begin object to terminate the transaction.

The transaction establishment procedure will fail if the TC provider is unable to establish a transaction dialogue, or if the destination TC user is unable to accept the `TC_BEGIN_IND` (see Transaction Termination primitive definition in Section 4.2.3.2 [Transaction End Indication], page 55).

3.2.1.1 User Primitives for Successful Transaction Establishment

The following user primitves support Operation Class 1, 2, or 3 Phase I (Transaction Establishment) services:

- `TC_BEGIN_REQ`: This primitive requests that the TC provider form a transaction dialogue with the specified destination TC user.
- `TC_BEGIN_RES`: This primitive requests that the TC provider accept a previous transaction indication.

3.2.1.2 Provider Primitives for Successful Transaction Establishment

The following provider primitives support Operation Class 1, 2, or 3 Phase I (Transaction Establishment) services:

- `TC_BEGIN_IND`: This primitive indicates to the TC user that a transaction dialogue request has been made by a user at the specified source address.
- `TC_BEGIN_CON`: This primitive indicates to the TC user that a transaction initiation request has been confirmed on the specified responding address.

The sequence of primitives in a successful transaction initiation is defined by the time sequence diagrams as shown in Figure 3.6.

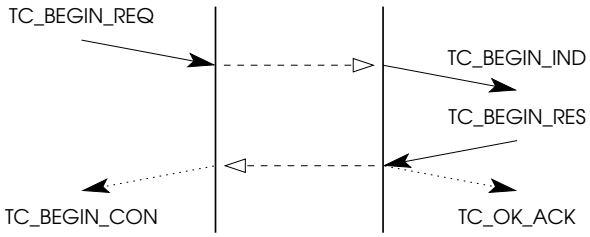

Figure 3.6: *Sequence of Primitives – Successful Transaction Initiation*

The sequence of primitives for the transaction initiation response token value determination is shown in Figure 3.7 (procedures for transaction initiation response token value determination are discussed in Section 4.1.2.1 [Transaction Bind Request], page 28, and Section 4.1.2.2 [Transaction Bind Acknowledgement], page 30).

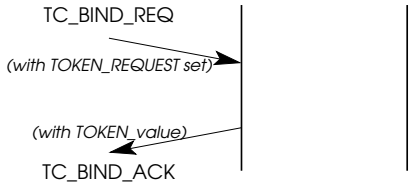

Figure 3.7: *Sequence of Primitives – Transaction Reponse Token Value Determination*

3.2.2 Transaction Component Transfer

Flow control on the transaction dialogue is done by management of the queue capacity, and by allowing objects of certain types to be inserted to the queues, as shown in *Table 4*.

3.2.2.1 Primitives for Component Transfer

The following primitives support Operation Class 1, 2, or 3 Phase II (Transaction Component Transfer) services:

- `TC_CONT_REQ`: This primitive requests that the TC provider transfer the specified components.
- `TC_CONT_IND`: This primitive indicates to the TC user that this message contains components.

Figure 3.8 shows the sequence of primitives for successful component transfer. The sequence of primitives may remain incomplete if a `TC_END_REQ`, `TC_ABORT_REQ`, or `TC_ABORT_IND` primitive occurs.

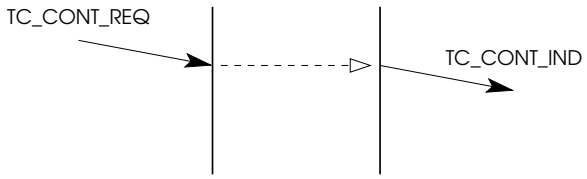

Figure 3.8: *Sequence of Primitives – Component Transfer*

3.2.3 Transaction Termination

The transaction dialogue procedure is initialized by insertion of an end or abort object (associated with a `TC_END_REQ` or `TC_ABORT_REQ`) into the queue. As shown in Table?, the termination procedure is destructive with respect to other objects in the queue, and eventually results in the emptying of queues and termination of the transaction dialogue.

The sequence of primitives depends on the origin of the termination action. The sequence may be:

1. invoked by on TC user, with a request from that TC user leading to an indication to the other;
2. invoked by both TC users, with a request from each of the TC users;
3. invoked by the TC provider, with an indication to each of the TC users;
4. invoked independently by one TC user and the TC provider, with a request from the originating TC user and an indication to the other.

3.2.3.1 Primitives for Transaction Termination

The following primitives support Operation Class 1, 2, or 3 Phase III (Transaction Termination) services:

- `TC_END_REQ`: This primitive requests that the TC provider deny an outstanding request for a transaction dialogue or normal termination of an existing transaction.
- `TC_ABORT_REQ`: This primitive requests that the TC provider deny an outstanding request for a transaction dialogue or abnormal termination of an existing transaction.
- `TC_END_IND`: This primitive indicates to the TC user that either a request for transaction initiation has been denied or an existing transaction has been terminated normally.
- `TC_ABORT_IND`: This primitive indicates to the TC user that either a request for transaction initiation has been denied or an existing transaction has been terminated abnormally.

The sequence of primitives are shown in the time sequence diagrams in the figures that follow:

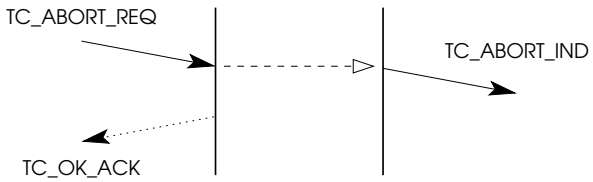

Figure 3.9: *Sequence of Primitives – TC User Invoked Termination*

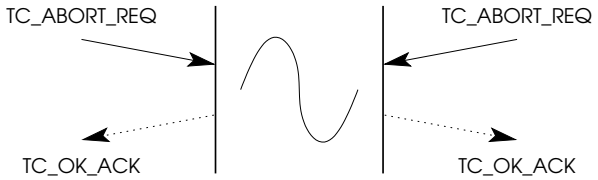

Figure 3.10: *Sequence of Primitives – Simultaneous TC User Invoked Termination*

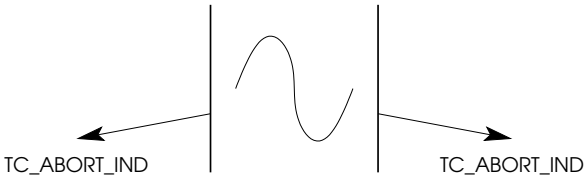

Figure 3.11: *Sequence of Primitives – TC Provider Invoked Termination*

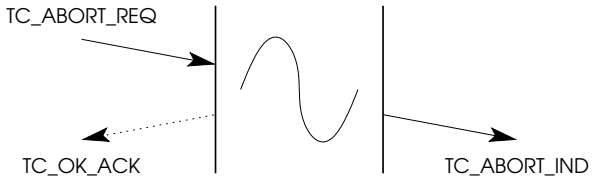

Figure 3.12: *Sequence of Primitives – Simultaneous TC User and Provider Invoked Termination*

A TC user may reject a transaction initiation attempt by issuing a `TC_ABORT_REQ`. The originator parameter in the `TC_ABORT_REQ` will indicate TC user invoked termination. The sequence of primitives is shown in Figure 3.13.

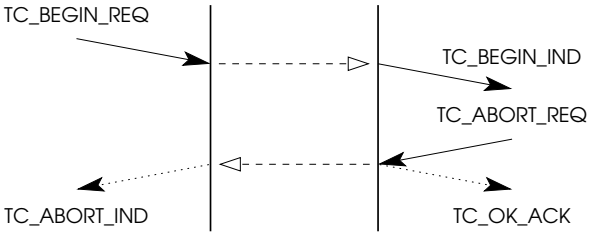

Figure 3.13: *Sequence of Primitives – TC User Rejection of a Transaction Initiation Attempt*

If the TC provider is unable to establish a transaction, it indicates this to the requester by an `TC_ABORT_IND`. The originator of the primitive indicates a TC provider invoked release. This is shown in Figure 3.14.

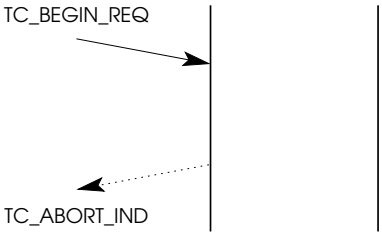

Figure 3.14: *Sequence of Primitives – TC Provider Rejection of a Transaction Initiation Attempt*

3.3 Operation Class 4 Transaction Services Definition

The operation class 4 service allows for the transfer of transaction components in one and both directions simultaneously without establishing a transaction dialogue. A set of primitives are defined that carry transaction components and control information between the TC user and the TC provider entities. The primitives are modelled as requests initiated by the TC user and indications initiated by the TC provider. Indications may be initiated by the TC provider independently from requests by the TC user. The operation class 4 transaction service consists of one phase.

3.3.1 Request and Response Primitives

- `TC_UNI_REQ`: This primitive requests that the TC provider send the transaction components to the specified destination.
- `TC_UNI_IND`: This primitive indicates to the TC user that a component sequence has been received from the specified originating address.

Figure 3.15 shows the sequence of primitives for the operation class 4 mode of transfer.

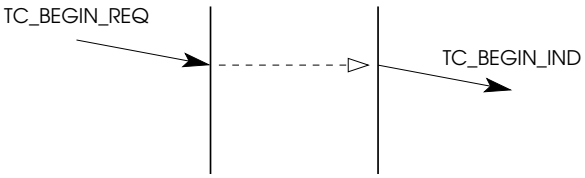

Figure 3.15: *Sequence of Primitives – Operations Class 4 Component Transfer*

- `TC_NOTICE_IND`:
  This primitive indicates to the TC user that the components with the specified destination address and QOS parameters produced an error. This primitive is specific to operation class 4.

Figure 3.16 shows the sequence of primitives for the operation class 4 error management primitive.

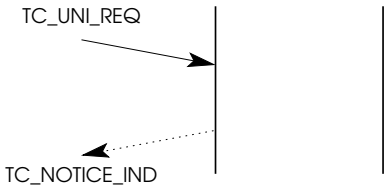

Figure 3.16: *Sequence of Primitives – Operations Class 4 Indication Service*

3.4 Component Handling Services Definition

3.4.1 Component Invoke Service

- `TC_INVOKE_REQ`:
- `TC_INVOKE_IND`:

3.4.2 Component Return Result Service

- `TC_RESULT_REQ`:
- `TC_RESULT_IND`:

3.4.3 Component Error Service

- `TC_ERROR_REQ`:
- `TC_ERROR_IND`:

3.4.4 Component Cancel Service

- `TC_CANCEL_REQ`:
- `TC_CANCEL_IND`:

3.4.5 Component Reject Service

- `TC_REJECT_REQ`:
- `TC_REJECT_IND`:

4 TCI Primitives

This section describes the format and parameters of the TCI primitives (Appendix A shows the mapping of TCI primitives to the primitives defined in ITU-T Q.771). In addition, it discusses the states in which the primitive is valid, the resulting state, and the acknowledgement that the primitive expects. (The state/event tables for these primitives are shown in Appendix B. The precedence tables for the TCI primitives are shown in Appendix C.) Rules for SS7 conformance are described in Addendum 1 to this document. The following tables provide a summary of the TC primitives and their parameters.

Table 4. *Transaction Initiation Transaction Service Primitives*

SERVICE	**PRIMITIVE**	**PARAMETERS**
TC Initiation	`TC_BEGIN_REQ`	()
	`TC_BEGIN_IND`	()
	`TC_BEGIN_RES`	()
	`TC_BEGIN_CON`	()

Table 5. *Transaction Continuation Transaction Service Primitives*

SERVICE	**PRIMITIVE**	**PARAMETERS**
TC Initiation	`TC_CONT_REQ`	()
	`TC_CONT_IND`	()

Table 6. *Transaction Termination Transaction Service Primitives*

SERVICE	**PRIMITIVE**	**PARAMETERS**
TC Initiation	`TC_END_REQ`	()
	`TC_END_IND`	()
	`TC_ABORT_REQ`	()
	`TC_ABORT_IND`	()

4.1 Management Primitives

These primitives apply to all operation classes.

4.1.1 Transaction Information

4.1.1.1 Transaction Information Request

TC_INFO_REQ

This primitive request the TC provider to return the values of all supported protocol parameters (see Section 4.1.1.2 [Transaction Information Acknowledgement], page 26), and also the current state of the TC provider (as defined in ⟨undefined⟩ [⟨undefined⟩], page ⟨undefined⟩). This primitive does not affect the state of the TC provider and does not appear in the state tables.

Format

The format of the message is one `M_PCPROTO` message block and its structure is as follows:

```
typedef struct TC_info_req {
    ulong PRIM_type;      /* Always TC_INFO_REQ */
} TC_info_req_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Indicates the primitive type. Always `TC_INFO_REQ`.

Modes

This primitive is valid in Operations Classes 1, 2, 3, or 4.

Originator

This primitive is originated by the TC User.

Valid States

This primitive is valid in any state where a local acknowledgement is not pending.

New State

The new state remains unchanged.

Rules

For the rules governing the requests made by this primitive, see the `TC_INFO_ACK` primitive described in Section 4.1.1.2 [Transaction Information Acknowledgement], page 26.

Acknowledgements

This primitive requires the TC provider to generate one of the following acknowledgements upon receipt of the primitive:

— *Successful:* Acknowledgement of the primitive is indicatd with the `TC_INFO_ACK` primitive described in Section 4.1.1.2 [Transaction Information Acknowledgement], page 26.

— *Non-fatal Errors:* These errors will be indicated with the `TC_ERROR_ACK` primitive described in Section 4.1.4.2 [Transaction Error Acknowledgement], page 38. The allowable errors are as follows:

There are no errors associated with the issuance of this primitive.

4.1.1.2 Transaction Information Acknowledgement

`TC_INFO_ACK`

This primitive indicates to the TC user any relevant protocol-dependent parameters.[1] It should be initiated in response to the `TC_INFO_REQ` primitive described above under Section 4.1.1.1 [Transaction Information Request], page 24.

Format

The format of the message is one `M_PCPROTO` message block and its structure is as follows:

```
typedef struct TC_info_ack {
    long PRIM_type;       /* always TC_INFO_ACK */
    long TSDU_size;       /* maximum TSDU size */
    long ETSDU_size;      /* maximum ETSDU size */
    long CDATA_size;      /* connect data size */
    long DDATA_size;      /* disconnect data size */
    long ADDR_size;       /* maximum address size */
    long OPT_size;        /* maximum options size */
    long TIDU_size;       /* transaction interface data size */
    long SERV_type;       /* service type */
    long CURRENT_state;   /* current state */
    long PROVIDER_flag;   /* provider flags */
    long TCI_version;     /* TCI version */
} TC_info_ack_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Indicates the primitive type. Always `TC_INFO_ACK`.

TSDU_size
: Indicates the maximum size (in octets) of Transaction Service User Data supported by the TR provider.

ETSDU_size
: Indicates the maximum size (in octets) of Expedited Transaction Service User Data supported by the TR provider.

CDATA_size
: Indicates the maximum number of octets of data that may be associated with a transaction initiation primitive.

DDATA_size
: Indicates the maximum number of octets of data that may be associated with a transaction termination primitive.

ADDR_size
: Indicates the maximum size (in decimal digits) of a network address.

OPT_size
: Indicates the maximum size (in decimal digits) of the protocol options.

[1]

TIDU_size Indicates the maximum amount of TC user data that may be present in a single `TC_CONT_REQ` primitive. This is the size fo the transaction protocol interface data unit, and should not exceed the tunable system limit, if non-zero, for the size of a STREAMS message.

SERV_type Indicates the service type supported by the TC provider, and is a bitwise OR of zero or more of the following:

`TC_OPCLASS1`
: Indicates that the TC provider service is operations class 1.

`TC_OPCLASS2`
: Indicates that the TC provider service is operations class 2.

`TC_OPCLASS3`
: Indicates that the TC provider service is operations class 3.

`TC_OPCLASS4`
: Indicates that the TC provider service is operations class 4.

CURRENT_state
: Indicates the current state of the TC provider.

PROVIDER_flag
: Indicates additional properties specific to the TC provider and may alter the way the TC user communicates. The following flags may be set by the TC provider:

`SENDZERO` Indicates that the TC provider supports the sending of zero-length TSDUs.

`XPG4_1` Indicates that the TC provider supports XPG4 semantics.

TCI_version
: Indicates the version of the TC interface. The current version is Version 1.

Modes

This primitive is valid in Operations Classes 1, 2, 3, or 4.

Originator

This primitive is originated by the TC provider.

Valid State

This primitive is valid in repsonse to a `TC_INFO_REQ` primitive.

New State

The state is unchanged.

Rules

The following rules apply when the TC provider issues the `TC_INFO_ACK` primitive:

4.1.2 Transaction Protocol Address Management

4.1.2.1 Transaction Bind Request

`TC_BIND_REQ`

This primitive requests that the TC provider bind a protocol address to the *Stream*, negotiate the number of transaction dialogue begin indications allowed to be outstanding by the TC provider for the specified protocol address, and activates the *Stream* associated with the protocol address.[1]

Format

This message consists of one `M_PROTO` message block formatted as follows:

```
typedef struct TC_bind_req {
    ulong PRIM_type;
    ulong ADDR_length;    /* address length */
    ulong ADDR_offset;    /* address offset */
    ulong XACT_number;    /* maximum outstanding transaction reqs. */
    ulong BIND_flags;     /* bind flags */
} TC_bind_req_t;

typedef struct TC_subs_bind_req {
    ulong PRIM_type;
} TC_subs_bind_req_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Specifies the primitive type. Always `TC_BIND_REQ`.

ADDR_length
: Specifies the lengt of the protocol address to be bound to the *Stream*.

ADDR_offset
: Specifies the offset from the beginning of the `M_PROTO` message block where the protocol address begins. Note that all lengths, offsets, and sizes in all structures refer to the number of bytes.

XACT_number
: Specifies the requested number of transaction dialogue begin indications allowed to be outstanding by the TC provider for the specified protocol address.[2]

BIND_flags Specifies the options flags associated with the bind.

[1] Note that a *Stream* is viewed as active when the TC provider may receive and transmit TPDUs (transaction protocol data units) associated with the *Stream*.

[2] Note that the *XACT_number* should be ignored by those providing a connectionless (only) transaction service. Also note that if the number of outstanding transaction dialogue begin indications equals *XACT_number*, the TC provider need not discard further incoming transaction dialogue begin indications, but may choose to queue then internally until the number of outstanding transaction dialogue begin indications drops below *XACT_number*.

Flags

None.

Modes

This primitive is valid in Operations Classes 1, 2, 3, and 4.

Originator

This primitive is originated by the TC user.

Valid State

This primitive is valid in state `TCS_UNBND`.

New State

The new state is `TCS_WACK_BREQ`.

Rules

For rules governing the requests made by these primitives, see the `TC_BIND_ACK` primitive, Section 4.1.2.2 [Transaction Bind Acknowledgement], page 30.

Acknowledgement

This primitive requires the TC provider to generate one of the following acknowledgements on receipt of the primitive, and the TC user must wait for acknowledgement before issuing any other primitives:

— *Successful:* Correct acknowledgement of the primitive is indicated with the `TC_BIND_ACK` primitive, Section 4.1.2.2 [Transaction Bind Acknowledgement], page 30.

— *Non-fatal Errors:* These errore will be indicated with the `TC_ERROR_ACK` primitive described in Section 4.1.4.2 [Transaction Error Acknowledgement], page 38. The allowable errors are as follows:

`[TCACCES]` This error indicates that the TC user did not have proper permissions for the use of the requested address.

`[TCADDRBUSY]`
This error indicates that the requested address is in use.

`[TCBADADDR]`
This error indicates that the protocol address was in an incorrect format or the address contained invalid information. It is not intended to indicate protocol errors.

`[TCNOADDR]`
This error indicates that the TC provider could not allocate an address.

`[TCOUTSTATE]`
This error indicates that the primitive would place the transaction component interface out of state.

`[TCSYSERR]`
This error indicates that a system error has occurred and that the Linux system error is indicated in the primitive.

4.1.2.2 Transaction Bind Acknowledgement

`TC_BIND_ACK`

This primitive indicates to the TC user that the specified protocol address has been bound to the *Stream*, that the specified number of transaction association begin indications are allowed to be queued by the TC provider for the specified protocol address, and that the *Stream* associated with the specified protocol address has been activated.

Format

This message consists of one `M_PCPROTO` message block formatted as follows:

```
typedef struct TC_bind_ack {
    ulong PRIM_type;
    ulong ADDR_length;
    ulong ADDR_offset;
    ulong XACT_number;
    ulong TOKEN_value;
} TC_bind_ack_t;

typedef struct TC_subs_bind_ack {
    ulong PRIM_type;
} TC_subs_bind_ack_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Indicates the primitive type. Always `TC_BIND_ACK`.

ADDR_length
: Indicates the length of the protocol address that was bound to the *Stream*.

ADDR_offset
: Indicates the offset from the beginning of the `M_PCPROTO` message block where the protocol address begins.

XACT_number
: Indicates the accepted number of transaction begin indications allowed ot be outstanding by the TC provider for the specified protocol address. Note that this field does not apply to Operations Class 4 (only) TC providers.

TOKEN_value
: Unused.

Flags

Modes

This primitive is valid in Operations Classes 1, 2, 3 or 4.

Originator

This primitive is originated by the TC provider.

Valid State

This primitive is only issued by the TC provider in the `TCS_WACK_BREQ` state.

New State

The new state is `TCS_IDLE`.

Rules

The following rules apply to the binding of the specified protocol address to the *Stream*:

— If the *ADDR_length* field in the `TC_BIND_REQ` primitive is zero ('`0`'), then the TC provider is to assign a transaction protocol address to the TC user. If the TC provider cannot assign a transaction protocol address, the TC provider will return `[TCNOADDR]`.

— The TC provider is to bind the transaction protocol address as specified in the `TC_BIND_REQ` primitive.

— If the TC provider cannot bind the specified address, the TC provider will return `[TCADDRBUSY]`.

The following rules apply to negotiating the *XACT_number* argument:

— The returned value must be less than or equal to the corresponding requested number as indicated in the `TC_BIND_REQ` primitive.

— If the requested value is greater than zero, the returned value must also be greater than zero.

— Only one *Stream* that is bound to the indicated protocol address may have a negotiated accepted number of maximum transaction association begin requests greater than zero.

— A *Stream* requesting an *XACT_number* of zero should always be valid. This indicates to the TC provider that the *Stream* is to be used to request transaction associations only.

— A *Stream* with a negotiated *XACT_number* greater than zero may generate transaction association begin requests or accept transaction association begin indications.

Acknowledgement

If the above rules result in an error condition, then the TC provider must issue an `TC_ERROR_ACK` primitive to the TC user indicating the error as defined in the description of the `TC_BIND_REQ` primitive, Section 4.1.2.1 [Transaction Bind Request], page 28.

4.1.2.3 Transaction Unbind Request

`TC_UNBIND_REQ`

This primitive requests that the TC provider unbind the protocol address associated with the *Stream* and deactivate the *Stream*.

Format

This message consists of a `M_PROTO` message block, formatted as follows:

```
typedef struct TC_unbind_req {
    ulong PRIM_type;      /* Always TC_UNBIND_REQ */
} TC_unbind_req_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Indicates the primitive type. Always `TC_UNBIND_REQ`.

Modes

This primitive is valid in Operations Classes 1, 2, 3 or 4.

Originator

This primitive is originated by the TC user.

Valid State

This primitive is valid in state `TCS_IDLE`.

New State

The new state, when successful, is `TCS_WACK_UREQ`.

Acknowledgement

This primitive requires the TC provider to generate the following acknowledgements on receipt of the primitive and that the TC user must wait for the acknowledgement before issuing any other primitive:

- *Successful:* Correct acknowledgement of the primitive is indicated with the `TC_OK_ACK` primitive described in Section 4.1.4.1 [Transaction Successful Receipt Acknowledgement], page 37.
- *Non-fatal errors:* These errrors will be indicated with the `TC_ERROR_ACK` primitive described in Section 4.1.4.2 [Transaction Error Acknowledgement], page 38. The allowable errors are as follows:

  `[TCOUTSTATE]`
  : The primitive would place the transaction component interface out of state.

  `[TCSYSERR]`
  : A system error has occurred and the *Linux* system error is indicated in the primitive.

4.1.3 Transaction Options Management

4.1.3.1 Transaction Options Management Request

`TC_OPTMGMT_REQ`

This primitive allows the TC user to manage the options associated with the *Stream*. The format of the message is one `M_PROTO` message block.

Format

This message consists of one `M_PROTO` message block formatted as follows:

```
typedef struct TC_optmgmt_req {
    ulong PRIM_type;
    ulong OPT_length;
    ulong OPT_offset;
    ulong MGMT_flags;
} TC_optmgmt_req_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Specifies the primitive type. Always `TC_OPTMGMT_REQ`.

OPT_length
: Specifies the length of the protocol options associated with the primitive.

OPT_offset
: Specifies the offset from the beginning of the `M_PROTO` message block where the protocol options begin.

MGMT_flags
: Specifies the management flags.

Flags

The allowable *MGMT_flags* are as follows:

`TC_NEGOTIATE`
: Negotiate and set the options with the TC provider.

`TC_CHECK`
: Check the validity of the specified options.

`TC_DEFAULT`
: Return the default options.

`TC_CURRENT`
: Return the currently effective option values.

Modes

This primitive is valid in Operations Classes 1, 2, 3 or 4.

Originator

This primitive is originated by the TC user.

Valid State

This primitive is valid in any state where a local acknowledgement is not pending.

New State

The new state remains unchanged.

Rules

For the rules governing the requests made by this primitive see the `TC_OPTMGMT_ACK` primitive, Section 4.1.3.2 [Transaction Options Management Acknowledgement], page 35.

Acknowledgement

This primitive requires the TC provider to generate one of the following acknowledgements on receipt of the primitive, and that the TC user wait for the acknolwedgement before issuing any other primitives.

— *Successful:* Acknowledgement of the primitive is with the `TC_OPTMGMT_ACK` primitive, Section 4.1.3.2 [Transaction Options Management Acknowledgement], page 35.

— *Non-fatal errors:* These errors will be indicated via the `TC_ERROR_ACK` primitive described in Section 4.1.4.2 [Transaction Error Acknowledgement], page 38.

Errors

The allowable non-fatal errors are as follows:

`[TCACCES]` The TC user did not have proper permissions for the use of the requested options.

`[TCBADFLAG]`
The flags as specified were incorrect or invalid.

`[TCBADOPT]`
The options as specified were in an incorrect ofrmat, or they contained invalid information.

`[TCOUTSTATE]`
The primitive would place the transaction interface out of state.

`[TCNOTSUPPORT]`
This primitive is not supported by the TC provider.

`[TCSYSERR]`
A system error has occurred and the *Linux* system error is indicated in the primitive.

4.1.3.2 Transaction Options Management Acknowledgement

`TC_OPTMGMT_ACK`

This primitive indicates to the TC user that the options management request has completed.

Format

The format of the message is one `M_PCPROTO` message block structured as follows:

```
typedef struct TC_optmgmt_ack {
    ulong PRIM_type;
    ulong OPT_length;
    ulong OPT_offset;
    ulong MGMT_flags;
} TC_optmgmt_ack_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Indicates the primitive type. Always `TC_OPTMGMT_ACK`.

OPT_length
: Indicates the length of the protocol options associated with the primitive.

OPT_offset
: Indicates the offset from the beginning of the `M_PROTO` message block where the protocol options begin.

MGMT_flags
: Indicates the overall result of the options management operation.

Flags

The flags returned in the *MGMT_flags* represents the single most severe result of the operation. The flags returned will be one of the following values (in order of descending severity):

`TC_NOTSUPPORT`
: This flag indicates that at least one of the options specified in the `TC_OPTMGMT_REQ` primitive was not supported by the trasnaction provider at the current privilege level of the requesting user.

`TC_READONLY`
: This flag indicates that at least one of the options specified in the `TC_OPTMGMT_REQ` primitive is read-only (for the current TRI state). This flag does not apply when the *MGMT_flags* field in the `TC_OPTMGMT_REQ` primitive was `T_DEFAULT`.

`TC_FAILURE`
: This flag indicates that negotiation of at least one of the options specified in the `TC_OPTMGMT_REQ` primitive failed. This is not used for illegal format or values. This flag does not apply when the *MGMT_flags* field in the `TC_OPTMGMT_REQ` primitive was `T_DEFAULT` or `T_CURRENT`.

`TC_PARTSUCCESS`
: This flag indicates that the negotiation of at least one of the options specified in the `TC_OPTMGMT_REQ` primitive was negotiated to a value of lesser quality than the value

requested. This flag only applies when the *MGMT_flags* field of the `TC_OPGMGMT_REQ` primitive was `T_NEGOTIATE`.

`TC_SUCCESS`
: This flag indicates that all of the specified options were negoitated or returned successfully.

Mode

This primitive is valid in Operations Classes 1, 2, 3 and 4.

Originator

This primitive is originated by the TC provider.

Valid State

This primitive is issued in response to a `TC_OPTMGMT_REQ` primitive and is valid in any state.

New State

The new state remains unchanged.

Rules

The following rules apply to the `TC_OPTMGMT_ACK` primitive:

- If the value of *MGMT_flags* in the `TC_OPTMGMT_REQ` primitive is `TC_DEFAULT`, the TC provider should return the default TC provider options without changing the existing options associated with the *Stream*.
- If the value of *MGMT_flags* in the `TC_OPTMGMT_REQ` primitive is `TC_CHECK`, the TC provider should return the options as specified in the `TC_OPTMGMT_REQ` primitive along with the additional flags `TC_SUCCESS` or `TC_FAILURE` which indicate to the TC user whether the specified options are supportable by the TC provider. The TC provider should not change any existing options associated with the *Stream*.
- If the value of *MGMT_flags* in the `TC_OPTMGMT_REQ` is `TC_NEGOTIATE`, the TC provider should set and negotiate the option as specified by the following rules:
  - If the *OPT_length* field of the `TC_OPTMGMT_REQ` is zero ('0'), then the TC provider is to set and return the default options associated with the *Stream* in the `TC_OPTMGMT_ACK` primitive.
  - If options are specified in the `TC_OPTMGMT_REQ` primitive, then the TC provider should negotiate options in the `TC_OPTMGMT_ACK` primitive. It is the TC user's responsibility to check the negotiated options returned in the `TC_OPTMGMT_ACK` primitive and take appropriate action.
- If the value of *MGMT_flags* in the `TC_OPTMGMT_REQ` primitive is `TC_CURRENT`, the TC provider should return the currently effective option values without changing any existing options associated with the *Stream*.

Acknowledgement

If the above rules result in an error condition, the TC provider must issue a `TC_ERROR_ACK` primitive to the TC user specifying the error as defined in the description of the `TC_OPTMGMT_REQ` primitive, Section 4.1.3.1 [Transaction Options Management Request], page 33.

4.1.4 Transaction Error Management

4.1.4.1 Transaction Successful Receipt Acknowledgement

`TC_OK_ACK`

This primitive indicates to the TC user that the previous TC user originated primitive was received successfully by the TC provider. It does not indicate to the TC user any transaction protocol action taken due to issuing the primitive. This may only be initiated as an acknowledgement for those primitives that require one.

Format

The format of the message is one `M_PCPROTO` message block structured as follows:

```
typedef struct TC_ok_ack {
    ulong PRIM_type;
    ulong CORRECT_prim;
} TC_ok_ack_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Indicates the primitive type. Always `TC_OK_ACK`.

CORRECT_prim
: Indicates the successfully received primitive type.

Mode

This primitive is valid in Operations Classes 1, 2, 3 and 4.

Valid State

This primitive is valid in any state where a local acknowledgement requiring `TR_OK_ACK` response is pending.

New State

The new state depends on the current state; see ⟨undefined⟩ [⟨undefined⟩], page ⟨undefined⟩.

Rules

Acknowledgement

4.1.4.2 Transaction Error Acknowledgement

`TC_ERROR_ACK`

This primitive indicates to the TC user that a non-fatal[1] error has occurred in the last TC-user-originated primitive. This may only be initiated as an acknowledgement for those primitives that require one. It also indicates to the TR user that no action was taken on the primitive that cause the error.

Format

The format of the message is one `M_PCPROTO` message block structured as follows:

```
typedef struct TC_error_ack {
    ulong PRIM_type;
    ulong ERROR_prim;
    ulong TRPI_error;
    ulong UNIX_error;
    ulong DIALOG_id;
    ulong INVOKE_id;
} TC_error_ack_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Indicates the primitive type. Always `TC_ERROR_ACK`.

ERROR_prim
: Indicates the primitive type that was in error.

TRPI_error
: Indicates the Transaction Sub-Layer Interface error code.

UNIX_error
: Indicates the UNIX System error code. This field is zero (0) unless the *TRPI_error* is equal to `[TCSYSERR]`.

DIALOG_id
: Indicates the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

INVOKE_id
: Indicates the invoke identifier for the operation for which the primitive caused an error.

Modes

This primitive is valid in Operations Classes 1, 2, 3 and 4.

Originator

This primitive is originated by the TC provider.

[1] For an overview of the error handling capabilities available to the TC provider, see ⟨undefined⟩ [⟨undefined⟩], page ⟨undefined⟩.

Valid State

This primitive is valid in any state where a local acknowledgement is pending and an error has occurred.

New State

The new state is the state that the interface was in before the primitive in error was issued, see ⟨undefined⟩ [⟨undefined⟩], page ⟨undefined⟩.

Rules

This primitive may only be issued as an acknowledgement for those primitives that require one. It also indicates to the user that no action was taken on the primtiive that caused the error.

Errors

The TC provider is allowed to return any of the following TC error codes:

`[TCBADADDR]`
: Indicates that the protocol address as specified in the primitive was of an incorrect format or the address contained illegal information.

`[TCBADOPT]`
: Indicates that the options as specified in the primitive were in an incorrect format, or they contained illegal information.

`[TCBADF]`
: Indicates that the *Stream* queue pointer as specified in the primitive was illegal.

`[TCNOADDR]`
: Indicates that the TC provider could not allocate a protocol address.

`[TCACCES]`
: Indicates that the user did not have proper permissions to use the protocol address or options specified in the primitive.

`[TCOUTSTATE]`
: Indicates that the primitive would place the interface out of state.

`[TCBADSEQ]`
: Indicates that the transaction identifier specified in the primitive was incorrect or illegal.

`[TCBADFLAG]`
: Indicates that the flags specified in the primitive were incorrect or illegal.

`[TCBADDATA]`
: Indicates that the amount of user data specified was illegal.

`[TCSYSERR]`
: Indicates that a system error has occurred and that the UNIX System error is indicated in the primitive.

`[TCADDRBUSY]`
: Indicates that the requested address is already in use.

`[TCRESADDR]`
: Indicates that the TC provider requires the responding *Stream* be bound to the same protocol address as the *Stream* on which the dialogue "begin" indication (see Section 4.2.1.2 [Transaction Begin Indication], page 45) was received.

`[TCNOTSUPPORT]`
Indicates that the TC provider does not support the requested capability.

4.2 Operation Class 1, 2 and 3 Primitives

This section describes the operation class 1, 2, and 3 dialogue handling primitives. Primitives are grouped into phases:

1. Transaction Establishment Phase
   See Section 4.2.1 [Transaction Establishment Phase], page 41.
2. Transaction Data Transfer Phase
   See Section 4.2.2 [Transaction Data Transfer Phase], page 52.
3. Transaction Termination Phase
   See Section 4.2.3 [Transaction Termination Phase], page 54.

4.2.1 Transaction Establishment Phase

The transaction begin service provides means to start a transaction dialogue between two TC-users. This may be accompanied by the transfer of components previously accumulated using the component handling primitives described in Section 4.4 [Component Handling Primitives], page 64.

4.2.1.1 Transaction Begin Request

`TC_BEGIN_REQ`

This primitive requests that the transaction component provider form a transaction dialogue to the specified destination protocol address, from the specified source protocol address, using the specified options. Any components that have been accumulated using the component handling primitives (see Section 4.4 [Component Handling Primitives], page 64), will accompany the primitive.

Format

The format of the message is one `M_PROTO` message block followed by zero or more `M_DATA` message blocks containing raw transaction user information. The `M_PROTO` message block is structured as follows:

```
typedef struct TC_begin_req {
    ulong PRIM_type;       /* Always TC_BEGIN_REQ */
    ulong SRC_length;      /* Source address length */
    ulong SRC_offset;      /* Source address offset */
    ulong DEST_length;     /* Destination address length */
    ulong DEST_offset;     /* Destination address offset */
    ulong OPT_length;      /* Options associated with the primitive */
    ulong OPT_offset;      /* Options associated with the primitive */
    ulong DIALOG_id;       /* Dialog Identifier */
    ulong COMP_flags;      /* For use with ANSI QWP/QWOP */
} TC_begin_req_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Specifies the primitive type. Always `TC_BEGIN_REQ`.

SRC_length
: Specifies the length of the source protocol address associated with the primitive.

SRC_offset
: Specifies the offset from the beginning of the `M_PROTO` message block where the source protocol address begins. Proper alignment of the protocol address in the `M_PROTO` message block is not guaranteed.

DEST_length
: Specifies the length of the destination protocol address associated with the primitive.

DEST_offset
: Specifies the offset from the beginning of the `M_PROTO` message block where the destination protocol address begins. Proper alignment of the protocol address in the `M_PROTO` message block is not guaranteed.

OPT_length
: Specifies the length of the protocol options associated with the primitive.

OPT_offset
: Specifies the offset from the beginning of the `M_PROTO` message block where the protocol options begin.

DIALOG_id
: Specifies the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

COMP_flags
: Specifies additional information about the components. See "Flags" below. Component flags may be provider specific.

Flags

The *COMP_flags* field can contain any of the following flags:

COMPONENTS_PRESENT
: Specifies, when set, that components previously accumulated with the component handling primitives (see Section 4.4 [Component Handling Primitives], page 64) are to be associated with the primitive.

NO_PERMISSION
: Specifies, when set, that the the peer is not granted permission to end the transaction upon the receipt of the corresponding `TC_BEGIN_IND` primitive.

Valid State

This primitive is valid in transaction state `TCS_IDLE`.

New State

The new state for the transaction is `TCS_WACK_CREQ`.

Rules

The following rules apply to the specification of parameters to this primitive:

— When the source address is not specified, *SRC_length* and *SRC_offset* must be specified as zero (0).

— When the *SRC_length* and *SRC_offset* are zero (0), the source protocol address is the local address that is implicitly associated with the access point from the local bind service (see Section 4.1.2.1 [Transaction Bind Request], page 28).

— The destination protocol address must be specified and the TC provider will return error `[TCNOADDR]` if the *DEST_length* and *DEST_offset* are zero (0).

Acknowledgement

This primitive requires the transaction provider to generate one of the following acknowledgements upon receipt of the primitive:

— *Successful Dialogue Establishment:* This is indicated with the `TC_BEGIN_CON` primitive described in Section 4.2.1.1 [Transaction Begin Request], page 42. This results in the `TCS_DATA_XFER` state for the transaction. Successful establishment and tear down can also be indicated with the `TC_END_IND` primitive described in Section 4.2.3.2 [Transaction End Indication], page 55. This results in the `TCS_IDLE` state for the transaction.

— *Unsuccessful Dialogue Establishment:* This is indicated with the `TC_ABORT_IND` primitive described in Section 4.2.3.4 [Transaction Abort Indication], page 57. For example, an dialogue

may be rejected because either the called transaction user cannot be reached, or the transaction provider or the called transaction user did not agree on the specified options. This results in the `TCS_IDLE` state for the transaction.

— *Successful:* Correct acknowledgement of the primitive is indicated with the `TC_OK_ACK` primitive described in Section 4.1.4.1 [Transaction Successful Receipt Acknowledgement], page 37.

— *Non-fatal errors:* These are indicated with the `TC_ERROR_ACK` primitive. The applicable non-fatal errors are defined as follows:

`[TCACCES]` This indicates that the user did not have proper permissions for the use of the requested protocol address or protocol options.

`[TCBADADDR]`
This indicates that the protocol address was in an incorrect format or the address contained illegal information. It is not intended to indicate protocol connection errors, such as an unreachable destination. Those types of errors are indicated with the `TC_ABORT_IND` primitive described in Section 4.2.3.4 [Transaction Abort Indication], page 57.

`[TCBADOPT]`
This indicates that the options were in an incorrect format or they contained illegal information.

`[TCOUTSTATE]`
The primitive would place the transaction interface out of state.

`[TCBADDATA]`
The amount of user data specified was illegal (see Section 4.1.1.2 [Transaction Information Acknowledgement], page 26).

`[TCBADFLAG]`
The flags specified were incorrect, not supported by the provider, or contained illegal information.

`[TCBADSEQ]`
The specified dialogue identifier *DIALOG_id* was incorrect, or contained illegal information. This error would normally occur if the TC user selected a dialogue identifier reserved for the provider (high bit set to 0).

`[TCSYSERR]`
A system error has occured and the UNIX System error is indicated in the primitive.

4.2.1.2 Transaction Begin Indication

`TC_BEGIN_IND`

The transaction indication service primitive indicates that a peer TC user has initiated a transaction dialogue, the source protocol address associated with the peer TC user, the destination address to which the transaction dialogue is initiated, the options for the dialogue.

Format

The format of the message is one `M_PROTO` message block structured as follows:

```
typedef struct TC_begin_ind {
    ulong PRIM_type;      /* Always TC_BEGIN_IND */
    ulong SRC_length;     /* Source address length */
    ulong SRC_offset;     /* Source address offset */
    ulong DEST_length;    /* Destination address length */
    ulong DEST_offset;    /* Destination address offset */
    ulong OPT_length;     /* Options associated with the primitive */
    ulong OPT_offset;     /* Options associated with the primitive */
    ulong DIALOG_id;      /* Dialog Identifier */
    ulong COMP_flags;     /* For use with ANSI QWP/QWOP */
} TC_begin_ind_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Indicates the primitive type. Always `TC_BEGIN_IND`.

SRC_length
: Indicates the length of the source protocol address associated with the primitive.

SRC_offset
: Indicates the offset from the beginning of the `M_PROTO` message block where the source protocol address begins.

DEST_length
: Indicates the length of the destination protocol address associated with the primitive.

DEST_offset
: Indicates the offset from the beginning of the `M_PROTO` message block where the destination protocol address begins.

OPT_length
: Indicates the length of the protocol options associated with the primitive.

OPT_offset
: Indicates the offset from the beginning of the `M_PROTO` message block where the protocol options begin.

DIALOG_id
: Indicates the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

COMP_flags
: Indicates additional information about the components. See "Flags" below. Component flags may be provider specific.

Flags

The *COMP_flags* field can contain any of the following flags:

`TC_COMPONENTS_PRESENT`
: Indicates, when set, that component handling primitives representing the components associated with the begin indication follow this primitive.

`TC_NO_PERMISSION`
: Indicates, when set, that the TC user is not permitted to end the dialogue upon receipt of this primitive, nor when issuing a response.

Valid State

This primitive is valid in transaction state `TCS_IDLE`.

New State

The new state of the transaction is `TCS_WRES_CIND`.

Rules

The following rules apply to the issuance of this primitive by the transaction provider:

- The dialogue identifier provided by the transaction provider uniquely identifies this transaction begin indication within the *Stream* upon which the primitive is issued. This must be a positive, non-zero value. The high bit of the transaction identifier is reserved for exclusive use by the transaction user in generating correlation identifiers.
- It is not necessary to indicate a destination address in *DEST_length*, and *DEST_offset* when the protocol address to which the begin indication corresponds is the same as the local protocol address to which the listening *Stream* is bound. In the case that the destination protocol address is not provided, *DEST_length* and *DEST_offset* must both be set to zero (0). When the local protocol address to which the begin indication corresponds is not the same as the bound address for the *Stream*, the transaction provider must indicate the destination protocol address using *DEST_length* and *DEST_offset*.
- The source protocol address is a mandatory field. The transaction provider must indicate the source protocol address corresponding to the begin indication using the *SRC_length* and *SRC_offset* fields.
- Any indicated options are included in the *OPT_length* and *OPT_offset* fields.
- When the `TC_NO_PERMISSION` flag is set, the TC user must not issue a `TC_END_REQ` primitive in response to this indication.

4.2.1.3 Transaction Begin Response

`TC_BEGIN_RES`

This primitive allows the destination TC user to request that the TC provider accept a previous transaction dialogue begin indication, either on the current *Stream* or on a specified acceptor *Stream*.

Format

The format of the message is one `M_PROTO` message block structured as follows:

```
typedef struct TC_begin_res {
    ulong PRIM_type;      /* Always TC_CONT_REQ */
    ulong SRC_length;     /* Source address length */
    ulong SRC_offset;     /* Source address offset */
    ulong OPT_length;     /* Options associated with the primitive */
    ulong OPT_offset;     /* Options associated with the primitive */
    ulong DIALOG_id;      /* Dialog Identifier */
    ulong COMP_flags;     /* For use with ANSI CWP/CWOP */
} TC_begin_res_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Specifies the primitive type. Always `TC_BEGIN_RES`.

SRC_length
: Specifies the length of the source protocol address associated with the primitive.

SRC_offset
: Specifies the offset from the beginning of the `M_PROTO` message block where the source protocol address begins. Proper alignment of the protocol address in the `M_PROTO` message block is not guaranteed.

OPT_length
: Specifies the length of the protocol options associated with the primitive.

OPT_offset
: Specifies the offset from the beginning of the `M_PROTO` message block where the protocol options begin.

DIALOG_id
: Specifies the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

COMP_flags
: Specifies additional information about the components. See "Flags" below. Component flags may be provider specific.

Flags

The *COMP_flags* field can contain any of the following flags:

`TC_COMPONENTS_PRESENT`
: Specifies, when set, that component handling primitives representing the components associated with the begin indication precede this primitive.

`TC_NO_PERMISSION`
: Specifies, when set, that the TC user peer is not permitted to end the dialogue upon receipt of this primitive, nor when issuing a response.

Valid State

This primitive is valid in transaction state `TCS_WRES_CIND`.

New State

The new state of the transaction is `TCS_DATA_XFER`.

Rules

Acknowledgement

This primitive requires the TC provider to generate one of the following acknowledgements upon receipt of the primitive:

— *Successful:* Correct acknowledgement of the primitive is indicated with the `TC_OK_ACK` primitive described in Section 4.1.4.1 [Transaction Successful Receipt Acknowledgement], page 37.

— *Unsuccessful (Non-fatal errors):* These errors will be indicated with the `TC_ERROR_ACK` primitive described in Section 4.1.4.2 [Transaction Error Acknowledgement], page 38. The allowable errors are as follows:

`[TCBADF]` The token specified is not associated with an open *Stream*.

`[TCBADOPT]`
: The options were in an incorrect format, or they contained illegal information.

`[TCACCES]` The user did not have proper permissions for the use of the responding protocol address or protocol options.

`[TCOUTSTATE]`
: The primitive would place the transaction interface out of state for the indicated transaction.

`[TCBADDATA]`
: The amount of user data specified was outside the range supported by the transaction provider.

`[TCBADFLAG]`
: The flags specified were incorrect, not supported by the provider, or contained illegal information.

`[TCBADSEQ]`
: The specified dialogue identifier *DIALOG_id* was incorrect, or contained illegal information. This error would normally occur if the TC user selected a dialogue identifier reserved for the provider (high bit set to 0).

`[TCSYSERR]`
: A system error occurred and the UNIX System error is indicated in the primitive.

`[TCRESADDR]`
: The transaction provider requires that the responding *Stream* is bound to the same address as the *Stream* on which the transaction dialogue begin indication was received.

`[TCBADADDR]`
: This indicates that the protocol address was in an incorrect format or the protocol address contained illegal information.

4.2.1.4 Transaction Begin Confirm

`TC_BEGIN_CON`

This primitive indicates to the TC user that a dialogue begin request ahs been confirmed on the specified responding address.

Format

This message consists of one `M_PROTO` message block followed by zero or more `M_DATA` message blocks if any TC user data is associated with the primitive. The format of the `M_PROTO` message block is as follows:

```
typedef struct TC_begin_con {
    ulong PRIM_type;      /* Always TC_CONT_IND */
    ulong OPT_length;     /* Options associated with the primitive */
    ulong OPT_offset;     /* Options associated with the primitive */
    ulong DIALOG_id;      /* Dialog Identifier */
    ulong COMP_flags;     /* For use with ANSI CWP/CWOP */
} TC_begin_con_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Indicates the primitive type. Always `TC_BEGIN_CON`.

OPT_length
: Indicates the length of the protocol options associated with the primitive.

OPT_offset
: Indicates the offset from the beginning of the `M_PROTO` message block where the protocol options begin.

DIALOG_id
: Indicates the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

COMP_flags
: Indicates additional information about the components. See "Flags" below. Component flags may be provider specific.

Flags

The *COMP_flags* field can contain any of the following flags:

`TC_COMPONENTS_PRESENT`
: Confirms, when set, that component handling primitives representing the components associated with the begin confirmation precede this primitive.

`TC_NO_PERMISSION`
: Confirms, when set, that the TC user is not permitted to end the dialogue upon receipt of this primitive, nor when issuing a response.

Mode

This primitive is only valid in Operation Classes 1, 2 or 3.

Originator

This primitive is originated by the TC provider.

Valid State

This primitive is only issued by the TC provider in state `TCS_WCON_BREQ` for the indicated *DIALOG_id*.

New State

The new state of the dialogue is `TCS_DATA_XFER`.

Rules

The rules observed by the TC provider when issuing the `TC_BEGIN_CON` primitive are as follows:

— The TC provider maintains a transaction state for each instance of a *DIALOG_id*. This primitive is only issued for a given *DIALOG_id* when the dialogue is in the `TCS_WCON_BREQ` state.

Acknowledgement

This primitive does not require an acknowledgement.

4.2.2 Transaction Data Transfer Phase

The component transfer service primtiives provide for an exchange of component user data known as TSDUs, in either direction or in both directions simultaneously on a transaction dialogue. The transaction service preserves both the sequence and the boundaries of the TSDUs.

4.2.2.1 Transaction Continue Request

`TC_CONT_REQ`

Format

The format of the message is one `M_PROTO` message block structured as follows:

```
typedef struct TC_cont_req {
    ulong PRIM_type;      /* Always TC_CONT_REQ */
    ulong OPT_length;     /* Options associated with the primitive */
    ulong OPT_offset;     /* Options associated with the primitive */
    ulong DIALOG_id;      /* Dialog Identifier */
    ulong COMP_flags;     /* For use with ANSI CWP/CWOP */
} TC_cont_req_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Specifies the primitive type. Always `TC_CONT_REQ`.

OPT_length
: Specifies the length of the protocol options associated with the primitive.

OPT_offset
: Specifies the offset from the beginning of the `M_PROTO` message block where the protocol options begin.

DIALOG_id
: Specifies the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

COMP_flags
: Specifies additional information about the components. See "Flags" below. Component flags may be provider specific.

Flags

Valid State

New State

Rules

Acknowledgement

4.2.2.2 Transaction Continue Indication

`TC_CONT_IND`

Format

The format of the message is one `M_PROTO` message block structured as follows:

```
typedef struct TC_cont_ind {
    ulong PRIM_type;      /* Always TC_CONT_IND */
    ulong OPT_length;     /* Options associated with the primitive */
    ulong OPT_offset;     /* Options associated with the primitive */
    ulong DIALOG_id;      /* Dialog Identifier */
    ulong COMP_flags;     /* For use with ANSI CWP/CWOP */
} TC_cont_ind_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Indicates the primitive type. Always `TC_CONT_IND`.

OPT_length
: Indicates the length of the protocol options associated with the primitive.

OPT_offset
: Indicates the offset from the beginning of the `M_PROTO` message block where the protocol options begin.

DIALOG_id
: Indicates the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

COMP_flags
: Indicates additional information about the components. See "Flags" below. Component flags may be provider specific.

Flags

Valid State

New State

Rules

Acknowledgement

4.2.3 Transaction Termination Phase

4.2.3.1 Transaction End Request

`TC_END_REQ`

Format

The format of the message is one `M_PROTO` message block structured as follows:

```
typedef struct TC_end_req {
    ulong PRIM_type;      /* Always TC_END_REQ */
    ulong OPT_length;     /* Options associated with the primitive */
    ulong OPT_offset;     /* Options associated with the primitive */
    ulong DIALOG_id;      /* Dialog Identifier */
    ulong TERM_scenario;/* Reason for termination */
} TC_end_req_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Specifies the primitive type. Always `TC_END_REQ`.

OPT_length
: Specifies the length of the protocol options associated with the primitive.

OPT_offset
: Specifies the offset from the beginning of the `M_PROTO` message block where the protocol options begin.

DIALOG_id
: Specifies the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

TERM_scenario

Flags

Valid State

New State

Rules

Acknowledgement

4.2.3.2 Transaction End Indication

`TC_END_IND`

Format

The format of the message is one `M_PROTO` message block structured as follows:

```
typedef struct TC_end_ind {
    ulong PRIM_type;      /* Always TC_END_IND */
    ulong OPT_length;     /* Options associated with the primitive */
    ulong OPT_offset;     /* Options associated with the primitive */
    ulong DIALOG_id;      /* Dialog Identifier */
    ulong COMP_flags;     /* Components present flag */
} TC_end_ind_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Indicates the primitive type. Always `TC_END_IND`.

OPT_length
: Indicates the length of the protocol options associated with the primitive.

OPT_offset
: Indicates the offset from the beginning of the `M_PROTO` message block where the protocol options begin.

DIALOG_id
: Indicates the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

COMP_flags
: Indicates additional information about the components. See "Flags" below. Component flags may be provider specific.

Flags

Valid State

New State

Rules

Acknowledgement

4.2.3.3 Transaction Abort Request

`TC_ABORT_REQ`

Format

The format of the message is one `M_PROTO` message block structured as follows:

```
typedef struct TC_abort_req {
    ulong PRIM_type;      /* Always TC_ABORT_REQ */
    ulong OPT_length;     /* Options associated with the primitive */
    ulong OPT_offset;     /* Options associated with the primitive */
    ulong DIALOG_id;      /* Dialog Identifier */
    ulong ABORT_reason;   /* Abort reason */
} TC_abort_req_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Specifies the primitive type. Always `TC_ABORT_REQ`.

OPT_length
: Specifies the length of the protocol options associated with the primitive.

OPT_offset
: Specifies the offset from the beginning of the `M_PROTO` message block where the protocol options begin.

DIALOG_id
: Specifies the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

ABORT_reason

Flags

Valid State

New State

Rules

Acknowledgement

4.2.3.4 Transaction Abort Indication

`TC_ABORT_IND`

Format

The format of the message is one `M_PROTO` message block structured as follows:

```
typedef struct TC_abort_ind {
    ulong PRIM_type;      /* Always TC_ABORT_IND */
    ulong OPT_length;     /* Options associated with the primitive */
    ulong OPT_offset;     /* Options associated with the primitive */
    ulong DIALOG_id;      /* Dialog Identifier */
    ulong ABORT_reason;   /* Abort reason */
    ulong ORIGINATOR;     /* Either User or Provider originated */
} TC_abort_ind_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Indicates the primitive type. Always `TC_ABORT_IND`.

OPT_length
: Indicates the length of the protocol options associated with the primitive.

OPT_offset
: Indicates the offset from the beginning of the `M_PROTO` message block where the protocol options begin.

DIALOG_id
: Indicates the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

ABORT_reason
ORIGINATOR

Flags

Valid State

New State

Rules

Acknowledgement

4.3 Operation Class 4 Primitives

4.3.1 Transaction Phase

4.3.1.1 Transaction Unidirectional Request

`TC_UNI_REQ`

Format

The format of the message is one `M_PROTO` message block, followed by zero or more `M_DATA` message blocks if any components are specified by the TC user. The format of the `M_PROTO` message block is as follows:

```
typedef struct TC_uni_req {
    ulong PRIM_type;      /* Always TC_UNI_REQ */
    ulong SRC_length;     /* Source address length */
    ulong SRC_offset;     /* Source address offset */
    ulong DEST_length;    /* Destination address length */
    ulong DEST_offset;    /* Destination address offset */
    ulong OPT_length;     /* Options associated with the primitive */
    ulong OPT_offset;     /* Options associated with the primitive */
    ulong DIALOG_id;      /* Dialog Identifier */
} TC_uni_req_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Specifies the primitive type. Always `TC_UNI_REQ`.

SRC_length
: Specifies the length of the source protocol address associated with the primitive.

SRC_offset
: Specifies the offset from the beginning of the `M_PROTO` message block where the source protocol address begins.

DEST_length
: Specifies the length of the destination protocol address associated with the primitive.

DEST_offset
: Specifies the offset from the beginning of the `M_PROTO` message block where the destination protocol address begins.

OPT_length
: Specifies the length of the protocol options associated with the primitive.

OPT_offset
: Specifies the offset from the beginning of the `M_PROTO` message block where the protocol options begin.

DIALOG_id
: Specifies the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

Flags

Valid State

New State

Rules

Acknowledgement

4.3.1.2 Transaction Unidirectional Indication

`TC_UNI_IND`

Format

The format of the message is one `M_PROTO` message block, followed by zero or more `M_DATA` message blocks if any components are specified by the TC user. The format of the `M_PROTO` message block is as follows:

```
typedef struct TC_uni_ind {
    ulong PRIM_type;      /* Always TC_UNI_IND */
    ulong SRC_length;     /* Source address length */
    ulong SRC_offset;     /* Source address offset */
    ulong DEST_length;    /* Destination address length */
    ulong DEST_offset;    /* Destination address offset */
    ulong OPT_length;     /* Options associated with the primitive */
    ulong OPT_offset;     /* Options associated with the primitive */
    ulong DIALOG_id;      /* Dialog Identifier */
    ulong COMP_flags;     /* Components preset flag */
} TC_uni_ind_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Indicates the primitive type. Always `TC_UNI_IND`.

SRC_length
: Indicates the length of the source protocol address associated with the primitive.

SRC_offset
: Indicates the offset from the beginning of the `M_PROTO` message block where the source protocol address begins.

DEST_length
: Indicates the length of the destination protocol address associated with the primitive.

DEST_offset
: Indicates the offset from the beginning of the `M_PROTO` message block where the destination protocol address begins.

OPT_length
: Indicates the length of the protocol options associated with the primitive.

OPT_offset
: Indicates the offset from the beginning of the `M_PROTO` message block where the protocol options begin.

DIALOG_id
: Indicates the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

COMP_flags
: Indicates additional information about the components. See “Flags” below. Component flags may be provider specific.

Flags

Valid State

New State

Rules

Acknowledgement

4.3.1.3 Transaction Notice Indication

`TC_NOTICE_IND`

Format

The format of the message is one `M_PROTO` message block, followed by zero or more `M_DATA` message blocks if any components are specified by the TC user. The format of the `M_PROTO` message block is as follows:

```
typedef struct TC_notice_ind {
    ulong PRIM_type;      /* Always TC_NOTICE_IND */
    ulong DIALOG_id;      /* Dialog Identifier */
    ulong REPORT_cause;   /* Report cause */
} TC_notice_ind_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Indicates the primitive type. Always `TC_NOTICE_IND`.

DIALOG_id
: Indicates the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

REPORT_cause

Flags

Valid State

New State

Rules

Acknowledgement

4.4 Component Handling Primitives

4.4.1 Invocation of an Operation

4.4.1.1 Invoke Request

`TC_INVOKE_REQ`

Format

The format of the message is one `M_PROTO` message block, followed by zero or more `M_DATA` message blocks if any components are specified by the TC user. The format of the `M_PROTO` message block is as follows:

```
typedef struct TC_invoke_req {
    ulong PRIM_type;               /* Always TC_INVOKE_REQ */
    ulong DIALOG_id;               /* Dialog identifier */
    ulong PROTOCOL_class; /* Application protocol class */
    ulong INVOKE_id;               /* Invoke Identifier */
    ulong LINKED_id;               /* Linked Invoke Identifier */
    ulong OPERATION;               /* Requested operation to invoke */
    ulong MORE_flag;               /* Not last */
    ulong TIMEOUT;                 /* Timeout */
} TC_invoke_req_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Specifies the primitive type. Always `TC_INVOKE_REQ`.

DIALOG_id
: Specifies the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

PROTOCOL_class
INVOKE_id
LINKED_id
OPERATION
MORE_flag
TIMEOUT

Flags

Valid State

New State

Rules

Acknowledgement

4.4.1.2 Invoke Indication

`TC_INVOKE_IND`

Format

The format of the message is one `M_PROTO` message block, followed by zero or more `M_DATA` message blocks if any components are specified by the TC user. The format of the `M_PROTO` message block is as follows:

```
typedef struct TC_invoke_ind {
    ulong PRIM_type;      /* Always TC_INVOKE_IND */
    ulong DIALOG_id;      /* Dialog identifier */
    ulong OP_class;       /* Application operation class */
    ulong INVOKE_id;      /* Invoke Identifier */
    ulong LINKED_id;      /* Linked Invoke Identifier */
    ulong OPERATION;      /* Requested operation to invoke */
    ulong MORE_flag;      /* Not last */
} TC_invoke_ind_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Indicates the primitive type. Always `TC_INVOKE_IND`.

DIALOG_id
: Indicates the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

OP_class

INVOKE_id
LINKED_id
OPERATION
MORE_flag

Flags

Valid State

New State

Rules

Acknowledgement

4.4.2 Result of a Successful Operation

4.4.2.1 Return Result Request

`TC_RESULT_REQ`

Format

The format of the message is one `M_PROTO` message block, followed by zero or more `M_DATA` message blocks if any components are specified by the TC user. The format of the `M_PROTO` message block is as follows:

```
typedef struct TC_result_req {
    ulong PRIM_type;      /* Always TC_RESULT_REQ */
    ulong DIALOG_id;      /* Dialog Identifier */
    ulong INVOKE_id;      /* Invoke Identifier */
    ulong OPERATION;      /* Requested operation result */
    ulong MORE_flag;      /* Not last */
} TC_result_req_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Specifies the primitive type. Always `TC_RESULT_REQ`.

DIALOG_id
: Specifies the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

INVOKE_id
OPERATION
MORE_flag

Flags

Valid State

New State

Rules

Acknowledgement

4.4.2.2 Return Result Indication

`TC_RESULT_IND`

Format

The format of the message is one `M_PROTO` message block, followed by zero or more `M_DATA` message blocks if any components are specified by the TC user. The format of the `M_PROTO` message block is as follows:

```
typedef struct TC_result_ind {
    ulong PRIM_type;      /* Always TC_RESULT_IND */
    ulong DIALOG_id;      /* Dialog Identifier */
    ulong INVOKE_id;      /* Invoke Identifier */
    ulong OPERATION;      /* Requested operation result */
    ulong MORE_flag;      /* Not last */
} TC_result_ind_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Indicates the primitive type. Always `TC_RESULT_IND`.

DIALOG_id
: Indicates the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

INVOKE_id
OPERATION
MORE_flag

Flags

Valid State

New State

Rules

Acknowledgement

4.4.3 Error Reply to an Invoked Operation

4.4.3.1 Return Error Request

`TC_ERROR_REQ`

Format

The format of the message is one `M_PROTO` message block, followed by zero or more `M_DATA` message blocks if any components are specified by the TC user. The format of the `M_PROTO` message block is as follows:

```
typedef struct TC_error_req {
    ulong PRIM_type;       /* Always TC_ERROR_REQ */
    ulong DIALOG_id;       /* Dialog Identifier */
    ulong INVOKE_id;       /* Invoke Identifier */
    ulong ERROR_code;      /* Error code */
    ulong MORE_flag;       /* Not last */
} TC_error_req_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Specifies the primitive type. Always `TC_RESULT_REQ`.

DIALOG_id
: Specifies the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

INVOKE_id
ERROR_code
MORE_flag

Flags

Valid State

New State

Rules

Acknowledgement

4.4.3.2 Return Error Indication

`TC_ERROR_IND`

Format

The format of the message is one `M_PROTO` message block, followed by zero or more `M_DATA` message blocks if any components are specified by the TC user. The format of the `M_PROTO` message block is as follows:

```
typedef struct TC_error_ind {
    ulong PRIM_type;      /* Always TC_ERROR_IND */
    ulong DIALOG_id;      /* Dialog Identifier */
    ulong INVOKE_id;      /* Invoke Identifier */
    ulong ERROR_code;     /* Error code */
} TC_error_ind_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Indicates the primitive type. Always `TC_ERROR_IND`.

DIALOG_id
: Indicates the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

INVOKE_id
ERROR_code

Flags

Valid State

New State

Rules

Acknowledgement

4.4.4 Termination of an Operation Invocation

4.4.4.1 Cancel Request

`TC_CANCEL_REQ`

Format

The format of the message is one `M_PROTO` message block, followed by zero or more `M_DATA` message blocks if any components are specified by the TC user. The format of the `M_PROTO` message block is as follows:

```
typedef struct TC_cancel_req {
    ulong PRIM_type;      /* Always TC_CANCEL_REQ */
    ulong DIALOG_id;      /* Dialog Identifier */
    ulong INVOKE_id;      /* Invoke identifier */
} TC_cancel_req_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Specifies the primitive type. Always `TC_CANCEL_REQ`.

DIALOG_id
: Specifies the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

INVOKE_id

Flags

Valid State

New State

Rules

Acknowledgement

4.4.4.2 Cancel Indication

`TC_CANCEL_IND`

Format

The format of the message is one `M_PROTO` message block, followed by zero or more `M_DATA` message blocks if any components are specified by the TC user. The format of the `M_PROTO` message block is as follows:

```
typedef struct TC_cancel_ind {
    ulong PRIM_type;      /* Always TC_CANCEL_IND */
    ulong DIALOG_id;      /* Dialog Identifier */
    ulong INVOKE_id;      /* Invoke identifier */
} TC_cancel_ind_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Indicates the primitive type. Always `TC_CANCEL_IND`.

DIALOG_id
: Indicates the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

INVOKE_id

Flags

Valid State

New State

Rules

Acknowledgement

4.4.5 Rejection of a Component

4.4.5.1 Reject Request

`TC_REJECT_REQ`

Format

The format of the message is one `M_PROTO` message block, followed by zero or more `M_DATA` message blocks if any components are specified by the TC user. The format of the `M_PROTO` message block is as follows:

```
typedef struct TC_reject_req {
    ulong PRIM_type;       /* Always TC_REJECT_REQ */
    ulong DIALOG_id;       /* Dialog Identifier */
    ulong INVOKE_id;       /* Invoke identifier */
    ulong PROBLEM_code;    /* Problem code */
} TC_reject_req_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Specifies the primitive type. Always `TC_REJECT_REQ`.

DIALOG_id
: Specifies the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

INVOKE_id
PROBLEM_code

Flags

Valid State

New State

Rules

Acknowledgement

4.4.5.2 Reject Indication

`TC_REJECT_IND`

Format

The format of the message is one `M_PROTO` message block, followed by zero or more `M_DATA` message blocks if any components are specified by the TC user. The format of the `M_PROTO` message block is as follows:

```
typedef struct TC_reject_ind {
    ulong PRIM_type;      /* Always TC_REJECT_IND */
    ulong DIALOG_id;      /* Dialog Identifier */
    ulong INVOKE_id;      /* Invoke identifier */
    ulong ORIGINATOR;     /* Either User, Local or Remote */
    ulong PROBLEM_code;   /* Problem code */
} TC_reject_ind_t;
```

Parameters

The primitive has the following arguments:

PRIM_type
: Indicates the primitive type. Always `TC_REJECT_IND`.

DIALOG_id
: Indicates the dialogue identifier which uniquely identifies this transaction dialogue within the *Stream*. Dialogue identifiers assigned by the component user must have the high bit set to one (1); those assigned by the component provider have the high bit set to zero (0).

INVOKE_id
ORIGINATOR
PROBLEM_code

Flags

Valid State

New State

Rules

Acknowledgement

5 TCI Header File

```
#ifndef __SS7_TC_H__
#define __SS7_TC_H__

/*
 *  Primitive definitions for TC-Users and TC-Providers.
 */

#define TC_INFO_REQ              0      /* Information request */
#define TC_BIND_REQ              1      /* Bind to network address */
#define TC_UNBIND_REQ            2      /* Unbind from network address */
#define TC_SUBS_BIND_REQ         3      /* Subsequent bind to network address */
#define TC_SUBS_UNBIND_REQ       4      /* Subsequent unbind from network address */
#define TC_OPTMGMT_REQ           5      /* Options management */
#define TC_UNI_REQ               6      /* Unidirectional request */
#define TC_BEGIN_REQ             7      /* Begin transaction request */
#define TC_BEGIN_RES             8      /* Begin transaction response */
#define TC_CONT_REQ              9      /* Continue transaction request */
#define TC_END_REQ              10      /* End transaction request */
#define TC_ABORT_REQ            11      /* User abort request */

#define TC_INFO_ACK             12      /* Information acknowledgement */
#define TC_BIND_ACK             13      /* Bound to network address */
#define TC_SUBS_BIND_ACK        14      /* Bound to network address */
#define TC_OK_ACK               15      /* Success acknowledgement */
#define TC_ERROR_ACK            16      /* Error acknowledgement */
#define TC_OPTMGMT_ACK          17      /* Optionas management acknowledgement */
#define TC_UNI_IND              18      /* Unidirectional indication */
#define TC_BEGIN_IND            19      /* Begin transaction indication */
#define TC_BEGIN_CON            20      /* Begin transaction confirmation-Continue */
#define TC_CONT_IND             21      /* Continue transaction indication */
#define TC_END_IND              22      /* End transaction indication */
#define TC_ABORT_IND            23      /* TC-User abort indication */
#define TC_NOTICE_IND           24      /* Network Service Provider notice */

/*
 *  Additional primitives for component handling.
 */
#define TC_INVOKE_REQ           48      /* Invocation of an operation */
#define TC_RESULT_REQ           49      /* Result of a successful operation */
#define TC_ERROR_REQ            50      /* Error reply to an invoked operation */
#define TC_CANCEL_REQ           51      /* Termination of an operation invocation */
#define TC_REJECT_REQ           52      /* Rejection of a component */

#define TC_INVOKE_IND           53      /* Invocation of an operation */
#define TC_RESULT_IND           54      /* Result of a successful operation */
#define TC_ERROR_IND            55      /* Error reply to an invoked operation */
#define TC_CANCEL_IND           56      /* Termination of an operation invocation */
#define TC_REJECT_IND           57      /* Rejection of a component */

/*
 *  TCAP managment primitives.
 */
#define TC_COORD_REQ            35      /* coordinated withdrawal request */
```

```
#define TC_COORD_RES            36      /* coordinated withdrawal response */
#define TC_COORD_IND            37      /* coordinated withdrawal indication */
#define TC_COORD_CON            38      /* coordinated withdrawal confirmation */
#define TC_STATE_REQ            39      /* subsystem state request */
#define TC_STATE_IND            40      /* subsystem state indication */
#define TC_PCSTATE_IND          41      /* pointcode state indication */
#define TC_TRAFFIC_IND          42      /* traffic mix indication */

#define TC_QOS_SEL1             0x0701

typedef struct {
        t_scalar_t type;                /* Always TC_QOS_SEL1 */
        t_scalar_t flags;               /* Return option */
        t_scalar_t seq_ctrl;            /* Sequence Control */
        t_scalar_t priority;            /* Message priority */
} TC_qos_sel1_t;

/*
 * TCPI interface states
 */
#define TCS_UNBND                0      /* TC user not bound to network address */
#define TCS_WACK_BREQ            1      /* Awaiting acknowledgement of N_BIND_REQ */
#define TCS_WACK_UREQ            2      /* Pending acknowledgement for N_UNBIND_REQ */
#define TCS_IDLE                 3      /* Idle, no connection */
#define TCS_WACK_OPTREQ          4      /* Pending acknowledgement of N_OPTMGMT_REQ */
#define TCS_WACK_RRES            5      /* Pending acknowledgement of N_RESET_RES */
#define TCS_WCON_CREQ            6      /* Pending confirmation of N_CONN_REQ */
#define TCS_WRES_CIND            7      /* Pending response of N_CONN_REQ */
#define TCS_WACK_CRES            8      /* Pending acknowledgement of N_CONN_RES */
#define TCS_DATA_XFER            9      /* Connection-mode data transfer */
#define TCS_WCON_RREQ           10      /* Pending confirmation of N_RESET_REQ */
#define TCS_WRES_RIND           11      /* Pending response of N_RESET_IND */
#define TCS_WACK_DREQ6          12      /* Waiting ack of N_DISCON_REQ */
#define TCS_WACK_DREQ7          13      /* Waiting ack of N_DISCON_REQ */
#define TCS_WACK_DREQ9          14      /* Waiting ack of N_DISCON_REQ */
#define TCS_WACK_DREQ10         15      /* Waiting ack of N_DISCON_REQ */
#define TCS_WACK_DREQ11         16      /* Waiting ack of N_DISCON_REQ */

#define TCS_NOSTATES    17

/*
 * TC_ERROR_ACK error return code values
 */
#define TCBADADDR                1      /* Incorrect address format/illegal address information */
#define TCBADOPT                 2      /* Options in incorrect format or contain illegal information */
#define TCACCESS                 3      /* User did not have proper permissions */
#define TCNOADDR                 5      /* TC Provider could not allocate address */
#define TCOUTSTATE               6      /* Primitive was issues in wrong sequence */
#define TCBADSEQ                 7      /* Sequence number in primitive was incorrect/illegal */
#define TCSYSERR                 8      /* UNIX system error occurred */
#define TCBADDATA               10      /* User data spec.  outside range supported by TC provider */
#define TCBADFLAG               16      /* Flags specified in primitive were illegal/incorrect */
#define TCNOTSUPPORT            18      /* Primitive type not supported by the TC provider */
#define TCBOUND                 19      /* Illegal second attempt to bind listener or default listener */
#define TCBADQOSPARAM           20      /* QOS values specified are outside the range supported by the TC provi
#define TCBADQOSTYPE            21      /* QOS structure type specified is not supported by the TC provider */
```

```
#define TCBADTOKEN              22      /* Token used is not associated with an open stream */
#define TCNOPROTOID             23      /* Protocol id could not be allocated */

/*
 *  TC_ABORT_IND originator
 */
#define TC_PROVIDER             0x0001
#define TC_USER                 0x0002

/*
 *  TC_ABORT abort reasons
 */
/* Application-Wide ITU Q.773 abort reasons */
#define TCAP_AAB_UNREC_MSG_TYPE                 0x0a00  /* unrecognized message type */
#define TCAP_AAB_UNREC_TRANS_ID                 0x0a01  /* unrecognized transaction id */
#define TCAP_AAB_BAD_XACT_PORTION               0x0a02  /* badly formatted transaction portion */
#define TCAP_AAB_INCORRECT_XACT_PORTION         0x0a03  /* incorrect transaction portion */
#define TCAP_AAB_RESOURCE_LIMITATION            0x0a04  /* resource limitation */
/* Private-TCAP ANSI T1.114 abort reasons */
#define TCAP_PAB_UNREC_PKG_TYPE                 0x1701  /* unrecognized package type */
#define TCAP_PAB_INCORRECT_XACT_PORTION         0x1702  /* incorrect transaction portion */
#define TCAP_PAB_BAD_XACT_PORTION               0x1703  /* badly structured transaction portion */
#define TCAP_PAB_UNASSIGNED_RESP_TRANS_ID       0x1704  /* unassigned responding transaction id */
#define TCAP_PAB_PERM_TO_RELEASE_PROB           0x1705  /* permission to release problem */
#define TCAP_PAB_RESOURCE_UNAVAIL               0x1706  /* resource unavailable */
#define TCAP_PAB_UNREC_DIALOG_PORTION_ID        0x1707  /* unrecognized dialogue portion id */
#define TCAP_PAB_BAD_DIALOG_PORTION             0x1708  /* badly structured dialogue portion */
#define TCAP_PAB_MISSING_DIALOG_PORTION         0x1709  /* missing dialogue portion */
#define TCAP_PAB_INCONSIST_DIALOG_PORTION       0x170a  /* inconsistent dialogue portion */

/*
 *  TC_REJECT problem codes
 */
/* Application Wide ITU Q.773 reject problem codes */
#define TCAP_ARJ_GN_UNRECOGNIZED_COMPONENT      0x0000  /* unrecognized component */
#define TCAP_ARJ_GN_MISTYPED_COMPONENT          0x0001  /* mistyped component */
#define TCAP_ARJ_GN_BADLY_STRUCTURED_COMPONENT  0x0002  /* badly structured component */
#define TCAP_ARJ_IN_DUPLICATE_INVOKE_ID         0x0100  /* duplicate invoke id */
#define TCAP_ARJ_IN_UNRECOGNIZED_OPERATION      0x0101  /* unrecognized operation */
#define TCAP_ARJ_IN_MISTYPED_PARAMETER          0x0102  /* mistyped parameter */
#define TCAP_ARJ_IN_RESOURCE_LIMITATION         0x0103  /* resource limitation */
#define TCAP_ARJ_IN_INITIATING_RELEASE          0x0104  /* initiating release */
#define TCAP_ARJ_IN_UNRECOGNIZED_LINKED_ID      0x0105  /* unrecognized linked id */
#define TCAP_ARJ_IN_LINKED_RESPONSE_EXPECTED    0x0106  /* linked response expected */
#define TCAP_ARJ_IN_UNEXPECTED_LINKED_OPERATION 0x0107  /* unexpected linked operation */
#define TCAP_ARJ_RR_UNRECOGNIZED_INVOKE_ID      0x0200  /* urecognized invoke id */
#define TCAP_ARJ_RR_RETURN_RESULT_UNEXPECTED    0x0201  /* return result unexpected */
#define TCAP_ARJ_RR_MISTYPED_PARAMETER          0x0202  /* mistyped parameter */
#define TCAP_ARJ_RE_UNRECOGNIZED_INVOKE_ID      0x0300  /* unrecognized invoke id */
#define TCAP_ARJ_RE_RETURN_ERROR_UNEXPECTED     0x0301  /* return error unexpected */
#define TCAP_ARJ_RE_UNRECOGNIZED_ERROR          0x0302  /* unrecognized error */
#define TCAP_ARJ_RE_UNEXPECTED_ERROR            0x0303  /* unexpected error */
#define TCAP_ARJ_RE_MISTYPED_PARAMETER          0x0304  /* mistyped parameter */
/* Private TCAP ANSI T1.114 reject problem codes */
#define TCAP_PRJ_GN_UNRECOGNIZED_COMPONENT_TYPE 0x0101  /* unrecognized component type */
#define TCAP_PRJ_GN_INCORRECT_COMPONENT_PORTION 0x0102  /* incorrect component portion */
```

```
#define TCAP_PRJ_GN_BADLY_STRUCTURED_COMP_PRTN  0x0103  /* badly structure component portion */
#define TCAP_PRJ_GN_INCORRECT_COMPONENT_CODING  0x0104  /* incorrect component coding */
#define TCAP_PRJ_IN_DUPLICATE_INVOCATION        0x0201  /* duplicate invocation */
#define TCAP_PRJ_IN_UNRECOGNIZED_OPERATION      0x0202  /* unrecognized operation */
#define TCAP_PRJ_IN_INCORRECT_PARAMETER         0x0203  /* incorrect parameter */
#define TCAP_PRJ_IN_UNRECOGNIZED_CORRELATION_ID 0x0204  /* unrecognized correlation id */
#define TCAP_PRJ_RR_UNRECOGNIZED_CORRELATION_ID 0x0301  /* unrecognized correlation id */
#define TCAP_PRJ_RR_UNEXPECTED_RETURN_RESULT    0x0302  /* unexpected return result */
#define TCAP_PRJ_RR_INCORRECT_PARAMETER         0x0303  /* incorrect parameter */
#define TCAP_PRJ_RE_UNRECOGNIZED_CORRELATION_ID 0x0401  /* unrecognized correlation id */
#define TCAP_PRJ_RE_UNEXPECTED_RETURN_ERROR     0x0402  /* unexpected return error */
#define TCAP_PRJ_RE_UNRECOGNIZED_ERROR          0x0403  /* unrecognized error */
#define TCAP_PRJ_RE_UNEXPECTED_ERROR            0x0404  /* unexpected error */
#define TCAP_PRJ_RE_INCORRECT_PARAMETER         0x0405  /* incorrect parameter */

/*
 *  TC_INFO_REQ
 */
typedef struct TC_info_req {
        t_scalar_t PRIM_type;           /* Always TC_INFO_REQ */
} TC_info_req_t;

/*
 *  TC_INFO_ACK
 */
typedef struct TC_info_ack {
        t_scalar_t PRIM_type;           /* always TC_INFO_ACK */
        t_scalar_t TSDU_size;           /* maximum TSDU size */
        t_scalar_t ETSDU_size;          /* maximum ETSDU size */
        t_scalar_t CDATA_size;          /* connect data size */
        t_scalar_t DDATA_size;          /* disconnect data size */
        t_scalar_t ADDR_size;           /* maximum address size */
        t_scalar_t OPT_size;            /* maximum options size */
        t_scalar_t TIDU_size;           /* transaction interface data size */
        t_scalar_t SERV_type;           /* service type */
        t_scalar_t CURRENT_state;       /* current state */
        t_scalar_t PROVIDER_flag;       /* provider flags */
        t_scalar_t TCI_version;         /* TCI version */
} TC_info_ack_t;

/*
 *  TC_BIND_REQ
 */
typedef struct TC_bind_req {
        t_scalar_t PRIM_type;           /* always TC_BIND_REQ */
        t_scalar_t ADDR_length;         /* address length */
        t_scalar_t ADDR_offset;         /* address offset */
        t_scalar_t XACT_number;         /* maximum outstanding transaction reqs.  */
        t_scalar_t BIND_flags;          /* bind flags */
} TC_bind_req_t;

/*
 *  TC_BIND_ACK:- one M_PCPROTO message block.
 */
typedef struct TC_bind_ack {
        t_scalar_t PRIM_type;           /* always TC_BIND_ACK */
```

```
        t_scalar_t ADDR_length;
        t_scalar_t ADDR_offset;
        t_scalar_t XACT_number;
        t_scalar_t TOKEN_value;
} TC_bind_ack_t;

/*
 *  TC_SUBS_BIND_REQ
 */
typedef struct TC_subs_bind_req {
        t_scalar_t PRIM_type;           /* always TC_SUBS_BIND_REQ */
} TC_subs_bind_req_t;

/*
 *  TC_SUBS_BIND_ACK:- one M_PCPROTO message block.
 */
typedef struct TC_subs_bind_ack {
        t_scalar_t PRIM_type;           /* always TC_SUBS_BIND_ACK */
} TC_subs_bind_ack_t;

/*
 *  TC_SUBS_UNBIND_REQ
 */
typedef struct TC_subs_unbind_req {
        t_scalar_t PRIM_type;           /* always TC_SUBS_UNBIND_REQ */
} TC_subs_unbind_req_t;

/*
 *  TC_UNBIND_REQ
 */
typedef struct TC_unbind_req {
        t_scalar_t PRIM_type;           /* Always TC_UNBIND_REQ */
} TC_unbind_req_t;

/*
 *  TC_OK_ACK:- one M_PCPROTO message block.
 */
typedef struct TC_ok_ack {
        t_scalar_t PRIM_type;           /* Always TC_OK_ACK */
        t_scalar_t CORRECT_prim;        /* correct primitive */
} TC_ok_ack_t;

/*
 *  TC_ERROR_ACK:- one M_PCPROTO message block.
 */
typedef struct TC_error_ack {
        t_scalar_t PRIM_type;           /* always TC_ERROR_ACK */
        t_scalar_t ERROR_prim;
        t_scalar_t TRPI_error;
        t_scalar_t UNIX_error;
        t_scalar_t DIALOG_id;
        t_scalar_t INVOKE_id;
} TC_error_ack_t;

/*
 *  TC_OPTMGMT_REQ
```

```
 */
typedef struct TC_optmgmt_req {
        t_scalar_t PRIM_type;           /* always TC_OPTMGMT_REQ */
        t_scalar_t OPT_length;          /* Options length */
        t_scalar_t OPT_offset;          /* Options offset */
        t_scalar_t MGMT_flags;
} TC_optmgmt_req_t;

/*
 *  TC_OPTMGMT_ACK:- one M_PCPROTO message block.
 */
typedef struct TC_optmgmt_ack {
        t_scalar_t PRIM_type;           /* always TC_OPTMGMT_ACK */
        t_scalar_t OPT_length;          /* Options length */
        t_scalar_t OPT_offset;          /* Options offset */
        t_scalar_t MGMT_flags;
} TC_optmgmt_ack_t;

/*
 *  TC_UNI_REQ,  Send unidirctional message.    One M_PROTO block followed by one or more M_DATA
 *  blocks containing User Information.    Components to be delivered in the unstructured dialog must
 *  have been previously provided with the same Dialog Id and using the component handling request
 *  primitives.    An Application Context is required if there is User Information in attached M_DATA
 *  blocks.
 *
 *  Note:      Source Address may be implicitly associatedw tih the access point at which the primitive
 *          is being issued.
 *
 *  Note:      Dialog identifier has only local significance and is used between the local TC-User and
 *          TC-Provider to refer to a dialog.
 */
typedef struct TC_uni_req {
        t_scalar_t PRIM_type;           /* Always TC_UNI_REQ */
        t_scalar_t SRC_length;          /* Source address length */
        t_scalar_t SRC_offset;          /* Source address offset */
        t_scalar_t DEST_length;         /* Destination address length */
        t_scalar_t DEST_offset;         /* Destination address offset */
        t_scalar_t OPT_length;          /* Options associated with the primitive */
        t_scalar_t OPT_offset;          /* Options associated wtih the primitive */
        t_scalar_t DIALOG_id;           /* Dialog Identifier */
} TC_uni_req_t;

/*
 *  TC_UNI_RES,  Received unidirectional message.    One M_PROTO block followed by one or more M_DATA
 *  blocks containing User Information.    Components to be delivered from the unstructured dialog
 *  will be indicated using the component handling indication primitives.    An Application Context
 *  will be present where there is User Information in attached M_DATA blocks.
 *
 *  Note:      When QOS is provided by SCCP, QOS must be passed up to the TC-User.
 *
 *  Note:      When Application Context is provided in the corresponding message, it must be passed up
 *          in the indication.
 */
typedef struct TC_uni_ind {
        t_scalar_t PRIM_type;           /* Always TC_UNI_IND */
        t_scalar_t SRC_length;          /* Source address length */
```

```
        t_scalar_t SRC_offset;          /* Source address offset */
        t_scalar_t DEST_length;         /* Destination address length */
        t_scalar_t DEST_offset;         /* Destination address offset */
        t_scalar_t OPT_length;          /* Options associated with the primitive */
        t_scalar_t OPT_offset;          /* Options associated wtih the primitive */
        t_scalar_t DIALOG_id;           /* Dialog Identifier */
        t_scalar_t COMP_flags;          /* Components preset flag */
} TC_uni_ind_t;

/*
 *  TC_BEGIN_REQ.  Requests the opening of a dialog.    One M_PROTO block followed by one or more
 *  M_DATA blocks containing User Information.    Components to be delivered in the structured dialog
 *  must have been previously provided wtih the same Dialog Id and using the component handling
 *  request primitives.    An Application Context is required if there is User Information in attached
 *  M_DATA blocks.
 *
 *  Also T_QUERY_REQ for ANSI.
 */
typedef struct TC_begin_req {
        t_scalar_t PRIM_type;           /* Always TC_BEGIN_REQ */
        t_scalar_t SRC_length;          /* Source address length */
        t_scalar_t SRC_offset;          /* Source address offset */
        t_scalar_t DEST_length;         /* Destination address length */
        t_scalar_t DEST_offset;         /* Destination address offset */
        t_scalar_t OPT_length;          /* Options associated with the primitive */
        t_scalar_t OPT_offset;          /* Options associated wtih the primitive */
        t_scalar_t DIALOG_id;           /* Dialog Identifier */
        t_scalar_t COMP_flags;          /* For use with ANSI QWP/QWOP */
} TC_begin_req_t;

typedef struct TC_begin_req TC_query_req_t;

/*
 *  TC_BEGIN_IND.  Indicates the opening of a dialog.    One M_PROTO block followed by one or more
 *  M_DATA blocks containing User Information.    Components to be delivered in the structured dialog
 *  will be subsequently indicated with the same Dialog Id and using the component handling
 *  indication primitives.    An Application Context is present if there is User Information in
 *  attached M_DATA blocks.
 *
 *  Also T_QUERY_IND for ANSI.
 */
typedef struct TC_begin_ind {
        t_scalar_t PRIM_type;           /* Always TC_BEGIN_IND */
        t_scalar_t SRC_length;          /* Source address length */
        t_scalar_t SRC_offset;          /* Source address offset */
        t_scalar_t DEST_length;         /* Destination address length */
        t_scalar_t DEST_offset;         /* Destination address offset */
        t_scalar_t OPT_length;          /* Options associated with the primitive */
        t_scalar_t OPT_offset;          /* Options associated wtih the primitive */
        t_scalar_t DIALOG_id;           /* Dialog Identifier */
        t_scalar_t COMP_flags;          /* For use with ANSI QWP/QWOP */
} TC_begin_ind_t;

typedef struct TC_begin_ind TC_query_ind_t;

/*
```

```
 *  TC_END_REQ.
 *
 *  Also TC_RESP_REQ for ANSI.
 */
typedef struct TC_end_req {
        t_scalar_t PRIM_type;           /* Always TC_END_REQ */
        t_scalar_t OPT_length;          /* Options associated with the primitive */
        t_scalar_t OPT_offset;          /* Options associated wtih the primitive */
        t_scalar_t DIALOG_id;           /* Dialog Identifier */
        t_scalar_t TERM_scenario;       /* Reason for termination */
} TC_end_req_t;

typedef struct TC_end_req TC_resp_req_t;

/*
 *  TC_END_IND.
 *
 *  Also TC_RESP_IND for ANSI.
 */
typedef struct TC_end_ind {
        t_scalar_t PRIM_type;           /* Always TC_END_IND */
        t_scalar_t OPT_length;          /* Options associated with the primitive */
        t_scalar_t OPT_offset;          /* Options associated wtih the primitive */
        t_scalar_t DIALOG_id;           /* Dialog Identifier */
        t_scalar_t COMP_flags;          /* Components present flag */
} TC_end_ind_t;

typedef struct TC_end_ind TC_resp_ind_t;

/*
 *  TC_CONT_REQ.  The first TC_CONT_REQ after a TC_BEGIN_IND requests that the dialog be confirmed
 *  and may contain the Source address and Application Context parameters.    Once these have been
 *  provided on the first TC_CONT_REQ, they are in place for the remainder of the dialog.
 *  Subsequent TC_CONT_REQ primitives do not contain the SRC and CONTEXT parameters.
 *
 *  Also TC_CONV_REQ for ANSI.
 */
typedef struct TC_begin_res {
        t_scalar_t PRIM_type;           /* Always TC_CONT_REQ */
        t_scalar_t SRC_length;          /* Source address length */
        t_scalar_t SRC_offset;          /* Source address offset */
        t_scalar_t OPT_length;          /* Options associated with the primitive */
        t_scalar_t OPT_offset;          /* Options associated wtih the primitive */
        t_scalar_t DIALOG_id;           /* Dialog Identifier */
        t_scalar_t COMP_flags;          /* For use with ANSI CWP/CWOP */
        t_scalar_t ACCEPTOR_id;         /* Token of accepting stream */
} TC_begin_res_t;

typedef struct TC_cont_req {
        t_scalar_t PRIM_type;           /* Always TC_CONT_REQ */
        t_scalar_t OPT_length;          /* Options associated with the primitive */
        t_scalar_t OPT_offset;          /* Options associated wtih the primitive */
        t_scalar_t DIALOG_id;           /* Dialog Identifier */
        t_scalar_t COMP_flags;          /* For use with ANSI CWP/CWOP */
} TC_cont_req_t;
```

```
typedef struct TC_cont_req TC_conv_req_t;

/*
 *  TC_CONT_IND.  The first TC_CONT_IND after a TC_BEGIN_REQ indicates that the dialog is confirmed
 *  but may contain the Source address and Application Context parameters.    Once these have been
 *  provided on the first TC_CONT_IND, they are in place for the remainder of the dialog.
 *  Subsequent TC_CONT_IND primitives will not contain the SRC and CONTEXT parameters.
 *
 *  Also TC_CONV_IND for ASNI.
 */
typedef struct TC_begin_con {
        t_scalar_t PRIM_type;           /* Always TC_BEGIN_CON */
        t_scalar_t OPT_length;          /* Options associated with the primitive */
        t_scalar_t OPT_offset;          /* Options associated wtih the primitive */
        t_scalar_t DIALOG_id;           /* Dialog Identifier */
        t_scalar_t COMP_flags;          /* For use with ANSI CWP/CWOP */
} TC_begin_con_t;

typedef struct TC_cont_ind {
        t_scalar_t PRIM_type;           /* Always TC_CONT_IND */
        t_scalar_t OPT_length;          /* Options associated with the primitive */
        t_scalar_t OPT_offset;          /* Options associated wtih the primitive */
        t_scalar_t DIALOG_id;           /* Dialog Identifier */
        t_scalar_t COMP_flags;          /* For use with ANSI CWP/CWOP */
} TC_cont_ind_t;

typedef struct TC_cont_ind TC_conv_ind_t;

/*
 *  TC_ABORT_REQ.
 *
 *  Note:      Application context is only present if the abort reason indicates "application context
 *          not supported".
 */
typedef struct TC_abort_req {
        t_scalar_t PRIM_type;           /* Always TC_ABORT_REQ */
        t_scalar_t OPT_length;          /* Options associated with the primitive */
        t_scalar_t OPT_offset;          /* Options associated wtih the primitive */
        t_scalar_t DIALOG_id;           /* Dialog Identifier */
        t_scalar_t ABORT_reason;        /* Abort reason */
} TC_abort_req_t;

/*
 *  TC_ABORT_IND.
 *
 *  Note:      Application context is only present if the abort reason indicates "application context
 *          not supported".
 */
typedef struct TC_abort_ind {
        t_scalar_t PRIM_type;           /* Always TC_ABORT_IND */
        t_scalar_t OPT_length;          /* Options associated with the primitive */
        t_scalar_t OPT_offset;          /* Options associated wtih the primitive */
        t_scalar_t DIALOG_id;           /* Dialog Identifier */
        t_scalar_t ABORT_reason;        /* Abort reason */
        t_scalar_t ORIGINATOR;          /* Either User or Provider originated */
} TC_abort_ind_t;
```

```
/*
 *  TC_NOTICE_IND.
 */
typedef struct TC_notice_ind {
        t_scalar_t PRIM_type;           /* Always TC_NOTICE_IND */
        t_scalar_t DIALOG_id;           /* Dialog Identifier */
        t_scalar_t REPORT_cause;        /* Report cause */
} TC_notice_ind_t;

/*
 *  Component handling primitives.
 */

/*
 *  TC_INVOKE_REQ.  This primitive is one M_PROTO message block followed by zero or more M_DATA
 *  blocks containing the parameters of the operation.
 */
typedef struct TC_invoke_req {
        t_scalar_t PRIM_type;           /* Always TC_INVOKE_REQ */
        t_scalar_t DIALOG_id;           /* Dialog identifier */
        t_scalar_t PROTOCOL_class;      /* Application protocol class */
        t_scalar_t INVOKE_id;           /* Invoke Identifier */
        t_scalar_t LINKED_id;           /* Linked Invoke Identifier */
        t_scalar_t OPERATION;           /* Requested operation to invoke */
        t_scalar_t MORE_flag;           /* Not last */
        t_scalar_t TIMEOUT;             /* Timeout */
} TC_invoke_req_t;

/*
 *  TC_INVOKE_IND.  This primitive is one M_PROTO message block followed by zero or more M_DATA
 *  blocks containing the parameters of the operation.
 *
 *  Note:      Dialog Id is ignored for Class 4 (TC_UNI_IND) operations.
 */
typedef struct TC_invoke_ind {
        t_scalar_t PRIM_type;           /* Always TC_INVOKE_IND */
        t_scalar_t DIALOG_id;           /* Dialog identifier */
        t_scalar_t OP_class;            /* Application operation class */
        t_scalar_t INVOKE_id;           /* Invoke Identifier */
        t_scalar_t LINKED_id;           /* Linked Invoke Identifier */
        t_scalar_t OPERATION;           /* Requested operation to invoke */
        t_scalar_t MORE_flag;           /* Not last */
} TC_invoke_ind_t;

/*
 *  TC_RESULT_REQ.  This primitive consists of one M_PROTO message block followed by zero or more
 *  M_DATA blocks containing the parameters of the operation.
 */
typedef struct TC_result_req {
        t_scalar_t PRIM_type;           /* Always TC_RESULT_REQ */
        t_scalar_t DIALOG_id;           /* Dialog Identifier */
        t_scalar_t INVOKE_id;           /* Invoke Identifier */
        t_scalar_t OPERATION;           /* Requested operation result */
        t_scalar_t MORE_flag;           /* Not last */
} TC_result_req_t;
```

```
/*
 *  TC_RESULT_IND.  This primitive consists of one M_PROTO message block followed by zero or more
 *  M_DATA blocks containing the parameters of the operation.
 *
 *  This primitive is only valid (expected) for operation class 1 and 3.
 */
typedef struct TC_result_ind {
        t_scalar_t PRIM_type;           /* Always TC_RESULT_IND */
        t_scalar_t DIALOG_id;           /* Dialog Identifier */
        t_scalar_t INVOKE_id;           /* Invoke Identifier */
        t_scalar_t OPERATION;           /* Requested operation result */
        t_scalar_t MORE_flag;           /* Not last */
} TC_result_ind_t;

/*
 *  TC_ERROR_REQ.  This primitive consists of one M_PROTO message block followed by zero or more
 *  M_DATA blocks containing the parameters of the error.
 */
typedef struct TC_error_req {
        t_scalar_t PRIM_type;           /* Always TC_ERROR_REQ */
        t_scalar_t DIALOG_id;           /* Dialog Identifier */
        t_scalar_t INVOKE_id;           /* Invoke Identifier */
        t_scalar_t ERROR_code;          /* Error code */
        t_scalar_t MORE_flag;           /* Not last */
} TC_error_req_t;

/*
 *  TC_ERROR_IND.  This primitive consists of one M_PROTO message block followed by zero or more
 *  M_DATA blocks containing the parameters of the error.
 */
typedef struct TC_error_ind {
        t_scalar_t PRIM_type;           /* Always TC_ERROR_IND */
        t_scalar_t DIALOG_id;           /* Dialog Identifier */
        t_scalar_t INVOKE_id;           /* Invoke Identifier */
        t_scalar_t ERROR_code;          /* Error code */
} TC_error_ind_t;

/*
 *  TC_REJECT_REQ.  This primitive consists of one M_PROTO message block.
 */
typedef struct TC_reject_req {
        t_scalar_t PRIM_type;           /* Always TC_REJECT_REQ */
        t_scalar_t DIALOG_id;           /* Dialog Identifier */
        t_scalar_t INVOKE_id;           /* Invoke identifier */
        t_scalar_t PROBLEM_code;        /* Problem code */
} TC_reject_req_t;

/*
 *  TC_REJECT_IND.  This primitive consists of one M_PROTO message block.
 */
typedef struct TC_reject_ind {
        t_scalar_t PRIM_type;           /* Always TC_REJECT_IND */
        t_scalar_t DIALOG_id;           /* Dialog Identifier */
        t_scalar_t INVOKE_id;           /* Invoke identifier */
        t_scalar_t ORIGINATOR;          /* Either User, Local or Remote */
```

```
        t_scalar_t PROBLEM_code;        /* Problem code */
} TC_reject_ind_t;

/*
 *  TC_CANCEL_REQ.  This primitive consists of one M_PROTO message block.
 */
typedef struct TC_cancel_req {
        t_scalar_t PRIM_type;           /* Always TC_CANCEL_REQ */
        t_scalar_t DIALOG_id;           /* Dialog Identifier */
        t_scalar_t INVOKE_id;           /* Invoke identifier */
} TC_cancel_req_t;

/*
 *  TC_CANCEL_IND.  This primitive consists of one M_PROTO message block.
 */
typedef struct TC_cancel_ind {
        t_scalar_t PRIM_type;           /* Always TC_CANCEL_REQ */
        t_scalar_t DIALOG_id;           /* Dialog Identifier */
        t_scalar_t INVOKE_id;           /* Invoke identifier */
} TC_cancel_ind_t;

/*
 *  Management primitives.
 */

/*
 * TC_COORD_REQ.
 */
typedef struct TC_coord_req {
        t_scalar_t PRIM_type;           /* alwyas TC_COORD_REQ */
        t_scalar_t ADDR_length;         /* affected subsystem */
        t_scalar_t ADDR_offset;
} TC_coord_req_t;

/*
 * TC_COORD_RES.
 */
typedef struct TC_coord_res {
        t_scalar_t PRIM_type;           /* always TC_COORD_RES */
        t_scalar_t ADDR_length;         /* affected subsystem */
        t_scalar_t ADDR_offset;
} TC_coord_res_t;

/*
 * TC_COORD_IND.
 */
typedef struct TC_coord_ind {
        t_scalar_t PRIM_type;           /* alwyas TC_COORD_IND */
        t_scalar_t ADDR_length;         /* affected subsystem */
        t_scalar_t ADDR_offset;
        t_scalar_t SMI;                 /* subsystem multiplicity indicator */
} TC_coord_ind_t;

/*
 * TC_COORD_CON.
 */
```

```
typedef struct TC_coord_con {
        t_scalar_t PRIM_type;           /* always TC_COORD_CON */
        t_scalar_t ADDR_length;         /* affected subsystem */
        t_scalar_t ADDR_offset;
        t_scalar_t SMI;                 /* subsystem multiplicity indicator */
} TC_coord_con_t;

/*
 * TC_STATE_REQ.
 */
typedef struct TC_state_req {
        t_scalar_t PRIM_type;           /* always TC_STATE_REQ */
        t_scalar_t ADDR_length;         /* affected subsystem */
        t_scalar_t ADDR_offset;
        t_scalar_t STATUS;              /* user status */
} TC_state_req_t;

/*
 * TC_STATE_IND.
 */
typedef struct TC_state_ind {
        t_scalar_t PRIM_type;           /* always TC_STATE_IND */
        t_scalar_t ADDR_length;         /* affected subsystem */
        t_scalar_t ADDR_offset;
        t_scalar_t STATUS;              /* user status */
        t_scalar_t SMI;                 /* subsystem multiplicity indicator */
} TC_state_ind_t;

/*
 * TC_PCSTATE_IND.
 */
typedef struct TC_pcstate_ind {
        t_scalar_t PRIM_type;           /* always TC_PCSTATE_IND */
        t_scalar_t ADDR_length;         /* affected point code */
        t_scalar_t ADDR_offset;
        t_scalar_t STATUS;              /* status */
} TC_pcstate_ind_t;

/*
 * TC_TRAFFIC_IND
 */
typedef struct TC_traffic_ind {
        t_scalar_t PRIM_type;           /* always TC_TRAFFIC_IND */
        t_scalar_t ADDR_length;         /* affected user */
        t_scalar_t ADDR_offset;
        t_scalar_t TRAFFIC_mix;         /* traffic mix */
} TC_traffic_ind_t;

#endif                          /* __SS7_TC_H__ */
```

Glossary

Signalling Data Link Service Data Unit
: A grouping of SDL user data whose boundaries are preserved from one end of the signalling data link connection to the other.

Data transfer
: The phase in connection and connectionless modes that supports the transfer of data between to signalling data link users.

SDL provider
: The signalling data link layer protocol that provides the services of the signalling data link interface.

SDL user
: The user-level application or user-level or kernel-level protocol that accesses the services of the signalling data link layer.

Local management
: The phase in connection and connectionless modes in which a SDL user initializes a *Stream* and attaches a PPA address to the *Stream*. Primitives in this phase generate local operations only.

PPA
: The point at which a system attaches itself to a physical communications medium.

PPA identifier
: An identifier of a particular physical medium over which communication transpires.

Acronyms

ITU-T	International Telecommunications Union - Telecom Sector
PPA	Physical Point of Attachment
SDLI	Signalling Data Link Interface
SDL SDU	Signalling Data Link Service Data Unit
SDL	Signalling Data Link

References

1. ITU-T Recommendation X.210, (Geneva, 1993), "Information Technology — Open Systems Interconnection — Basic reference model: Conventions for the definition of OSI services," ISO/IEC 10731:1994.
2. ITU-T Recommendation X.217, (Geneva, 1995), "Information Technology — Open Systems Interconnection — Service definition for the Association Control Service Element," ISO/IEC 8649:1996.
3. ITU-T Recommendation X.227, (Geneva, 1995), "Information Technology — Open Systems Interconnection — Connection-oriented protocol for the Association Control Service Element: Protocol Specification," ISO/IEC 8650-1.
4. ITU-T Recommendation X.237, (Geneva, 1995), "Information Technology — Open Systems Interconnection — Connectionless protocol for the Association Control Service Element: Protocol Specification," ISO/IEC 10035-1 : 1995.
5. ITU-T Recommendation X.216, (Geneva, 1994), "Information Technology — Open Systems Interconnection — Presentation service definition," ISO/IEC 8822:1994.
6. ITU-T Recommendation X.226, (Geneva, 1994), "Information Technology — Open Systems Interconnection — Connection-oriented presentation protocol: Protocol specification," ISO/IEC 8823-1:1994.
7. ITU-T Recommendation X.236, (Geneva, 1995), "Information Technology — Open Systems Interconnection — Connectionless presentation protocol: Protocol specification," ISO/IEC 9576-1:1995.
8. ITU-T Recommendation X.215, (Geneva, 1995), "Information Technology — Open Systems Interconnection — Session service definition," ISO/IEC 8326:1996.
9. ITU-T Recommendation X.225, (Geneva, 1995), "Information Technology — Open Systems Interconnection — Connection-oriented session protocol: Protocol specification," ISO/IEC 8327-1:1996.
10. ITU-T Recommendation X.235, (Geneva, 1995), "Information Technology — Open Systems Interconnection — Connectionless session protocol: Protocol specification," ISO/IEC 9548-1:1995.
11. ITU-T Recommendation X.214, (Geneva, 1995), "Information Technology — Open Systems Interconnection — Transport service definition," ISO/IEC 8072:1996.
12. ITU-T Recommendation X.224
13. ITU-T Recommendation Q.700
14. ITU-T Recommendation Q.701
15. ITU-T Recommendation Q.702
16. ITU-T Recommendation Q.703
17. ITU-T Recommendation Q.704
18. Geoffrey Gerrien, "CDI - Application Program Interface Guide," Gcom, Inc., March 1999.
19. ITU-T Recommendation Q.771, (Geneva, 1993), "Signalling System No. 7 — Functional description of transaction capabilities," (White Book).

Licenses

All code presented in this manual is licensed under the [GNU Affero General Public License], page 95. The text of this manual is licensed under the [GNU Free Documentation License], page 105, with no invariant sections, no front-cover texts and no back-cover texts. Please note, however, that it is just plain wrong to modify statements of, or attribute statements to, the Author or *OpenSS7 Corporation*.

GNU Affero General Public License

The GNU Affero General Public License.
Version 3, 19 November 2007

Preamble

The GNU Affero General Public License is a free, copyleft license for software and other kinds of works, specifically designed to ensure cooperation with the community in the case of network server software.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, our General Public Licenses are intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

Developers that use our General Public Licenses protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License which gives you legal permission to copy, distribute and/or modify the software.

A secondary benefit of defending all users' freedom is that improvements made in alternate versions of the program, if they receive widespread use, become available for other developers to incorporate. Many developers of free software are heartened and encouraged by the resulting cooperation. However, in the case of software used on network servers, this result may fail to come about. The GNU General Public License permits making a modified version and letting the public access it on a server without ever releasing its source code to the public.

The GNU Affero General Public License is designed specifically to ensure that, in such cases, the modified source code becomes available to the community. It requires the operator of a network server to provide the source code of the modified version running there to the users of that server. Therefore, public use of a modified version, on a publicly accessible server, gives the public access to the source code of the modified version.

An older license, called the Affero General Public License and published by Affero, was designed to accomplish similar goals. This is a different license, not a version of the Affero GPL, but Affero has released a new version of the Affero GPL which permits relicensing under this license.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions

0. Definitions.

   "This License" refers to version 3 of the GNU Affero General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

   The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

   The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the

source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   a. The work must carry prominent notices stating that you modified it, and giving a relevant date.

b. The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c. You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d. If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c. Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d. Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e. Convey the object code using peer-to-peer transmission, provided you inform other peers

where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a. Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b. Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c. Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d. Limiting the use for publicity purposes of names of licensors or authors of the material; or

e. Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f. Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

   You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

    Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

    An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

    You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

    A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

    A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

    Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

    In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

    If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to

downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Remote Network Interaction; Use with the GNU General Public License.

    Notwithstanding any other provision of this License, if you modify the Program, your modified version must prominently offer all users interacting with it remotely through a network (if your version supports such interaction) an opportunity to receive the Corresponding Source of your version by providing access to the Corresponding Source from a network server at no charge, through some standard or customary means of facilitating copying of software. This Corresponding Source shall include the Corresponding Source for any work covered by version 3 of the GNU General Public License that is incorporated pursuant to the following paragraph.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the work with which it is combined will remain governed by version 3 of the GNU General Public License.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU Affero General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU Affero General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU Affero General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU Affero General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and a brief idea of what it does.
Copyright (C) year name of author

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU Affero General Public License as published by
the Free Software Foundation, either version 3 of the License, or (at
your option) any later version.

This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License
along with this program.  If not, see http://www.gnu.org/licenses/.
```

Also add information on how to contact you by electronic and paper mail.

If your software can interact with users remotely through a network, you should also make sure that it provides a way for users to get its source. For example, if your program is a web application, its interface could display a "Source" link that leads users to an archive of the code. There are many ways you could offer source, and different solutions will be better for different programs; see section 13 for the specific requirements.

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU AGPL, see http://www.gnu.org/licenses/.

GNU Free Documentation License

GNU FREE DOCUMENTATION LICENSE
Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

   It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

   A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

   B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

   C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

   D. Preserve all the copyright notices of the Document.

   E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

   F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index

T

U

X