XOM Call Control (XCC) Specification

XOM Call Control (XCC) 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 XOM Call Control (XCC) for OpenSS7. It contains recommendations on software architecture as well as platform and system applicability of the XOM Call Control (XCC). It provides abstraction of the Call Control (CC) interface to these components as well as providing a basis for Call Control control for other Call Control 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 43). 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 53) 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 XOM Call Control (XCC) for OpenSS7. It contains recommendations on software architecture as well as platform and system applicability of the XOM Call Control (XCC).

This document specifies a XOM Call Control (XCC) Specification in support of the OpenSS7 Call Control (CC) protocol stacks. It provides abstraction of the Call Control interface to these components as well as providing a basis for Call Control control for other Call Control protocols.

Purpose

The purpose of this document is to provide technical documentation of the XOM Call Control (XCC). 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 XOM Call Control (XCC) 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 XOM Call Control (XCC). This document is intended to provide information for writers of OpenSS7 XOM Call Control (XCC) applications as well as writers of OpenSS7 XOM Call Control (XCC) Users.

Audience

The audience for this document is software developers, maintainers and users and integrators of the XOM Call Control (XCC). 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: xcc.texi,v $
Revision 1.1.2.2  2011-02-07 02:21:48  brian
- updated manuals

Revision 1.1.2.1  2009-06-21 10:57:56  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

1.1 Overview

The XOM Call Control Programming Interface (abbreviated XCC) defines an Application Program Inteface (API) to call control services. It is referred to as *the interface* throughout this specification.

The interface is designed to offer services that are consistent with, but not limited to, the ITU-T Recommendation Q.761, Q.931, H.225.0 and IETF SIP-T standards. These standards were published in 1997 and have been stable for many years.

The interface is also designated to offer services that are consistent with the 3GPP TS 28.001 standards for GSM mobile networks.

All of the above standards are referred to in this document as *the Standards*.

Access to other call control services through the API is not prohibited, but has not been explicitly considered.

The interface is designed to be used and implemented in conjunction with the use and implementation of the general-purpose XOM API (reference **XOM**).

A brief introduction to Call Control Services is given in Section 1.3 [Introductory Concepts], page 7. Following this is an overview of OSI-Abstract-DATA Manipulation OM, which provides the Data Abstraction service as defined in the XOM specification (reference **XOM**). Then the optional features of this specification are described, and the chapter closes with a list of abbreviations. In all cases the reader should refer to the Standards (reference **ISUP**, reference **ISDN**, reference **H.323**, or to the XOM Specification (reference **XOM**) for further authoritative details.

The structure of the remaining chapters and appendices are described in the [Preface], page 3.

1.2 Format of the Specification

This specification describes a programming language-independent interface to the Call Control Services together with a specific C language binding of that interface. Several conventions are used to identify particular items. The general conventions are described in the [Preface], page 3, while the C language binding conventions are described in Chapter 2 [C Language Binding], page 11.

1.3 Introductory Concepts

1.3.1 Relationship to Call Control Protocols

The interaction between call control programs acting in a call control entity role are realized through the exchange of call control service information. The general communications service for the call control applications is the *Call Control Protocol*. Call control protocols define the following operations:

Service **Type**

This communication may be accomplished using the ITU-T or ANSI *Message Transfer Part* protocol.

1.3.2 XCC and the Call Control Provider

The XCC interface provides access to the CC service provider, which offers all of the facilities defined in the Standards. It also provides facilities such as automatic association management and automatic session handling. The interface is designed not to restrict the services offered to those of specific service packages of ISUP.

The interface defined in this specification is "symmetrical" in the sense that it can be used to implement call control programs acting in any of the CC producer or consumer roles (e.g. MSC, BSC). The interface supports:

- a call control program acting as a consumer of call control services. This is done by submitting service operation requests and receiving service operation responses.
- a call control program acting as a producer of call control services. This is done by receiving service operation requests and sending back service operation responses.

The interface provides the ability to send *requests* on the consumer side and to receive *indications* on the producer side within a call control interaction. Furthermore, if the service is confirmed, the producer will be able to send back *responses* that will be received as *confirmations* by the consumer.

1.4 Relationship to ISUP, ISDN, H.323 and SIP-T

The API is essentially based on the abstract services of the ITU-T ISUP and ISDN, but is independent of the underlying communications stack. The API allows the manipulation of ITU-T and ANSI call control service information. Thus this API does not preclude and does not force the use of either the ITU-T ISUP protocol or the ANSI ISUP protocol.

The XCC API offers several abstract call control service views: the on of ISUP, the one of ISDN, the one of H.323, the one of GSM BSSAP, and that of SIP-T.

The contents of ITU-T ISUP messages are described in ITU-T Recommendation Q.761. These messages implicitly define the ITU-T ISUP services. The mapping between ITU-T ISUP services and various service primitives and parameters of the XCC API are described below.

The services offered by the XCC API are a superset of those defined by ITU-T and ANSI. The general communication protocol for ANSI ISUP is the MTP as specified in ANSI T1.113.4/200. The general communication protocol for ITU-T ISUP is the MTP as specified in ITU-T Recommendation series Q.761 through Q.764.

The abstract call control views of XCC (ITU-T and ANSI, ISUP, ISDN, H.323, BSSAP or SIP-T) are independent of the underlying protocol. However, the ITU-T view on top of ANSI MTP requires that an appropriate mapping of ITU-T ISUP services over ANSI MTP is widely available.

1.5 Relationship to Data Abstraction Services

XCC is dependent on standard data abstraction services to ensure portability of call control software written to the XCC specification. XCC functions pass most arguments by reference. The data referenced by these arguments is modelled and manipulated in an object-oriented fashion. Call application data abstraction services are provided by the XOM API (reference **XOM**).

The definitions below introduce the various concepts that are used by Call Control data abstraction services.

Syntax A *syntax* is the classification and representation of values in OSI-Abstract-Data Manipulation. Examples of syntaxes are *Boolean*, *Integer*, *Real*, *String(Octet)*, *String(Object-Identifier)* and *Object*.

Value A *value* is a single datum, or piece of information. A value may be as simple as a Boolean value (for example, *True*), or as complicated as an entire OM object (for example, a *Message*).

OM Attribute
An *OM attribute type* is an arbitrary category into which a specification places some values. An *OM attribute* is an OM Attribute Type, together with an ordered sequence

of one or more values. The OM Attribute Type can be thought of as the name of the OM attribute.

OM Object An *OM object* is a collection of OM attributes.

OM Class An *OM class* is a category of OM objects set out in a specification. It determines the OM attributes that may be present in the OM object, and details the constraints on those OM attributes.

Package A *Package* is a set of OM classes that are grouped together by the specification, because tey are functionally related (for example, ISUP service package).

Package Closure
A *Package-Closure* is the set of classes that need to be supported to be able to create all possible instances of all classes defined in the package. Thus an OM Class may be defined to have an OM Attribute whose value is an OM Obejct of an OM Class that is defined in some other package, but within the same *Package-Closure*.

Workspace A *workspace* is allocated storage that contains one or more *Package-Closures*, together with an implementation of the Call Control data abstraction services, that supports all the OM classes of OM objects in the *Package-Closures*.

Descriptor A *descriptor* is a defined data structure that is used to represent an OM Attribute Type and a single value. The structure has three components: a type, a syntax and a value.

Public Object
Public Objects are represented by data structures that are manipulated directly using programming language constructs. Use of Public Objects therefore simplifies programming by this direct access and by enabling objects to be statically defined, when appropriate. Programs can efficiently access public objects.

Private Objects
Private Objects are held in data structures that are private to the service and can only be accessed from programs indirectly using interface functions. They are of particular use for structures that are infrequently manipulated by programs, being passed by reference to the service, which can then manipulate them efficiently. An example of such objects in XCC is the *session* object.

1.6 Mandatory and Optional Features

The interface defines an Application Program Interface (API) that application programs can use to access the functionality of the underlying Call Control Services. The interface does not define or imply any profile of that service.

Note that nothing in this specification requires that the implementation of the interface or the Call Control Services itself acutally makes use of TCAP or other parts of the model, just so long as it provides the defined service. Also, the *scope* of the Call Control Services to which an application has access is not determined; it is not restricted to ITU-T ISUP operations.

Some OM attributes are optional: these are marked *(Optional Functionality)* in the OM class definitions. They are:

- **File-Descriptor** in a **Session** object.

Some items of behaviour of the interface and a number of aspects of the Call Control Services provider are implemetnation-defined. These are:

- the maximum number of oustanding asynchronous operations
- whether an asyncrhonous function call returns before the operation is submitted to the Call Control Services provider
- the text and language of error messages
- the OM classes permitted as values of the **Address** and **Title** argument to interface functions.

The default values of some OM attributes on OM object **Session** are locally administered.

This API assumes the provision of automatic association management and automatic session handling by the CC provider.

The interface enables negotiation of the use of the various defined features of the CC provider and those of the interface.

1.7 Packages

The specification defines three Call Control packages (Common CC package, ITU-T CC package and ANSI CC package), Chapter 5 [Interface Class Definitions], page 29. These packages define the OM classes required by the interface functions to perform ITU-T CC or ANSI CC services. The Common CC package, which also includes the errors defined (see Chapter 6 [Errors], page 31), is mandatory. The ITU-T CC package and the ANSI CC package are optional, but at least one of them must be supported by the implementation. The ITU-T service view and the ANSI service view assume the support of the corresponding ITU-T CC or ANSI CC package by the implementation.

The use of the optional packages is negotiated using the `Negotiate()` function.

1.8 Terminology

The terms *implementation-defined*, *may*, *should*, *undefined*, *unspecified*, and *will* are used in this document with the meanings ascribed to them in reference **XPG4**, see also [Glossary], page 41.

1.9 Abbreviations

API	Application Program Interface
ANS.1	Abstract Syntax Notation One
ANSI	American National Standards Institute
BER	Basic Encoding Rules
GSM	Global Services Mobile
ISO	International Organisation for Standardisation
ITU-T	Internation Telecommunications Union - Telecom Sector
MAP	Mobile Application Part
OM	OSI-Abstract-Data Manipulation
OSI	Open Systems Interconnect
ROSE	Remote Operations Service Element
TCAP	Transaction Capabilities Application Part
XMAP	XOM Mobile Application Part API
XOM	X/Open: OSI-Abstract-Data Manipulation API

2 C Language Binding

This chapter sets out certain characteristics of the C language binding to the interface. The binding specifies C identifiers for all the elements of the interface, so that application programs written in C can access the Mobile Application Services. These elements include function names, *typedef* names and constants. All of the C identifiers are mechanically derived from the language independent names as explained below. There is a complete list of all the identifiers in Appendix A [C Headers], page 33. For ease of use, some of these identifiers are defined in the specification alongside the language-independent name.

A *Function()* is indicated as shown.

A `CONSTANT` is in Roman font.

The names of `[[ERRORS]]` and other return codes are surrounded by square brackets.

The definitions of the C identifiers appear in four headers:

<xom.h> This header file contains definitions for the associated OM interface.

<xcc.h> This header file contains common definitions for the access to the Mobile Application Part service (see Chapter 4 [Interface Functions], page 27, and ⟨undefined⟩ [⟨undefined⟩], page ⟨undefined⟩). A listing of this header file is provided in Section A.1 [xcc.h], page 33.

<xcc_gsm.h>
This header file contains specific definitions that reflect the Abstract Services of the ITU-T Mobile Services along with the ASN.1 productions of the related protocol (ITU-T CC), See ⟨undefined⟩ [⟨undefined⟩], page ⟨undefined⟩. A listing of this header file is provided in ⟨undefined⟩ [⟨undefined⟩], page ⟨undefined⟩.

<xcc_ansi.h>
This header file contains specific definitions that reflect the Abstract Services of the ANSI Mobile Services allong with the ASN.1 productions of the related protocol (ANSI CC), See ⟨undefined⟩ [⟨undefined⟩], page ⟨undefined⟩. A listing of this header file is provided in ⟨undefined⟩ [⟨undefined⟩], page ⟨undefined⟩.

<xcc_gsm_sm.h>
This header file contains specific definitions that reflect the Short Message ITU-T services along with ASN.1 productions of the related services (ITU-T CC Short Message services), See ⟨undefined⟩ [⟨undefined⟩], page ⟨undefined⟩. A listing of this header file is provided in ⟨undefined⟩ [⟨undefined⟩], page ⟨undefined⟩.

2.1 C Naming Conventions

The interfaces uses part of the C public namespace for its facilities. All identifiers start with the letters cc, CC or OCC, and more detail of the conventions used are given in the following table. Note that the interface reserves *all* identifiers starting with the letters *ccP* for private (i.e. internal) use by implementations of the interface. It also reserves *all* identifiers starting with the letters *ccX* or *CCX* for vendor-specific extensions of the interface. Application programmers should not use any identifier starting with these letters.

The OSI-Abstract-Data Manipulation API uses similar, thorugh not identical, naming conventions, that are described in XOM (reference **XOM**). All its identifiers are prefixes by the letters *OM* or *om*.

reserved for implementors	*ccP*

reserved for interface extensions	*ccX*
reserved for interface extensions	*CCX*
reserved for implementors	*OCC*

functions	*cc_*
error problem values	*CC_E_*
enumeration tags (except errors)	*CC_T_*
OM classs names	*CC_C_*
OM value length limits	*CC_VL_*
OM value number limits	*CC_VN_*
other constants	*CC_*

A complete list of all identifiers used (except those beginning *ccP*, *ccX*, *CCX* or *OCC*) is given in Appendix A [C Headers], page 33. No implementation of the interface will use any other public identifiers. A *public identifier* is any name except those reserved in section 4.1.2.1 of the ISO C Standard, and the *public namespace* is the set of all possible public identifiers.

The C identifiers are derived from the langauage-independent names used throughput this specification by a purely mechanical process which depends on the kind of name:

- Interface function names are made entirely lower-case and prefixed by *cc_*. Thus **Service-Req()** becomes `cc_service_req()`.
- C function parameters are derived from the argument and result names by making them entirely lower-case. In addition, the names of results have *_return* added as a suffix. Thus the argument **Session** becomes *session*, while the result of **Result** becomes *result_return*.
- OM class names are made entirely upper-case and prefixed by *CC_C_*. Thus **Service-Argument** becomes *CC_C_SERVICE_ARGUMENT*. Note that the symbolic OM class names are strictly those used in the abstract syntax ASN.1 of the TCAP and CC with the exception that names containing multiple words are separated by hyphens.
- Enumeration tags are derived from the name of the corresponding OM type and syntax by prefixing *CC_*. The case of letters is left unchanged. Thus **Enum(User-reason)** befomes *CC_User_reason*.
- Enumeration constants, except erros, are made entirely upper-case and prefixed by *CC_T_*. Thus **resource-limitation** becomes *CC_T_RESOURCE_LIMITATION*.
- The name of an OM attribute is local to its OM class, that measn the same name of an OM attribute may appear in different OM classes, for example, OM attribute **application-Context** is defined in both OM classes **Open-Arg** and **application-Context-List**. Independent-language atribute **application-Context** appears as *CC_APPLICATION_CONTEXT* in C-language. Note that the symbolic OM attribute names are strictly those used in the abstract syntax ASN.1 of the TCAP and CC with the exception that names containing multiple words are separated with hyphens.
- Errors are treated as a special case. Constants that are the possible values of the OM attribute **Error-Status** of a subclass of the OM class **Error** are made entirely upper-case and prefixed by *CC_E_*. Thus **invalid-session** becomes *CC_E_INVALID_SESSION*.
- The constants in the **Value Length** and **Value Number** columns of the OM class definition tables are also assigned identifiers. (They have no names in the language-independent specification.) Where the upper limit in on eof these columns is not “1” (one), it is given a name consisting of the OM attribute name, prefixed by *CC_VL_* for value length, or *CC_VN_* for value numbers.
- The sequence of octets for each object identifier is also assigned an identifier, for internal use

by certain OM macros. These identifiers are all upper case and are prefixed by *OMP_O_*. See reference **XOM** for further details on the use of object identifiers.

Note that hyphens are translated everywhere to underscores.

2.2 Use and Implementtion of Interfaces

Each of the following statements applies unless explicitly state otherwise in the detailed descriptions that follow:

If an argument to a function has an invalid value (such as a value outside the domain of the function, or a pointer outside the address space of the program, or a null pointer), the behvaiour is *undefined*.

Any function declared in a header may be implemented as a macro defined in the header, so a library function should not be declared explicitly if its header is included. Any macro definition of a function can be suppressed locally be encoding the name of the function in parentheses, because the name is not then followed by the left parthesis that indicate expansion of a macro function name. For the same syntactic reason, it is permitted to take the address of a library function even if it is also defined as a macro. The use of `#undef` to remove any macro defintion will also ensure that an actual function is referred to. Any invocation of a library function that is implemented as a macro will expand to code that evaluates each of its arguments exactly once, fully protected by parentheses where necessary, so it is generally safe to use arbitrary expressions as arguments. Likewise, those function-like macros described in the following sections may be invoked in an expression anywhere a function with a compatible return type could be called.

2.3 Function Return Values

The return value of a C function is always bound to the result of the language-independent description. Functions return a value of `CC_status`, which is an error indication. If and only if the function succeeds, its value will be **success**, expressed in C by the constant *CC_SUCCESS*. If a function returns a status other than this, then it has not updated the return parameters. The value of the status, in this case, is an error as described in Chapter 6 [Errors], page 31. In most cases the integer returned in **Status** is sufficient for error processing. However, in a few cases additional information is available if desired.

Since C does not provide multiple return values, functions must return all other results by writing into storage passed by the application program. Any argument that is a pointer to such storage has a name ending with *_return*. For example, the C parameter declaration '`OM_sint *invoke_id_return`' in the *Service-Req()* function indicates that the function will return an signed integer **Invoke-Id** as a result, so the actual argument to the function must be the address of a suitable variable. This notation allows the reader to distinguish between an input parameter that happes to be a pointer, and an output parameter where the `*` is used to simulate the semantics of passing by reference.

2.4 Compilation and Linking

All applications programs that use this interface include the `<xom.h>` and `<xcc.h>` headers in that order, and at least one of the `<xcc_gsm.h>` and `<xcc_ansi.h>` headers.

3 Description

The interface comprises a number of functions together with many OM classes and OM objects that are used as the arguments and results of the functions. Both the functions and the OM objects are based closely on the *Abstract Service* that is specified in the Standards (references ITU-T ISUP and ANSI ISUP).

The interface models mobile application interactions as service requests made through a number of interface *functions*, which take a number of input *arguments*. Each valid request causes an *operation* within the producer which eventually returns a *status* and any *result* of the operation.

All interactions between a Consumer and a Producer belong to a *session*, which is represented by an OM object passed as the first argument to most interface functions.

The other arguments to the function include a *context* and various service-specific arguments. The *context* includes a number of parameters that are common to many functions and that seldom change from operation to operation.

Each of the components of this model is described below, along with other features of the interface such as asynchronous function calls and security.

3.1 Services

As mentioned above, the Standards define Abstract Services that Consumers and Producers use. Each of these Absract Services maps to a single function call with the same name. The services are **Service-req** and **Service-rsp**.

These are three functions called *Receive()*, *Wait()*, and *Abandon()* which have no counterpart in the Abstract Service. *Receive()* is used to receive indications and results of asynchronous operations, and is explained in Chapter 4 [Interface Functions], page 27. *Wait()* is used to suspend execution until indications are available for specified sessions. *Abandon()* is used to abandon locally the result of a pending asyncrhonous operation. Two additional functions *Bind()*[1] and *Unbind()* are used to open and close a user-session.

There are also other interface specific functions called *Get-Assoc-Info()*, *Get-Last-Error()*, *Validate-object()*, *Error-Message()*, *Initialize()*, *Shutdown()* and *Negotiate()*.

The detailed specifications are given in Chapter 4 [Interface Functions], page 27.

3.1.1 Negotiation Sequence

The interface has an initialize and shutdown sequence that permits the negotiation of optional features. This involves the functions *Initialize()*, *Negotiate()*, and *Shutdown()*.

Every application program must first call *Initialize()*, that returns a workspace. This workspace supports only the standard Common ISUP package, See Chapter 5 [Interface Class Definitions], page 29.

The workspace can be extended to support either the ITU-T ISUP or ANSI ISUP package or both (see Chapter 5 [Interface Class Definitions], page 29, and any combination of the optional Mobile Application Services packages), or any vendor extensions. Vendor extensions may include additional packages, and may also include additional or modified functionality. All such packages or other extensions are identified by means of OSI Object Identifiers, and the Object Identifiers are supplied to the *Negotiate()* function to incorporate the extensions into the workspace. Features defined by this specification are described and assigned Object Identifiers in Chapter 4 [Interface Functions], page 27. A feature represents any package or any additional or modified functionality that is subject to negotiattion. The *Negotiate()* function allows some particular features to be made available.

[1] See ⟨undefined⟩ [⟨undefined⟩], page ⟨undefined⟩.

After a worksapce with the required features has been negotiated in this way, the application can use the workspace as required. It can create and manipulate OM objects using the OM functions, and can start one or more management sessions using *Bind()*.[2] All the sessions on a given workspace share the same features.

Eventually, when it has completed its tasks, terminated all is mobile application sessions using *Unbind()*, and released all its OM objects using *OM-Delete()*, the application should ensure that resources associated with the interface are freed by calling *Shutdown()*.

A miscellaneous error arises if an attempt is made to use an unavailable feature. If an instance of a class that is not in an available package is supplied as a function argument, the **bad-class** error arises.

3.1.2 Names, Addresses and Titles

To address a wide variety of mobile application transport protocols the interface is capable of accepting various forms of object names, system addresses and program or system titles.

- **Name** is an "abstract class" that contains various subclass types used to define specific subscribers or systems responsible for producing mobile application services.
- **Address** is an "abstract class" that contains various subclass types used to define the specific location to contact a particular consumer or producer of mobile services. For example, the SCCP-Address subclass is typically used to define the location of a producer or consumer.
- **Title** is an "abstract class" that contains various subclass types used to define a specific subscriber or system name reponsible for producing mobile application services.

All three abstract classes participate in an implementation-specific name resolution scheme. It is assumed that given a **Name**, an implementation can determine the **Title** responsible for that **Name**. It is also assumed that given a **Title**, an implementation can determine the **Address** of that **Title**.[3]

The producer of an invoked operation may be explicitly designated at the interface boundary using the following precedence rules:

1. A default Title or Address may be supplied as parameters to a bound "session". If both are provided, the implementation will verify that the Title resolves to the Address.
2. If automatic association management is used, a provider Title or Address may be supplied as parameters within the "context" or a specific operation request. If both are provided, the implementation will verify that the Title resolves to the Address. The "context" Title or or Address takes precedence over the "session" Title or Address for unassociated session objects.
3. A consumer address may be supplied as a parameter within the "argument" of a specific operation request. The "argument" Address takes precedence over either the "session" Title or Address or the "context" Title or Address.
4. If the producer of an invoked operation is not explicitly designated at the interface boundary, the implementation will resolve the Name to the appropriate Title or Address.

3.2 Session

A session identifies to which mobile application entity a particular operation will be sent. It contains some **Bind-Arguments**, such as the name of the consumer. The session is passed as the first argument to most interface functions.

[2] See ⟨undefined⟩ [⟨undefined⟩], page ⟨undefined⟩.

[3] Note that the way in which these relationships are resolved is implementation-dependent, but use of SCCP Global Title translations should play a significant role.

A session is described by an OM object of OM class **Session**. It is created, and appropriate parameter values may be set, using the OSI-Abstract-Data Manipulation functions. A mobile application session is then started with *Bind()*[4] and later is terminated with *Unbind()*. A session with default parameters can be started by passing the constant **Default-Session** (‘`(OM_object)CC_DEFAULT_SESSION`’) as the **Session** argument to *Bind()*.

Bind() must be called before the **Session** can be used as an argument to any other function in the interface. After *Unbind()* has been called, *Bind()* must be called again if another session is to be started.

The interface supports multiple concurrent sessions, so that an application implemented as a single process, such as a server in a client-server model, can interact with the Mobile Application Services using several identities; and so that a process can interact directly and concurrently with different mobile application services.

Detailed specifications of the OM class **Session** are given in Chapter 5 [Interface Class Definitions], page 29.

A session can be used either acting as a consumer of mobile application services, or acting as a producer of mobile application services, or both.

A session acn be restricted for use only with a designated program called the responder. When the responder is omitted and automatic association management is used, the session can be used to exchange mobile application service information with all processes.

The responder (title and address) parameters of an opened session, if present, specifies the producer of the requested operation. The precedence rules on address and title of the responder are described in Section 3.1.2 [Names, Addresses and Titles], page 16.

Other OM attributes (vendors’ implementation extensions) may be included to specify characteristics of the underlying protocol used.

There are three type of session objects:

3.2.1 AAM Enabled Session

The **Session** collects together all the information that described a particular management interaction. The parameters that are to control such a session are set up in an instance of this OM class, which is then passed as an argument to *Bind()*.[5] This sets the OM attributes that describe the actual characteristics of the session, and starts the session. Such a started session can be passed as the first argument to interface functions.

No attribute of a bound or connected session may be changed. The result of modifying a started session is unspecified.

Finally, *Unbind()* is used to terminate the sesion, after which the parameters can be modified an a new session started using the same instance, if required. Multiple concurrent sessions can be run, by using multiple instances of this OM class.

A session allows a requesting program (the requestor) to exchange mobile application information with another program designated (the responder) or by default to all programs.

An *AAM enabled* session thus allows a mobile application entity to access either a portion of the mobile application services (that is, that are accessible via the designated responder) or all mobile application services. In the later case, the producer mobile application entity resolution is performed by the Mobile Application Service provider, according to the mobile application services invoked.

This type of session object can not be used to receive or send ACSE related primitives or operations explicitly. The use ACSE explicitly, see Section 3.2.2 [AAM Disabled Session], page 18.

[4] See ⟨undefined⟩ [⟨undefined⟩], page ⟨undefined⟩.
[5] See ⟨undefined⟩ [⟨undefined⟩], page ⟨undefined⟩.

3.2.2 AAM Disabled Session

A session object can have Automatic Association Management disabled when it belongs to a workspace that has Automatic Association Management disabled via *Negotiate()*, which allows the user to explicitly send and receive ACSE operations to build and tear down associations. It gives explicit control over associations to the user. The Mobile Application Service provider does no ACSE operations on behalf of the user.

When the user creates and binds a session object in a workspace with AAM disabled, only the following attributes within the session object can be specified:

- *requestor-Address*
- *requestor-Title*

The session object is then passed as an argument to *Bind()*,[6] which binds the session. This bound session can only be used to send ACSE related operations an to receive ACSE related primitives. The following can be sent/received using this type of bound session:

- *Receive()* (`cc_receive()`/'`CC_ASSOC_IND`')
- *Receive()* (`cc_receive()`/'`CC_ASSOC_CNF`')
- *Assoc-req()* (`cc_assoc_req()`)
- *Assoc-rsp()* (`cc_assoc_rsp()`)

The other attributes that relate to ACSE are specified within an **Assoc-Argument** or **Assoc-Result** object that is passed to, or returned from, *Assoc-req()*, *Assoc-rsp()*, or *Receive()*.

3.2.3 Associated Session

Once a user has created a bound session that has AAM disabled, an association can be created. An association is represented by an *associated* or *partially associated* session object. An *associated* session is returned as the result of building a new association. The associated session is used, like a bound session, by sending and receiving mobile application dialog handling or service operations. The major difference is that an associated session object can only be used to send and receive operations to, or from, a single remote mobile application entity. After a session is associated, the user can abort the association, which implicitly unbinds the associated, or partialyl associated, session.

The precedence rules for common parameters within the **Session** and the **Context** objects are different for associated session objects. Once a session is in the associated state; the *responder-Address* and *responder-Title* cannot be overridden by the context object.

To terminate this type of session, the user should either abort the session, which implicitly unbinds the session. If the user unbinds the associated session prior to aborting the association, the service provider will abort the association.

3.2.4 ADH Enabled Session

The ADH enabled session allows a mobile application entity to invoke and respond to mobile application services requests and indications without regard for dialog handling. The Mobile Application Service provider provides all dialog handling.

This type of session cannot be used to send dialog handling primitives or operations explicitly. To use dialog handling explicitly, see Section 3.2.5 [ADH Disabled Session], page 19.

[6] See ⟨undefined⟩ [⟨undefined⟩], page ⟨undefined⟩.

3.2.5 ADH Disabled Session

A session object can have Automatic Dialog Handling disabled when it belongs to a workspace that has Automatic Dialog Handling disabled using the *Negotiate()* function. This allows the user to explicitly send and receive dialog handling operations to establish and tear down dialogs. It gives explicit control over dialogs to the user. The Mobile Application Service provider does no dialog handling operations on behalf of the user.

Once the session object is bound (AAM enabled) or associated (AAM disabled) and has ADH disabled, the session must explicitly issue dialog handling operations for each mobile application service request or response. This bound or associated session can only be used to send dialog handling primitives. The following can be sent/received using this type of bound or associated session:

- *Receive()* (`cc_receive()`/'`CC_OPEN_IND`')
- *Receive()* (`cc_receive()`/'`CC_ACCEPT_CNF`')
- *Receive()* (`cc_receive()`/'`CC_REFUSE_CNF`')
- *Open()* (`cc_open()`)
- *Accept()* (`cc_accept()`)
- *Refuse()* (`cc_refuse()`)

The other attributes that relate to dialog handling are specified within the **Open-Argument**, **Accept-Result** or **Refuse-Result** objects that are passed to, or returned from, *Open()*, *Accept()*, *Refuse()*, or *Receive()*.

3.2.6 Dialog Session

Once a user has created a bound or associated session that has ADH disabled, a dialog can be created. A dialog is represented by a *fully formed*, or *partially formed dialog*, session object. A *dialog* session is returned as the result of building a new dialog. The dialog session is used, like a bound or associated session, by sending and receiving mobile application service operations. The major difference is that a dialog session object can only be used to send and receive operations within a single dialog with a single remote mobile application entity. After a session forms a dialog, the user can close or abort the dialog, which returns the session to the bound or associated state.

The precedence rules for common parameters within the **Session** and the **Context** objects are different for dialog session objects. Once a session has formed a dialog, the dialog related arguments, *application-Context-Name*, cannot be overridden by the context object.

To terminate this type of session, the user should either abort or close the dialog, which implicitly unbind the session. If the user unbinds the dialog session prior to either closing or aborting the dialog, the service provider will first attempt to close the dialog, and if that is rejected, will abort the dialog.

3.3 Context

The context defines the characteristics of the mobile application interaction that are specific to a particular mobile application operation, but are often used unchanged for many operations. Since the parameters are presumed to be relatively static for a given user during a particular mobile application interaction, these arguments are collected into an OM object of OM class **Context**, which is supplied as the second argument of each mobile application operation. This serves to reduce the number of arguments passed to each function.

The context includes various administrative details, such as the *mode* defined in the Abstract Service, which affect the processing of each mobile application operation. These include a number of

Service Controls and *Local Controls* that allow control over some aspects of the operation. The *Service Controls* include **mode**, **responder-Address**, and **responder-Title**. The *Local Controls* include **asynchronous**, **reply-Limit** and **time-Limit**. Each of these is mapped onto an OM attribute in the **Context**, and they are detailed in Chapter 5 [Interface Class Definitions], page 29.

The effect is as if they were passed as a group of additional arguments on every function call. The value of each component of the context is determined when the interface function is called, and remains fixed throughout the operation.

The precedence rules on address and title of the responder are described in Section 3.1.2 [Names, Addresses and Titles], page 16.

Some of the OM attributes in the **Context** have default values, some of which are locally adminstered. The constant **Default-Context** ('`CC_DEFAULT_CONTEXT`') can be pased as the value of the **Context** argument to the interface functions, and has the same effect as a context OM object created with default values. The context must be a private object, unless it is **Default-Context**.

Detailed specifications of the OM class **Context** are given in Chapter 5 [Interface Class Definitions], page 29.

3.4 Function Arguments

The Abstract Service defines specific arguments for each operation. These are mapped onto coresponding arguments to each interface function (which are aso called input parameters). Although each service has different arguments, some specific arguments recur in several operations; these are briefly introduced here. As far as the ITU-T CC package is concerned, OM classes are defined with a one-to-one mapping to the ASN.1 Abstract Syntax of ITU-T CC. Full details of these and all the other arguments are given in the function definitions in Chapter 4 [Interface Functions], page 27, and the OM class definitions in Chapter 5 [Interface Class Definitions], page 29.

All arguments that are OM objects can generally be supplied to the interface functions as public objects (i.e, descriptor lists) or as private objects. Private objects must be created in the workspace that was returned by *Initialize()*. In some cases, constants can be supplied instead of OM objects.

Note that wherever a function is stated as accepting an instance of a particular OM class as the value of an argument, it will also accept an instance of any subclass of that OM class. For example, the **Service-Req** function has a parameter **argument**, which accepts values of OM class **Service-Argument**. Any of the subclasses of **Service-Argument** may be supplied as the value of **argument**.

Rules for interpretation of '`ANY`' syntax appearing in function arguments are defined in Section 3.4.1 [Encoding and Decoding], page 20.

3.4.1 Encoding and Decoding

XCC specifies two alternatives for encoding and decoding of Mobile Application Packages OM-Attribute values of type '`ANY`', or any OM-Attribute values in a Mobiled Application Services package.

1. The encoding and decoding functionality can be provided internally with the XCC API, without requiring the application to invoke any encoding or decoding functions. This option allows the application to be free from any knowledge of encoding rules. In this case, the OM class and attribute type and corresponding representation are defined in a mobile application or services package. The XCC API uses the package definition to attempt encoding or decoding; if automatic decoding fails, an OM String(Encoding) is used.
2. The application can perform encoding and decoding itself. This option gives the application responsibility and control over the encoding and decoding of OM attributes. In this case, all OM attribute values appear as an OM String(Encoding).

The encoding and decoding alternative to be used is negotiated through the *Negotiate()* function; See ⟨undefined⟩ [⟨undefined⟩], page ⟨undefined⟩.

The XCC API does not specify the use of OM-Encode or OM-Decode for the OM classes defined in this specification, or in mobile application or services packages used with this specification.

To ensure interoperability, the sender and receiver must follow the same encoding rules when converting between OM syntax and encoded syntax. If an algorithm is used to generate OM packages, then the algorithm must ensure that the generated OM syntax is consistent with the input abstract syntax (that is, the same encoded values must result from applying the encoding rules to either representation). The encoding rules used with the ITU-T CC and ANSI CC packages defined by this specification are ANS.1 BER. This does not imply that other encoding rules cannot be used with other packages defined in the future.

For the API to encode and decode the OM attribute values according to the ASN.1 standard scheme, ASN.1 taggin information must be stored for each OM object and each OM attribute. Thus, the package definitions in the workspace need to incorporate the ASN.1 tagging information for each OM object and each OM attribute definition for all Mobile Application Services packages.

As a minimum, the following requirements apply:

- All rules specified in ISO/IEC 8825 – Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1) shall be adhered to. Any exceptions or restrictions must be stated.
- ASN.1 tagging information must be retained for each OM object and each OM attribute in the Mobile Application Services packages.
- The specified encoding and decoding scheme (and any implementation thereof) should be extensible to accomodate the new encoding rules established subsequent to ISO/IEC 8825.

3.4.2 Argument and Response

Most operations and notifications take an argument to specify the argument of the operation and a response when issuing the response of the operation. These arguments and responses are specified to accept values of OM classes that are consistent with the abstract service view (ITU-T CC or ANSI CC) of the current operation.

The argument for a *Service-req()* function is represented by an instance of the OM Class **ITUT-Service-Req-Argument** for a ITU-T CC operation or an instance of the OM Class **ANSI-Service-Req-Argument** for an ANSI CC operation.

The reponse for a *Service-rsp()* function is represented by an instance of the OM Class **GSM-Service-Result**, **GSM-Linked-Reply-Argument Service-Error** or **GSM-Service-Reject** to represent the possible responses of the ITU-T service request operation, or an instance of the **ANSI-Service-Result**, **ANSI-Linked-Reply-Argument Service-Error** or **ANSI-Service-Reject** to represent the possible responses of the ANSI service request operation.

3.5 Function Results

All functions return a **Status** (which is the C function result). Most return an **Invoke-ID** which identfies the particular invocation. The confirmed operations each return a **Result**. (The **Invoke-ID** and **Result** are returned using pointers that are supplied as arguments of the C function). These three kinds of function results are introduced below.

All OM objects returned by interface functions (results and errors) will be private objects in the workspace assocaited with the session private object.

3.5.1 Invoke-ID

All interface functions that invoke a mobile application service operation return an **Invoke-ID**; an integer that identifies the particular invocation of an operation. The **Invoke-ID** is only relevant for asynchronous confirmed operations and may be used later to receive the **Status** and **Result**, or to abandon them. The **Invoke-ID** is also used to respond to a previously requested confirmed operation. Asynchronous operations are fully described in Section 3.6 [Synchronous and Asynchronous Operation], page 23. The interface functions that can be used to start them are the *Service-req()* function.

The numerical value of the invoke-Id returned from a call that successfully invoke an asynchronous confirmed operation is guaranteed to be unique amoungst all outstanding operations in given session. The value is such as could be returned from TCAP, the Transaction Capabilities Application Part defined in ITU-T Recommendation Q.771 through Q.775. Invoke IDs used by XCC are not necessarily those that are actually sent via a protocol such as TCAP. Invoke IDs may be mapped or altered by the Mobile Application Service provider.

The value returned for a synchronous operation or an asynchronous non-confirmed operation is unspecified, as is that for a call that fails to invoke an operation.

3.5.2 Result

Functions invoking confirmed mobile application service operations return a result only if they succeed. All errors from these functions are reported in the **Status** described below, as are errors from all other functions.

The value returned by a function call that invokes an asynchronous operation is unspecified, as is that for a call that fails to invoke an operation. The result of an asynchronous operation is returned by a later call to *Receive()*.

The result of a function invoking a confirmed operation can be composed of a single reply, or of multiple linked replies. In the later case, the term *partial result* is used to designate one of these linked replies. Only a confirmed **Service-req** may produce multiple results. Multiple replies to a single mobile application service operation may only occur if the invoker specifies multiple-reply in the functional unit attribute of the Session object.

In asynchronous mode, the partial results can be retrieved one at a time by subsequent calls to *Receive()*, which each time returns an instance of OM class **Linked-Reply-Argument**. In synchronous mode, the function returns an instance of OM class **Multiple-Reply**, which contains a list of sub-objects of OM class **Linked-Reply-Argument**.

The result (or partial result) of an operation is returned in a private object whose OM class is appropriate to the particular operation. The format of mobile application service operation results is driven both by the Abstract Service and by the need to provide asynchronous execution of functions. To simplify processing of asynchronous results, the result (or partial result) of a single operation is returned in a single OM obejct (corresponding to the abstract result defined in the Standards). The components of the result (or partial result) of an operation are represented by OM attribute in the operation's result object. All information contianed in the Abstract Service result is made available to the application program. The result (partial result) is inspected using the functions provided in the OSI-Abstract-Data Manipulation API.

Only confirmed mobile application service operations produce results, and each type of operation has a specific OM class of OM object for its result. These OM classes are defined in Chapter 5 [Interface Class Definitions], page 29.

The actual OM class of the result can always be a subclass of that named, to allow flexibility for extensions. Thus, the function *OM-Instance()* should always be used when testing the OM class.

3.5.3 Status

Every interface function returnes a **Status** value, which is either the constant **success** ('`(CC_status)0`' or '`CC_SUCCESS`') or an error. Function call errors are represented as integer constants and grouped in categories of System, Library and Communications as described in Chapter 6 [Errors], page 31.

Additional error information is available for System and Communications errors via the *Get-Last-Error()* function call. Additional error information is available for the **bad-argument** Library error via the *Validate-object()* function call.

A synchronous call with multiple linked replies is considered successful unless the reploy limit or time limit is exceeded. The function returns a **Status** value equal to success, and the argument *Result* is an OM object of OM class **Multiple-Reply**, which contains all the linked replies.

It should be noted that OM object **Linked-Reply-Argument** may contain an OM attribute that reflects an error.

If the reply limit or time limit is exceeded, the synchronous call fails and returns a status of the appropriate Library error. However, the *Result* is still considered valid and may contain an OM-Object **Multiple-Reply**, which contains all the received linked replies. A result of *CC_ABSENT_OBJECT* means no replies were received.

In most cases other results of functions are initialized to Null (*CC_ABSENT_OBJECT*) if the status does not have the value **success**. However, the *Result* is still considered valid and may contain an OM-Object of parital replies. A result of *CC_ABSENT_OBJECT* means no replies were received.

3.6 Synchronous and Asynchronous Operation

The asynchronous or synchronous mode of a requested operation is specified at the interface, and determined for each operation by the value of the OM attribute *Asynchronous* in the **Context** passed to the interface function. The default value of this OM attribute is **false**, causing all operations to by synchronous. Support for both synchronous and asynchronous operation is mandatory. There is a limit to the number of pending asynchronous operations; this limit is given by the constant **max-outstanding-operations**, and has a minimum value of 10.

In synchronous mode, all functions wait until the operation is complete before returning. Thus the thread of control is blocked within the interface after calling a function, and the application can make use of the result immediately after the function returns.

In asynchronous mode, some functions return before the operation is complete. The application is then able to continue with other processing while the operation is being executed by the Mobile Application Service provider, and can then access the result by calling *Receive()*. An application may initiate several concurrent asynchronous operations on the ame session before receiving any of the results, subject to the limit described below. The results are not guaranteed to be returned in any particular order. The functions that can execute asynchronously are the *Service-req()* function. This correponds to the mobile application services of the Standards that operate in a confirmed mode. Moreover, only confirmed operations return service results.

An asynchronous function call of a confirmed service returns an **Invoke-ID** of the operation to the application. The same **Invoke-ID** will be returned by *Receive()* on the corresponding result.

An **Invoke-ID** is also returned by *Receive()* on an indication of an invoked mobile application service operation. The same **Invoke-ID** will be used to respond to this operation.

Implementations of the interface are free to return from asynchronous function calls as soon as possible or may wait until the operation has been submitted to the unerlying Mobile Application Service provider. The actual policy used is implemetnation-defined.

Implementations will define a limit to the number of asyncrhonous operations that may be outstanding at any one time on any one session. An asynchronous operation is outstanding from the time

that the function is called until the last reply of the result is returned by *Receive()*, or the operation is abandoned by *Abandon()*, or the session is closed by *Unbind()*. The limit is given by the constant **max-outstanding-operations** ('`CC_MAX_OUTSTANDING_OPERATIONS`') and is at least 10 for conformant XCC implemetnations. While this number of operations is outstanding, attempts to invoke further asynchronous operations will report a **Library-Error** (too many operations).

Asynchronous operation calls can be aborted by executing an *Abandon()* or *Unbind()* call. In this case, the operation is no longer outstanding and the result will never be returned by further *Receive()* function calls.

If an error is detected before an asynchronous request is submitted to the Mobile Application Service provider, the function will return immediately and there will be no outstanding operation generated. Other errors are notified later by *Receive()*, when the result of the outstanding asynchronous confirmed operation is returned. All errors occurring during a synchronous request are reported when the function returns. Full details of error handling are given in Chapter 6 [Errors], page 31.

Where vendors provide suitable system primitives (such as System V `poll(2s)`, or BSD `select(2)`), applications can obtain a file descriptor from the **Session** by inspecting the value fo the OM attribute *File-Descriptor*. Applications may use the file descriptor to suspend the process until data is received on the particular file descriptor.

Applications should ensure that there are no outstanding asynchronous operations on a session when *Unbind()* is called on that session. Once *Unbind()* has been called there is no way to determine whether any outstanding operations succeed or even whether they were ever sent to the Mobile Application Service provider. Also no errors or results of any kind will be reported to the application. It is strongly recommended that *Receive()* is called repeatedly until **Completion-Flag** takes the value **nothing**.

3.7 Other Features

These features are not part of the interface itself, but are mandatory when specified by the Mobile Application Service provider.

The Mobile Applications are not restricted to those defined by ITU-T CC.

All the features listed below are for the most part necessary for ease of use in a mobile application environment. These features are classified as given registered identifiers (Object Identifier). They can be negotiated using the *Negotiate()* function in the same manner as packages. Other types of information that are critical in servicing an environment that incudes implemetnation from multiple vendors on various machines can also be classified and handled with the *Negotiate()* function. Features defined by this specification are described and assigned Object Identifiers in Chapter 4 [Interface Functions], page 27.

3.7.1 Automatic Association Management

When the Mobile Application Services provider makes use of association oriented communication services, such as TCAP, the Mobile Application Service provider implementations are assumed to provide automatic handling of the association between mobile application entities, establishing and releasing associations at its discretion. Such management is intended to bring benefits such as reduced communication charges. To allow this flexibility to the implementation, the interface does not specify when communication takes place. Automatic Association Management (AAM) may be enabled or disabled on a per-workspace basis using the *Negotiate()* function.

3.7.2 Automatic Dialog Handling

When the Mobile Application Services provider makes use of dialog oriented communication services, such as that provided by TCAP, the Mobile Application Service provider implementations are

assumed to provide automatic handling of dialogs between mobile application entities, establishing and releasing dialogs at its discretion. Such management is intended to bring benefits such as reduced communication overheads. To allow this flexibility to the implementation, the interface does not specify when communication takes place. Automatic Dialog Handling (ADH) may be enabled or disabled on a per-workspace basis using the *Negotiate()* function.[7]

3.7.3 Automatic Performer Resolution

The performer of an invoked operation my be explicitly designated by the responder name and responder address parameters of the bound session used.

However, in the case where the responder is specified as a wildcard, the Mobile Application Service provider may be assumed to provide automatic mobile application service and application context to consumer resolution: to find out the consumer that is in charge of the selected mobile application service specified in the mobile application service operation.

3.7.4 Responder Versatility

Responder versatility is the ability to change the consumer within a same bound-session at each function call. It is useful when the automatic consumer resolution is either not suported by the Mobile Application Service provider or note requested. This applies if the underlying Mobile Application Service provider is connection-less.

3.7.5 Automatic Name to Address Resolution

Mobile Application Service provider implementation may provide automatic resolution between program name and address to find the network address of a mobile application entity from its name using directory or translation services.

3.7.6 Automatic Dispatching to Appropriate Stack

The Mobile Application Services provideer implementation may provide a loop back facility if the destination of the operation or notification is local. It also may provide routing of the mobile application services operation to the proper underlying communications stack according to the implied mobile application service and the destination (for example over a ITU-T SCCP stack or an ANSI SCCP stack).

3.8 Function Sequencing

A minimum set of sequencing rules applies when using the interface to exchange mobile application service information between mobile application programs acting as a mobile application entity. These rules need to be respected by mobile application programs to ensure that interface functions are called in the proper sequence and that the state of the interface is not violated, otherwise **Library-error** status will be returned.[8]

The general rules to follow are:

1. Initialize a workspace (‘`cc_initialize()`’)
2. Negotiate features of the interface (‘`cc_negotiate()`’)
3. Open one or several sessions (‘`cc_bind()`’)

[7] Note that Automatic Dialog Handling is an independent concept from Automatic Association Management.

[8] Note the following is considered as tutorial information. The definitive information is contained in the Standards (see referenced documents).

4. Perform mobile application service interactions (operations) using the offered interface functions. An interaction is identified by its **Invoke-Id**.
5. Close the opened sessions (‘`cc_unbind()`’)
6. Discard the workspace (‘`cc_shutdown()`’)

Seven states are defined in the interface to cover both interface service operations and mobile application service interactions:

UNINIT	Workspace uninitialized.
INIT	Workspace initialized.
UNBND	Session closed.
BND	Session opened.
IDLE	Outstanding operatoin requested in a mobile application service interaction.
OUTOP	Operation indication received in a mobile application service interaction.
OPIND	

4 Interface Functions

5 Interface Class Definitions

6 Errors

Appendix A C Headers

A.1 `xcc.h`

A.2 xcc_isup.h

A.3 `xcc_isdn.h`

A.4 xcc_h323.h

A.5 `xcc_sipt.h`

Appendix B Examples

Glossary

Licenses

All code presented in this manual is licensed under the [GNU Affero General Public License], page 43. The text of this manual is licensed under the [GNU Free Documentation License], page 53, 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