
**Information technology —
Telecommunications and information
exchange between systems — Protocol
for Computer Supported
Telecommunications Applications (CSTA)
Phase III**

*Technologies de l'information — Télécommunications et échange
d'information entre systèmes — Protocole pour applications en
télécommunications supportées par ordinateur (CSTA) en phase III*

Reference number
ISO/IEC 18052:2000(E)



PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

© ISO/IEC 2000

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Contents

Foreword	x
Introduction	xi
1 Scope	1
2 Normative References	1
3 Conformance	2
3.1 Static Requirements	2
3.2 Dynamic Requirements	2
3.3 PICS Requirement	2
4 Definitions and Abbreviations	2
5 CSTA Service Definition Model	2
5.1 CSTA Application Layer Structure	2
5.2 Remote Operations	2
5.3 The CSTA Service Response	3
5.4 Cross Referencing of Event Reports	3
5.5 Handling of Private Data	3
6 Interconnection service boundary	3
7 Security	3
8 Association Management	4
8.1 Implicit association	4
8.2 Dynamic association management using ACSE	4
8.2.1 Encoding of application context name	4
8.2.2 Encoding of application context information	4
9 CSTA parameter types	8
9.1 Switching function objects	8
9.2 Device Identifiers	9
9.3 Call and connection identifiers	11
9.4 Connection states	12
9.5 Status reporting	13
9.6 Device and feature types and other parameters	15
9.7 Security	20
9.8 Common extensions	21
9.9 Call control	22
9.10 Capability Exchange	24
9.11 Call Detail Record	73
9.12 Charge information	75
9.13 Data call types	76
9.14 Escape types	77
9.15 Media services	78
9.16 Physical device features	80
9.17 Data Collection	82
9.18 Event Cause	83
9.19 Error Value	85

10	Event Report Service	90
11	Capability exchange services	95
11.1	Services	95
11.1.1	Get logical device information	95
11.1.2	Get physical device information	97
11.1.3	Get switching function capabilities	98
11.1.4	Get switching function devices	101
11.1.5	Switching function devices	102
12	System services	103
12.1	Registration services	103
12.1.1	Change system status filter	103
12.1.2	System register	104
12.1.3	System register abort	105
12.1.4	System register cancel	106
12.2	Services	107
12.2.1	Request system status	107
12.2.2	System status	108
12.2.3	Switching function capabilities changed	109
12.2.4	Switching function devices changed	110
13	Monitoring services	111
13.1	Services	111
13.1.1	Change monitor filter	111
13.1.2	Monitor start	112
13.1.3	Monitor stop	113
14	Snapshot services	114
14.1	Services	114
14.1.1	Snapshot call	114
14.1.2	Snapshot device	116
14.1.3	Snapshot calldata	117
14.1.4	Snapshot devicedata	118
15	Call control services and events	119
15.1	Services	119
15.1.1	Accept call	119
15.1.2	Alternate call	120
15.1.3	Answer call	121
15.1.4	Call back call-related	122
15.1.5	Call back message call-related	123
15.1.6	Camp on call	124
15.1.7	Clear call	125
15.1.8	Clear Connection	126
15.1.9	Conference call	127
15.1.10	Consultation call	128
15.1.11	Deflect call	129

15.1.12	Dial digits	130
15.1.13	Directed pickup call	131
15.1.14	Group pickup call	132
15.1.15	Hold call	133
15.1.16	Intrude call	134
15.1.17	Join call	135
15.1.18	Make call	136
15.1.19	Make predictive call	137
15.1.20	Park call	139
15.1.21	Reconnect call	140
15.1.22	Retrieve call	141
15.1.23	Single step conference call	142
15.1.24	Single step transfer call	143
15.1.25	Transfer call	144
15.2	Events	145
15.2.1	Bridged	145
15.2.2	Call cleared	146
15.2.3	Conferenced	147
15.2.4	Connection cleared	148
15.2.5	Delivered	149
15.2.6	Digits dialed	150
15.2.7	Diverted	151
15.2.8	Established	152
15.2.9	Failed	153
15.2.10	Held	154
15.2.11	Network capabilities changed	155
15.2.12	Network reached	156
15.2.13	Offered	157
15.2.14	Originated	158
15.2.15	Queued	159
15.2.16	Retrieved	160
15.2.17	Service initiated	161
15.2.18	Transferred	162
16	Call associated features	163
16.1	Services	163
16.1.1	Associate data	163
16.1.2	Cancel telephony tones	164
16.1.3	Generate digits	165
16.1.4	Generate telephony tones	166
16.1.5	Send user information	167
16.2	Events	168
16.2.1	Call information	168
16.2.2	Charging	169
16.2.3	Digits generated	170
16.2.4	Telephony tones generated	171
16.2.5	Service completion failure	172
17	Media attachment services and events	173
17.1	Services	173

17.1.1	Attach media service	173
17.1.2	Detach media service	174
17.2	Events	175
17.2.1	Media attached	175
17.2.2	Media detached	176
18	Routeing services	177
18.1	Registration services	177
18.1.1	Route register	177
18.1.2	Route register abort	178
18.1.3	Route register cancel	179
18.2	Services	180
18.2.1	Re-Route	180
18.2.2	Route end	181
18.2.3	Route reject	182
18.2.4	Route request	183
18.2.5	Route select	184
18.2.6	Route used	185
19	Physical device features	186
19.1	Services	186
19.1.1	Button press	186
19.1.2	Get auditory apparatus information	187
19.1.3	Get button information	188
19.1.4	Get display	189
19.1.5	Get hookswitch status	190
19.1.6	Get lamp information	191
19.1.7	Get lamp mode	192
19.1.8	Get message waiting indicator	193
19.1.9	Get microphone gain	194
19.1.10	Get microphone mute	195
19.1.11	Get ringer status	196
19.1.12	Get speaker mute	197
19.1.13	Get speaker volume	198
19.1.14	Set button information	199
19.1.15	Set display	200
19.1.16	Set hookswitch status	201
19.1.17	Set lamp mode	202
19.1.18	Set message waiting indicator	203
19.1.19	Set microphone gain	204
19.1.20	Set microphone mute	205
19.1.21	Set ringer status	206
19.1.22	Set speaker mute	207
19.1.23	Set speaker volume	208
19.2	Events	209
19.2.1	Button information	209
19.2.2	Button press	210
19.2.3	Display updated	211

19.2.4	Hookswitch	212
19.2.5	Lamp mode	213
19.2.6	Message waiting	214
19.2.7	Microphone gain	215
19.2.8	Microphone mute	216
19.2.9	Ringer status	217
19.2.10	Speaker mute	218
19.2.11	Speaker volume	219
20	Logical device features	220
20.1	Services	220
20.1.1	Call back non-call-related	220
20.1.2	Call back message non-call-related	221
20.1.3	Cancel call back	222
20.1.4	Cancel call back message	223
20.1.5	Get agent state	224
20.1.6	Get auto answer	225
20.1.7	Get auto work mode	226
20.1.8	Get caller id status	227
20.1.9	Get do not disturb	228
20.1.10	Get forwarding	229
20.1.11	Get last number dialed	230
20.1.12	Get routeing mode	231
20.1.13	Set agent state	232
20.1.14	Set auto answer	233
20.1.15	Set auto work mode	234
20.1.16	Set caller id status	235
20.1.17	Set do not disturb	236
20.1.18	Set forwarding	237
20.1.19	Set routeing mode	238
20.2	Events	239
20.2.1	Agent busy	239
20.2.2	Agent logged off	240
20.2.3	Agent logged on	241
20.2.4	Agent not ready	242
20.2.5	Agent ready	243
20.2.6	Agent working after call	244
20.2.7	Auto answer	245
20.2.8	Auto work mode	246
20.2.9	Call back	247
20.2.10	Call back message	248
20.2.11	Caller id status	249
20.2.12	Do not disturb	250
20.2.13	Forwarding	251
20.2.14	Routeing mode	252
21	Device maintenance events	253
21.1	Events	253
21.1.1	Back in service	253
21.1.2	Device capabilities changed	254
21.1.3	Out of service	255

22	I/O services	256
22.1	Registration services	256
22.1.1	I/O register	256
22.1.2	I/O register abort	257
22.1.3	I/O register cancel	258
22.2	Services	259
22.2.1	Data path resumed	259
22.2.2	Data path suspended	260
22.2.3	Fast data	261
22.2.4	Resume data path	262
22.2.5	Send broadcast data	263
22.2.6	Send data	264
22.2.7	Send multicast data	265
22.2.8	Start data path	266
22.2.9	Stop data path	267
22.2.10	Suspend data path	268
23	Data Collection Services	269
23.1	Services	269
23.1.1	Data Collected	269
23.1.2	Data Collection Resumed	271
23.1.3	Data Collection Suspended	272
23.1.4	Resume Data Collection	273
23.1.5	Start Data Collection	274
23.1.6	Stop Data Collection	275
23.1.7	Suspend Data Collection	276
24	Voice unit services and events	277
24.1	Services	277
24.1.1	Concatenate message	277
24.1.2	Delete message	278
24.1.3	Play message	279
24.1.4	Query voice attribute	280
24.1.5	Record message	281
24.1.6	Reposition	282
24.1.7	Resume	283
24.1.8	Review	284
24.1.9	Set voice attribute	285
24.1.10	Stop	286
24.1.11	Suspend	287
24.1.12	Synthesize message	288
24.2	Events	289
24.2.1	Play	289
24.2.2	Record	290
24.2.3	Review	291
24.2.4	Stop	292
24.2.5	Suspend play	293
24.2.6	Suspend record	294

24.2.7	Voice attribute changed	295
25	Call detail record services	296
25.1	Services	296
25.1.1	Call detail records notification	296
25.1.2	Call detail records report	297
25.1.3	Send stored call detail records	298
25.1.4	Start call detail records transmission	299
25.1.5	Stop call detail records transmission	300
26	Vendor specific extensions services and events	301
26.1	Registration services	301
26.1.1	Escape register	301
26.1.2	Escape register abort	302
26.1.3	Escape register cancel	303
26.2	Services	304
26.2.1	Escape	304
26.2.2	Private data version selection	305
26.3	Events	306
26.3.1	Private event	306
Annex A	Protocol Implementation Conformance Statement (PICS) Proforma	307

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 18052 was prepared by ECMA (as ECMA-285) and was adopted, under a special "fast-track procedure", by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, in parallel with its approval by national bodies of ISO and IEC.

Annex A forms a normative part of this International Standard.

Introduction

This International Standard defines Phase III of Protocol for Computer Supported Telecommunications Applications (CSTA) for OSI Layer 7 communication between a computing network and a telecommunications network. This International Standard is part of a Suite of Standards and Technical Reports for Phase III of CSTA. All of the Standards and Technical Reports in the Suite are based on practical experience of ECMA member companies and each one represents a pragmatic and widely-based consensus.

The evolution of this Suite began with CSTA Phase I, which included the CSTA Services and Protocol Standards (ECMA-179 and ECMA-180). In Phase II, Technical Report ECMA TR/68 was added illustrating how CSTA services and events may be used in typical call scenarios.

Phase III of CSTA extends the previous Phase II Standards (ECMA-217 and ECMA-218) in major theme directions as well as numerous details. This incorporates technology based upon the *versit* CTI Encyclopedia (Version 1.0), which was contributed to ECMA by *versit*. Major areas of advancement include:

- New categories of services and events such as capabilities exchange, charging, media attachment services, call data recording (CDR), etc.
- Additional services and events for call and device control.
- Enhancement to existing services and events.
- Organization of services and events to reflect a grouping based on function (call control, device control, etc.).

Information technology - Telecommunications and information exchange between systems - Protocol for Computer Supported Telecommunications Applications (CSTA) Phase III

1 Scope

This International Standard specifies application protocol data units (APDUs) for the services described in ISO/IEC 18051, *Information technology - Telecommunications and information exchange between systems - Services for Computer Supported Telecommunications Applications (CSTA) Phase III*.

Clause 5 to clause 7 inclusive describes the concepts underlying the Remote Operations model, notation and service.

Clause 8 to clause 26 inclusive contains CSTA-specific protocol details and forms the main part of this International Standard.

The protocol in this International Standard operates in the context of an application association.

2 Normative References

The following normative documents contain provisions which, through reference in this text, constitute provisions of this International Standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of IEC and ISO maintain registers of currently valid International Standards.

- ISO/IEC 8649:1996,** *Information technology - Open Systems Interconnection - Service definition for the Association Control Service Element* (this corresponds to ITU-T Rec. X.217 1995).
- ISO/IEC 8650-1:1996,** *Information technology - Open Systems Interconnection - Connection-oriented protocol for the Association Control Service Element: Protocol specification* (this corresponds to ITU-T Rec. X.227 4/95).
- ISO/IEC 9545:1994,** *Information technology - Open Systems Interconnection - Application Layer structure.*
- ISO/IEC 13712-1:1995,** *Information technology - Remote Operations: Concepts, model, and notation* (this corresponds to ITU-T Rec. X.880, 1994).
- ISO/IEC 13712-2:1994,** *Information technology - Remote Operations: OSI realisations - Remote Operations Service Element (ROSE) service definition* (this corresponds to ITU-T Rec. X.881, 1994).
- ISO/IEC 13712-3:1994,** *Information technology - Remote Operations: OSI realisations - Remote Operations Service Element (ROSE) protocol specification* (this corresponds to ITU-T Rec. X.882, 1994).
- ISO/IEC 18051:2000,** *Information technology - Telecommunications and information exchange between systems - Services for Computer Supported Telecommunications Applications (CSTA) Phase III.*
- ISO/IEC TR 18053:2000,** *Information technology - Telecommunications and information exchange between systems - Glossary of definitions and terminology for Computer Supported Telecommunications Applications (CSTA) Phase III.*

- ITU-T X.680, *Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation.*
- ITU-T X.690, *Information technology - ASN.1 encoding rules - Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).*

3 Conformance

A manufacturer may select any part of the CSTA protocol, as specified in this International Standard, for implementation on a system as long as it satisfies the minimum conformance requirements as specified in clause 3 of ISO/IEC 18051.

A Protocol Implementation Conformance Statement (PICS) shall be used to specify the operations which are provided by a particular implementation. A PICS shall also specify the parameter options which are used.

3.1 Static Requirements

To conform to this International Standard, a system shall support the transfer syntax (derived from the encoding rules specified in ITU-T X.690) named {joint-iso-ccitt asn1(1) basic-encoding(1)}; for the purpose of generating and interpreting CSTA protocol information as defined by the abstract syntax “CSTA-ASN.1-Object-Descriptor” for the operations supported.

3.2 Dynamic Requirements

To conform to this International Standard, a system shall:

- a. follow the procedures as specified in this International Standard, and ISO/IEC 18051, relevant to each CSTA operation that the system claims to implement; and
- b. satisfy the definitions, as specified in ISO/IEC 18051, relevant to each CSTA service that the system claims to implement.

3.3 PICS Requirement

To conform to this International Standard, the following shall be stated by the implementer when defining a PICS corresponding to an application or implementation:

- a. which CSTA operations, as defined in ISO/IEC 18051, are supported by the system for the particular implementation; and
- b. which optional parameters are supported by the PDUs belonging to the supported operations.

A PICS proforma is given in Annex A of this International Standard.

4 Definitions and Abbreviations

CSTA-specific terminology is defined in ISO/IEC TR 18053. For the purposes of this International Standard, the following additional definitions, defined in other standards, shall apply:

- Remote Operations (as per ISO/IEC 13712-1)
- Application Association (as per ISO/IEC 8649)
- Application Context (as per ISO/IEC 8649)

5 CSTA Service Definition Model

5.1 CSTA Application Layer Structure

The CSTA Application Layer structure conforms to the model described in ISO/IEC 9545.

5.2 Remote Operations

The services of CSTA are modeled as Remote Operations as described in ISO/IEC 13712-1. Typically, one entity requests that a particular operation be performed; the other entity attempts to perform the operation and responds to the requestor. Consequently the operation of the protocol is an elementary request/reply interaction, supported within the OSI Application Layer, and carried out within the context of an application association.

For some of the CSTA services, the entity to which the request is sent generates a reply which can indicate success or failure. For these services, CSTA shall adopt the operations Class 2, defined in ISO/IEC 13712-2 as:

- Asynchronous, reporting success or failure (result or error).

For some of the CSTA services, the entity to which the request is sent generates a reply which can only indicate failure. For these services, CSTA shall adopt the operations Class 3, defined in ISO/IEC 13712-2 as:

- Asynchronous, reporting failure (error) only, if any.

For some of the CSTA services, particularly the ongoing reporting of events, no reply is generated. For these services, CSTA shall adopt the operations Class 5, defined in ISO/IEC 13712-2 as:

- Asynchronous, outcome not reported.

The protocol description for the particular service defines the relevant class of the operation used for that service.

CSTA shall correlate the single response, denoting success or failure, with the originating request by using the mechanisms within the ROSE protocol.

5.3 The CSTA Service Response

CSTA employs a generic response mechanism which is, in principle, decoupled from the specifics of the switching activity. The following points describe the operation of the CSTA service response:

- Specific services may have an unconfirmed mode where responses to correct requests are not returned.
- The server shall check the correctness of the request (e.g. syntactical checks) before issuing the response. Incorrect requests shall result in an error response, even in the unconfirmed mode.

5.4 Cross Referencing of Event Reports

A computer application process may need to cross reference a CSTAEventReport to one of the following:

- a CSTA Object ID (Call ID or Device ID),
- an earlier Monitor request; or
- one of many Monitor requests (pertaining to the same CSTA Object).

For the above scenarios the necessary cross referencing function may be fulfilled by use of the parameter "MonitorCrossRefID". The content of MonitorCrossRefID depends upon the context and it may be one of the following: Call ID, Device ID or another independently switch managed static identifier. The independent identifier may have a unique correlation to either: one device, one call, or one monitor request.

The switching system limit on the number of monitor requests on one CSTA Object (Call or Device) is an implementation option. This International Standard does not stipulate how many monitor requests on one object are to be supported by the switch. If using Static Device or Call identifiers the limit can only be one.

5.5 Handling of Private Data

If an entity receives the parameter CSTAPrivateData, and it can not recognize the information contained, the parameter shall be discarded, and the rest of the message shall be processed.

6 Interconnection service boundary

The protocol in this International Standard is an OSI Application Layer protocol and uses the Remote Operations protocol defined in ISO/IEC 13712-3. The Remote Operations protocol assumes certain services are provided by the underlying layers, and these services are also assumed by the protocol for CSTA.

7 Security

This protocol also provides a mechanism for secure transmission of CSTA PDUs as defined in this International Standard.

8 Association Management

The protocol in this International Standard operates in the context of an application association. This application association can be either:

- an implicit association achieved via off-line agreement; or
- a dynamically negotiated association realized through the use of ACSE.

8.1 Implicit association

An a-priori agreement exists between switching and computing functions: the application context is implicit, dynamic negotiation is not possible.

8.2 Dynamic association management using ACSE

ACSE requires the use of an application context Name (as defined by ISO 8649). This is an object id that uniquely identifies CSTA independently of the different versions of the protocol.

The CSTA protocol version information is carried within the User Information field of the A-ASSOCIATE request and response PDUs.

An application context is established (using ACSE) as follows:

- the system generating the A-ASSOCIATE request includes the CSTA application context name and a list of all protocol versions that it is prepared to offer in the User Information field;
- on receipt of the A-ASSOCIATE request, the receiving system selects the protocol version to be used by identifying the highest version that is common to both systems;
- the protocol version selected is conveyed to the requestor in the User Information field of the A-ASSOCIATE response.

In addition to negotiating the protocol version, it is necessary for the requesting and responding systems to specify the CSTA services that they support. As with the protocol version information, this is also achieved by carrying additional information in the User Information field of the A-ASSOCIATE request and response PDUs. The application association requestor shall:

- list the services required from the serving application;
- list the services it can supply.

The responder shall include similar information for the responding application. At this point the association requestor will either accept or reject the association.

8.2.1 Encoding of application context name

```
CSTA-application-context-name
{ iso( 1) identified-organization( 3) icd-ecma( 12)
  standard( 0) csta( 218) }
-- Common to all CSTA protocol versions
```

8.2.2 Encoding of application context information

```
CSTA-application-context-information-csta3
{ iso( 1) identified-organization( 3) icd-ecma( 12)
  standard( 0) csta3( 285) application-context-information( 200) }
```

```
DEFINITIONS ::=
BEGIN
```

```
IMPORTS
```

```
CSTAF functionality FROM CSTA-application-context-information
{ iso( 1) identified-organization( 3) icd-ecma( 12)
  standard( 0) csta2( 218) application-context-information( 200) };
```

```
ACSEUserInformationForCSTA ::= CHOICE
{ oldDefinition
  newDefinition [0] IMPLICIT
  OldACSEUserInformationForCSTA,
  NewACSEUserInformationForCSTA }
```

```

OldACSEUserInformationForCSTA ::= SEQUENCE
{
    CSTAVersion                      CSTAVersion,
    CSTAFunctionsRequiredByApplication CSTAFunctionality,
    CSTAFunctionsThatCanBeSupplied    CSTAFunctionality
}

NewACSEUserInformationForCSTA ::= SEQUENCE
{
    CSTAVersion                      CSTAVersion,
    --
    --Note that the following two parameters are defined in ECMA-218 (CSTA Phase II).
    --They are defined in the context of CSTA Phase II Services and Events for
    --backward compatibility considerations.
    --For CSTA Phase III (and later), implementations should use CSTA Capability Exchange Services
    --to determine the Services and Events supported by a switching function.
    --
    CSTAFunctionsRequiredByApplication [0] IMPLICIT CSTAFunctionality OPTIONAL,
    CSTAFunctionsThatCanBeSupplied     [1] IMPLICIT CSTAFunctionality OPTIONAL,
    CSTAPrivateDataVersionList         [2] IMPLICIT CSTAPrivateDataVersionList OPTIONAL
}

CSTAPrivateDataVersionList ::= SEQUENCE OF INTEGER

CSTAVersion ::= BIT STRING
{
    versionOne      (0), -- CSTA protocol version defined in ECMA-180
    versionTwo      (1), -- CSTA protocol version defined in ECMA-218
    versionThree    (2), -- CSTA protocol version defined in ECMA-285
    versionFour     (3), -- CSTA protocol version defined in ISO/IEC 18052
    versionFive     (4), -- Reserved for future use
    versionSix      (5), -- Reserved for future use
    versionSeven    (6), -- Reserved for future use
    versionEight    (7), -- Reserved for future use
    versionNine     (8), -- Reserved for future use
    versionTen      (9), -- Reserved for future use
    versionEleven   (10), -- Reserved for future use
    versionTwelve   (11), -- Reserved for future use
    versionThirteen (12), -- Reserved for future use
    versionFourteen (13), -- Reserved for future use
    versionFifteen  (14), -- Reserved for future use
    versionSixteen  (15)} -- Reserved for future use

CallControlServices ::= BIT STRING
{
    acceptCall      (0),
    alternateCall   (1),
    answerCall      (2),
    callBack        (3),
    callBackMessage (4),
    campOnCall      (5),
    clearCall       (6),
    clearConnection (7),
    conferenceCall  (8),
    consultationCall (9),
    deflectCall     (10),
    dialDigits      (11),
    directedPickupCall (12),
    groupPickupCall (13),
    holdCall        (14),
    intrudeCall     (15),
    joinCall        (16),
    makeCall        (17),
    makePredictiveCall (18),
    parkCall        (19),
    reconnectCall   (20),
    retrieveCall     (21),
    singleStepConference (22),
    singleStepTransfer (23),
    transferCall     (24)}

CallAssociatedServices ::= BIT STRING
{
    associateData (0),

```

```

        cancelTelephonyTones    (1),
        generateDigits          (2),
        generateTelephonyTones  (3),
        sendUserInformation      (4)}

MediaAttachmentServices ::= BIT STRING
{
    attachMediaService    (0),
    detachMediaService    (1)}

RouteingServices ::= BIT STRING
{
    routeRegister          ( 0),
    routeRegisterCancel    ( 1),
    routeRegisterAbort     ( 2),
    reRoute                ( 3),
    routeEnd               ( 4),
    routeReject            ( 5),
    routeRequest           ( 6),
    routeSelect            ( 7),
    routeUsed              ( 8)}

VoiceUnitServices ::= BIT STRING
{
    concatenateMessage      (0),
    deleteMessage          (1),
    playMessage             (2),
    queryVoiceAttribute      (3),
    recordMessage           (4),
    reposition              (5),
    resume                  (6),
    review                  (7),
    setVoiceAttribute       (8),
    stop                    (9),
    suspend                 (10),
    synthesiseMessage       (11)}

CallControlEvents ::= BIT STRING
{bridged                    (15),
 callCleared                (0),
 conferenced                (1),
 connectionCleared          (2),
 delivered                  (3),
 digitsDialed               (14),
 diverted                   (4),
 established                 (5),
 failed                     (6),
 held                       (7),
 networkCapabilitiesChanged (16),
 networkReached              (8),
 offered                    (17),
 originated                 (9),
 queued                     (10),
 retrieved                  (11),
 serviceInitiated           (12),
 transferred                 (13) }

CallAssociatedEvents ::= BIT STRING
{
    callInformation          (0),
    charging                 (1),
    dTMFDigitsDetected      (2),
    telephonyTonesDetected  (3),
    serviceCompletionFailure (4)}

MediaAttachmentEvents ::= BIT STRING
{
    mediaAttached (0),
    mediaDetached (1)}

PhysicalDeviceFeatureEvents ::= BIT STRING
{buttonInformation (0),
 buttonPress       (1),

```

```

        displayUpdated          (2),
        hookswitch              (3),
        lampMode                 (4),
        messageWaiting           (5),
        microphoneGain           (6),
        microphoneMute           (7),
        ringerStatus             (8),
        speakerMute              (9),
        speakerVolume            (10)}

LogicalDeviceFeatureEvents ::= BIT STRING
{
    agentBusy                    ( 0),
    agentLoggedOn                ( 1),
    agentLoggedOff               ( 2),
    agentNotReady                ( 3),
    agentReady                   ( 4),
    agentWorkingAfterCall        ( 5),
    autoAnswer                   ( 6),
    autoWorkMode                 ( 7),
    callBack                     ( 8),
    callBackMessage              ( 9),
    callerIDStatus               (10),
    doNotDisturb                 (11),
    forwarding                   (12),
    routeingMode                 (13)}

DeviceMaintenanceEvents ::= BIT STRING
{
    backInService                (0),
    deviceCapabilityChanged       (2),
    outOfService                  (1)      }

VoiceUnitEvents ::= BIT STRING
{
    play                         (1),
    record                       (3),
    review                       (5),
    stop                         (0),
    suspendPlay                  (2),
    suspendRecord                (4),
    voiceAttributesChange        (6)      }

VendorSpecEvents ::= BIT STRING
{
    privateEvent                 (0)}

END -- of CSTA-application-context-information

```

9 CSTA parameter types

The major parameters have been assigned distinct application tags to facilitate parsing. The data is defined in logical groups in ascending order of application tag. Application tags used are:

- APPLICATION 1 - 8: Device identifiers
- APPLICATION 11 - 14: Connection identifiers and local connection states
- APPLICATION 21 - 24: . . . Status reporting
- APPLICATION 29 - 30: . . . CSTACommonArguments

9.1 Switching function objects

```

CSTA-switching-function-objects
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) switching-function-objects( 122) }
DEFINITIONS ::=
BEGIN

EXPORTS
CSTAObject;

IMPORTS
DeviceID FROM CSTA-device-identifiers
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) device-identifiers( 123) }
ConnectionID FROM CSTA-call-connection-identifiers
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) call-connection-identifiers( 124) };

CSTAObject ::= CHOICE
    {deviceObject    DeviceID,
     callObject      ConnectionID}

END -- of CSTA-switching-function-objects

```

9.2 Device Identifiers

```

CSTA-device-identifiers
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) device-identifiers( 123) }

DEFINITIONS ::=
BEGIN

EXPORTS
DeviceID, NumberDigits, CallingDeviceID, CalledDeviceID,
SubjectDeviceID, RedirectionDeviceID, AssociatedCallingDeviceID,
AssociatedCalledDeviceID, NetworkCallingDeviceID, NetworkCalledDeviceID;

IMPORTS
MediaCallCharacteristics FROM CSTA-media-services
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) media-services( 136) };

CallingDeviceID ::= [APPLICATION 1] CHOICE
    {deviceIdentifier      DeviceID,
     notKnown              [7] IMPLICIT      NULL}

CalledDeviceID ::= [APPLICATION 2] CHOICE
    {deviceIdentifier      DeviceID,
     notKnown              [7] IMPLICIT      NULL}

SubjectDeviceID ::= [APPLICATION 3] CHOICE
    {deviceIdentifier      DeviceID,
     notKnown              [7] IMPLICIT      NULL}

RedirectionDeviceID ::= [APPLICATION 4] CHOICE
    {numberdialed          DeviceID,
     notKnown              [7] IMPLICIT      NULL,
     notRequired           [8] IMPLICIT      NULL,
     notSpecified          [9] IMPLICIT      NULL}

AssociatedCallingDeviceID ::= [APPLICATION 5] CHOICE
    {deviceIdentifier      DeviceID,
     notKnown              [7] IMPLICIT      NULL}

AssociatedCalledDeviceID ::= [APPLICATION 6] CHOICE
    {deviceIdentifier      DeviceID,
     notKnown              [7] IMPLICIT      NULL}

NetworkCallingDeviceID ::= [APPLICATION 7] CHOICE
    {deviceIdentifier      DeviceID,
     notKnown              [7] IMPLICIT      NULL}

NetworkCalledDeviceID ::= [APPLICATION 8] CHOICE
    {deviceIdentifier      DeviceID,
     notKnown              [7] IMPLICIT      NULL}

DeviceID ::= SEQUENCE
{
    deviceIdentifier      CHOICE
    {
        dialingNumber      [0] IMPLICIT      NumberDigits,
        deviceNumber        [1] IMPLICIT      DeviceNumber,
        implicitPublic       [2] IMPLICIT      NumberDigits,
        explicitPublic       [3] IMPLICIT      PublicTON,
        implicitPrivate      [4] IMPLICIT      NumberDigits,
        explicitPrivate      [5] IMPLICIT      PrivateTON,
        other                [6] IMPLICIT      OtherPlan},
    mediaCallCharacteristics MediaCallCharacteristics OPTIONAL}

PublicTON ::= CHOICE
    {
        unknown            [0] IMPLICIT      IA5String,
        international      [1] IMPLICIT      IA5String,
        national            [2] IMPLICIT      IA5String,

```

```

networkspecific      [3] IMPLICIT      IA5String,
subscriber           [4] IMPLICIT      IA5String,
abbreviated          [5] IMPLICIT      IA5String  }

PrivateTON ::= CHOICE
{
    unknown           [0] IMPLICIT      IA5String,
    level3RegionalNumber [1] IMPLICIT      IA5String,
    level2RegionalNumber [2] IMPLICIT      IA5String,
    level1RegionalNumber [3] IMPLICIT      IA5String,
    pTNSpecificNumber  [4] IMPLICIT      IA5String,
    localNumber        [5] IMPLICIT      IA5String,
    abbreviated        [6] IMPLICIT      IA5String  }

OtherPlan ::= OCTET STRING      -- Allows future expansion to cover other numbering
                                   -- plans

NumberDigits ::= IA5String

DeviceNumber ::= INTEGER

END -- of CSTA-device-identifiers
```

9.3 Call and connection identifiers

```

CSTA-call-connection-identifiers
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) call-connection-identifiers( 124) }

DEFINITIONS ::=
BEGIN

EXPORTS
    CallID, ConnectionID;

IMPORTS
    DeviceID FROM CSTA-device-identifiers
        { iso( 1) identified-organization( 3) icd-ecma( 12)
          standard( 0) csta3( 285) device-identifiers( 123) };

ConnectionID ::= [APPLICATION 11] CHOICE
{
    callID      [0] IMPLICIT      CallID,
    deviceID    [1]                LocalDeviceID,
    both        SEQUENCE
    {
        callID      [0] IMPLICIT      CallID,
        deviceID    [1]                LocalDeviceID
    }
}

CallID ::= OCTET STRING (SIZE(0..8))

LocalDeviceID ::= CHOICE
{
    staticID      DeviceID,
    dynamicID     [3] IMPLICIT      OCTET STRING (SIZE(0..32)) }

END -- of CSTA-call-connection-identifiers

```

9.4 Connection states

```

CSTA-connection-states
{ iso( 1) identified-organization( 3) icd-ecma( 12)
  standard( 0) csta3( 285) connection-states( 125) }

DEFINITIONS ::=
BEGIN

EXPORTS
  ConnectionList, LocalConnectionState;

IMPORTS
  ConnectionID FROM CSTA-call-connection-identifiers
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) call-connection-identifiers( 124) }
  DeviceID FROM CSTA-device-identifiers
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) device-identifiers( 123) }
  ConnectionInformation FROM CSTA-media-services
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) media-services( 136) };

ConnectionList ::= SEQUENCE OF SEQUENCE
{
  newConnection [0] ConnectionID OPTIONAL,
  oldConnection [1] ConnectionID OPTIONAL,
  endpoint [2] CHOICE
  {
    deviceID DeviceID,
    notKnown NULL} OPTIONAL,
  associatedNID [3] CHOICE
  {
    deviceID DeviceID,
    notKnown NULL} OPTIONAL,
  resultingConnectionInfo ConnectionInformation OPTIONAL}

LocalConnectionState ::= [APPLICATION 14] IMPLICIT ENUMERATED
{
  null (0),
  initiated (1),
  alerting (2),
  connected (3),
  hold (4),
  queued (5),
  fail (6)}

END -- CSTA-connection-states

```

9.5 Status reporting

```

CSTA-status-reporting
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) status-reporting( 126) }

DEFINITIONS ::=
BEGIN

EXPORTS
MonitorObject, MonitorCrossRefID, MonitorFilter, MonitorType,
MonitorMediaClass, SnapshotCallData, SnapshotDeviceData;

IMPORTS
CallControlEvents, CallAssociatedEvents, MediaAttachmentEvents,
PhysicalDeviceFeatureEvents, LogicalDeviceFeatureEvents,
DeviceMaintenanceEvents, VoiceUnitEvents, VendorSpecEvents
    FROM CSTA-application-context-information-csta3
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) application-context-information( 200) }
ServicesPermitted FROM CSTA-call-control
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) call-control(130) }
CSTAObject FROM CSTA-switching-function-objects
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) switching-function-objects( 122) }
DeviceID, SubjectDeviceID FROM CSTA-device-identifiers
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) device-identifiers( 123) }
ConnectionID FROM CSTA-call-connection-identifiers
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) call-connection-identifiers( 124) }
LocalConnectionState FROM CSTA-connection-states
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) connection-states( 125) }
ConnectionInformation, MediaCallCharacteristics, MediaServiceType,
MediaServiceInstanceID, MediaStreamID FROM CSTA-media-services
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) media-services( 136) }
;

MonitorObject ::= CSTAObject;

MonitorCrossRefID ::= [APPLICATION 21] IMPLICIT OCTET STRING (SIZE(0..4))

MonitorFilter ::= SEQUENCE-- default is no filter (i.e. all events)
    {callControl          [0] IMPLICIT CallControlEvents          DEFAULT {},
     callAssociated       [6] IMPLICIT CallAssociatedEvents        DEFAULT {},
     mediaAttachment      [7] IMPLICIT MediaAttachmentEvents       DEFAULT {},
     physicalDeviceFeature [8] IMPLICIT PhysicalDeviceFeatureEvents DEFAULT {},
     logicalDeviceFeature [9] IMPLICIT LogicalDeviceFeatureEvents  DEFAULT {},
     maintenance          [3] IMPLICIT DeviceMaintenanceEvents     DEFAULT {},
     voiceUnit            [5] IMPLICIT VoiceUnitEvents             DEFAULT {},
     private              [4] IMPLICIT VendorSpecEvents            DEFAULT {}}

-- setting the relevant bit requests the filter for the appropriate events

MonitorType ::= ENUMERATED
    {
        call          (0),
        device        (1)
    }

MonitorMediaClass ::= BIT STRING
    {
        voice          (0),
        data            (1),
        image           (2),
        audio           (4),
        other           (3)}

```

SnapshotDeviceData ::= [APPLICATION 22] IMPLICIT SEQUENCE OF SnapshotDeviceResponseInfo

```
SnapshotDeviceResponseInfo ::= SEQUENCE
{
    connectionIdentifier      ConnectionID,
    localCallState            CallState,
    servicesPermitted         [0] IMPLICIT ServicesPermitted      OPTIONAL,
    mediaServiceInfoList      [1] IMPLICIT DeviceMediaInfoList    OPTIONAL,
    mediaCallCharacteristics  [2] IMPLICIT MediaCallCharacteristics OPTIONAL}
```

```
DeviceMediaInfoList ::= SEQUENCE OF SEQUENCE
{
    mediaStreamID      MediaStreamID      OPTIONAL,
    connectionInformation ConnectionInformation OPTIONAL}
```

SnapshotCallData ::= [APPLICATION 23] IMPLICIT SEQUENCE OF SnapshotCallResponseInfo

```
SnapshotCallResponseInfo ::= SEQUENCE
{
    deviceOnCall      SubjectDeviceID,
    callIdentifier     ConnectionID      OPTIONAL,
    localConnectionState LocalConnectionState OPTIONAL,
    servicesPermitted  [0] IMPLICIT ServicesPermitted OPTIONAL,
    mediaServiceInfoList [1] IMPLICIT CallMediaInfoList OPTIONAL
}
```

```
CallMediaInfoList ::= SEQUENCE OF SEQUENCE
{
    mediaServiceType      [0] IMPLICIT MediaServiceType,
    mediaServiceVersion   [1] IMPLICIT INTEGER          OPTIONAL,
    mediaServiceInstance  [2] IMPLICIT MediaServiceInstanceID OPTIONAL,
    mediaStreamID         [3] IMPLICIT MediaStreamID     OPTIONAL,
    connectionInformation [4] IMPLICIT ConnectionInformation OPTIONAL}
```

```
CallState ::= CHOICE
{
    compoundCallState [0] IMPLICIT CompoundCallState,
    simpleCallState   [1] IMPLICIT SimpleCallState,
    unknown           [2] IMPLICIT NULL}
```

CompoundCallState ::= SEQUENCE OF LocalConnectionState

```
SimpleCallState ::= ENUMERATED
{
    callNull          (0),      -- '00'H - null-null
    callPending       (1),      -- '01'H - null-initiate
    callOriginated    (3),      -- '03'H - null-connect
    callDelivered     (35),     -- '23'H - alerting-connect
    callDeliveredHeld (36),     -- '24'H - alerting-held
    callReceived      (50),     -- '32'H - connect-alerting
    callEstablished   (51),     -- '33'H - connect-connect
    callEstablishedHeld (52),   -- '34'H - connected-held
    callReceivedOnHold (66),    -- '42'H - held-alerting
    callEstablishedOnHold (67), -- '43'H - held-connect
    callQueued        (83),     -- '53'H - queued-connect
    callQueuedHeld    (84),     -- '54'H - queued-held
    callFailed        (99),     -- '63'H - failed-connect
    callFailedHeld    (100),    -- '64'H - failed-held
    callBlocked       (96)}     -- '60'H - failed-null
```

-- This represents the main call states in a simplified encoding. The semantics
 -- are identical to the sequence of connection states but they are represented by
 -- an item from an enumerated list.

END -- of CSTA-status-reporting

9.6 Device and feature types and other parameters

```

CSTA-device-feature-types
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) device-feature-types( 127) }

DEFINITIONS ::=
BEGIN

EXPORTS
AccountInfo, AgentID, AgentPassword, AgentState, AuthCode, CallOrigination, CorrelatorData,
DataPathType, DisplayAttributeList, ForwardList, IOCrossRefID, IORegisterReqID,
MessageID, ParticipationType, PendingAgentState, RetryValue, RouteRegisterReqID,
RouteingCrossRefID, SelectValue, SysStatRegisterID, SystemStatus, TerminatingConditions,
ForwardingType, ForwardDefault, AttributeInfo,
EncodingAlgorithm, ControlData, UserData, TelephonyTone;

IMPORTS
DeviceID FROM CSTA-device-identifiers
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) device-identifiers( 123) }
ConnectionID FROM CSTA-call-connection-identifiers
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) call-connection-identifiers( 124) };

AccountInfo ::= OCTET STRING (SIZE(0..32))

AgentID ::= OCTET STRING (SIZE(0..32))

AgentPassword ::= OCTET STRING (SIZE(0..32))

AgentState ::= ENUMERATED
{
    agentNotReady      (0),
    agentNull          (1),
    agentReady         (2),
    agentBusy          (3),
    agentWorkingAfterCall (4)}

AuthCode ::= OCTET STRING (SIZE(0..32))

CorrelatorData ::= OCTET STRING (SIZE(0..32))

IOCrossRefID ::= CHOICE
{
    switchProvided      [0] OCTET STRING (SIZE(0..4)),
    computerProvided    [1] OCTET STRING (SIZE(0..4))}

IORegisterReqID ::= OCTET STRING (SIZE(0..4))

MessageID ::= OCTET STRING

ParticipationType ::= ENUMERATED
{
    silent      (0),
    active      (1) }

RouteRegisterReqID ::= OCTET STRING (SIZE(0..4))

RouteingCrossRefID ::= [APPLICATION 24] IMPLICIT OCTET STRING (SIZE(0..4))

SelectValue ::= ENUMERATED
{
    normal      (0),
    leastCost   (1),
    emergency   (2),
    aCD         (3),
    userDefined (4) }

RetryValue ::= CHOICE-- used in RouteSelect Request service
{
    noListAvailable      [0] IMPLICIT BOOLEAN,
    noCountAvailable     [1] IMPLICIT BOOLEAN,
    retryCount           [2] IMPLICIT INTEGER }

```

```

SysStatRegisterID ::= OCTET STRING (SIZE(0..4))

SystemStatus ::= ENUMERATED
{
    disabled                (4),
    partiallyDisabled       (8),
    enabled                 (1),
    initializing            (0),
    messagesLost            (3),
    normal                  (2),
    overloadImminent        (5),
    overloadReached         (6),
    overloadRelieved        (7)      }

CallOrigination ::= BIT STRING
{
    internal    (0),
    external    (1)}

ForwardList ::= SEQUENCE OF SEQUENCE
{
    forwardingType      ForwardingType      OPTIONAL,
    forwardStatus       BOOLEAN,
    forwardDN           DeviceID             OPTIONAL,
    forwardDefault      ForwardDefault       OPTIONAL,
    ringCount           INTEGER (1..100)     OPTIONAL}

ForwardingType ::= ENUMERATED
{
    forwardImmediate      (0),
    forwardBusy           (1),
    forwardNoAns          (2),
    forwardDND            (9),
    forwardBusyInt        (3),
    forwardBusyExt        (4),
    forwardNoAnsInt       (5),
    forwardNoAnsExt       (6),
    forwardImmInt         (7),
    forwardImmExt         (8),
    forwardDNDInt         (10),
    forwardDNDExt         (11)}

ForwardDefault ::= ENUMERATED
{
    forwardingTypeAndForwardDN (0),
    forwardingType             (1),
    forwardDN                  (2)}

PendingAgentState ::= ENUMERATED
{
    agentNotReady      (0),
    agentNull          (1),
    agentReady         (2),
    agentWorkingAfterCall (3)}

DataPathType ::= ENUMERATED
{
    text    (0),
    voice   (1)}

DisplayAttributeList ::= SEQUENCE
{
    physicalBaseRowNumber      [0] IMPLICIT INTEGER OPTIONAL,
    physicalBaseColumnNumber   [1] IMPLICIT INTEGER OPTIONAL,
    offset                     [2] IMPLICIT INTEGER OPTIONAL}

TerminatingConditions ::= BIT STRING
{
    durationExceeded      (0),
    dTMFDigitDetected     (1),
    endOfMessageDetected  (2),
    speechDetected        (3)}

AttributeInfo ::= CHOICE
{
    encodingAlgorithm      [0] IMPLICIT EncodingAlgorithm,
    samplingRate           [1] IMPLICIT INTEGER,

```

```

duration          [2] IMPLICIT INTEGER,
filename          [3] IMPLICIT IA5String,
currentPosition   [4] IMPLICIT INTEGER,
currentSpeed      [5] IMPLICIT INTEGER,
currentVolume     [6] IMPLICIT INTEGER (0 .. 100),
currentGain       [7] IMPLICIT INTEGER (0 .. 100),
currentState      [8] IMPLICIT CurrentState}

EncodingAlgorithm ::= ENUMERATED
{
    aDPCM6K      (0),
    aDPCM8K      (1),
    muLawPCM6K   (2),
    aLawPCM6K    (3)}

CurrentState ::= ENUMERATED
{
    stop          (0),
    play          (1),
    record        (2),
    suspendPlay   (3),
    suspendRecord (4),
    review        (5)}

ControlData ::= SEQUENCE
{
    gender          ENUMERATED
                    {male          (0),
                     female        (1)},
    language        OCTET STRING}

UserData ::= [APPLICATION 29] OCTET STRING (SIZE(0..256))

TelephonyTone ::= ENUMERATED
{
    beep          ( 0),
    billing        ( 1),
    busy           ( 2),
    carrier        ( 3),
    confirmation   ( 4),
    dial           ( 5),
    faxCNG         ( 6),
    hold           ( 7),
    howler         ( 8),
    intrusion      ( 9),
    modemCNG       (10),
    park           (11),
    recordWarning  (12),
    reorder        (13),
    ringback       (14),
    silence        (15),
    sitVC          (16),
    sitIC          (17),
    sitRO          (18),
    sitNC          (19),
    switchSpec0    (20),
    switchSpec1    (21),
    switchSpec2    (22),
    switchSpec3    (23),
    switchSpec4    (24),
    switchSpec5    (25),
    switchSpec6    (26),
    switchSpec7    (27),
    switchSpec8    (28),
    switchSpec9    (29),
    switchSpec10   (30),
    switchSpec11   (31),
    switchSpec12   (32),
    switchSpec13   (33),
    switchSpec14   (34),
    switchSpec15   (35),

```

switchSpec16	(36),
switchSpec17	(37),
switchSpec18	(38),
switchSpec19	(39),
switchSpec20	(40),
switchSpec21	(41),
switchSpec22	(42),
switchSpec23	(43),
switchSpec24	(44),
switchSpec25	(45),
switchSpec26	(46),
switchSpec27	(47),
switchSpec28	(48),
switchSpec29	(49),
switchSpec30	(50),
switchSpec31	(51),
switchSpec32	(52),
switchSpec33	(53),
switchSpec34	(54),
switchSpec35	(55),
switchSpec36	(56),
switchSpec37	(57),
switchSpec38	(58),
switchSpec39	(59),
switchSpec40	(60),
switchSpec41	(61),
switchSpec42	(62),
switchSpec43	(63),
switchSpec44	(64),
switchSpec45	(65),
switchSpec46	(66),
switchSpec47	(67),
switchSpec48	(68),
switchSpec49	(69),
switchSpec50	(70),
switchSpec51	(71),
switchSpec52	(72),
switchSpec53	(73),
switchSpec54	(74),
switchSpec55	(75),
switchSpec56	(76),
switchSpec57	(77),
switchSpec58	(78),
switchSpec59	(79),
switchSpec60	(80),
switchSpec61	(81),
switchSpec62	(82),
switchSpec63	(83),
switchSpec64	(84),
switchSpec65	(85),
switchSpec66	(86),
switchSpec67	(87),
switchSpec68	(88),
switchSpec69	(89),
switchSpec70	(90),
switchSpec71	(91),
switchSpec72	(92),
switchSpec73	(93),
switchSpec74	(94),
switchSpec75	(95),
switchSpec76	(96),
switchSpec77	(97),
switchSpec78	(98),
switchSpec79	(99),
switchSpec80	(100),
switchSpec81	(101),
switchSpec82	(102),
switchSpec83	(103),

```
switchSpec84      (104),
switchSpec85      (105),
switchSpec86      (106),
switchSpec87      (107),
switchSpec88      (108),
switchSpec89      (109),
switchSpec90      (110),
switchSpec91      (111),
switchSpec92      (112),
switchSpec93      (113),
switchSpec94      (114),
switchSpec95      (115),
switchSpec96      (116),
switchSpec97      (117),
switchSpec98      (118),
switchSpec99      (119),
switchSpec100     (120)}
```

END -- of CSTA-device-feature-types

9.7 Security

```

CSTA-security
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) security( 128) }

DEFINITIONS ::=
BEGIN

EXPORTS
CSTASecurityData, TimeInfo;

CSTASecurityData ::= SEQUENCE
{
    messageSequenceNumber[0] IMPLICIT INTEGER    OPTIONAL,
    timestamp                TimeInfo            OPTIONAL,
    securityInfo              SecurityInfo        OPTIONAL}

SecurityInfo ::= CHOICE
{
    string    OCTET STRING,
    private   NULL}    -- The actual encoding is added here,
                      -- replacing NULL with another valid ASN.1 type.

TimeInfo ::= GeneralizedTime

END -- of CSTA-security

```

9.8 Common extensions

```

CSTA-extension-types
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) extension-types( 129) }
DEFINITIONS ::=
BEGIN

EXPORTS
CSTACommonArguments, CSTAPrivateData;

IMPORTS
CSTASecurityData FROM CSTA-security
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) security( 128) };

CSTACommonArguments ::= [APPLICATION 30] IMPLICIT SEQUENCE
    { security      [0] IMPLICIT  CSTASecurityData      OPTIONAL,
      privateData   [1] IMPLICIT  SEQUENCE OF CSTAPrivateData  OPTIONAL }

CSTAPrivateData ::= CHOICE
{
    string      OCTET STRING,
    private     NULL}      -- The actual encoding is added here,
                          -- replacing NULL with another valid ASN.1 type.

END -- of CSTA-extension-types

```

9.9 Call control

```

CSTA-call-control
{ iso( 1) identified-organization( 3) icd-ecma( 12)
  standard( 0) csta3( 285) call-control( 130) }

DEFINITIONS ::=
BEGIN

EXPORTS
AutoOriginate, CallCharacteristics, CallLinkageData, CallLinkageDataList, CallQualifyingData,
ConsultOptions, NetworkCapability,
ProgressIndicator, ServicesPermitted;

IMPORTS
TimeInfo FROM CSTA-security
{ iso( 1) identified-organization( 3) icd-ecma( 12) standard( 0)
  csta3( 285) security( 128) }
CallControlServices, CallAssociatedServices, MediaAttachmentServices, RoutingServices,
VoiceUnitServices FROM CSTA-application-context-information-csta3
{ iso( 1) identified-organization( 3) icd-ecma( 12)
  standard( 0) csta3( 285) application-context-information( 200) };

AutoOriginate ::= ENUMERATED
{
  prompt          (0),
  doNotPrompt     (1) }

CallCharacteristics ::= BIT STRING
{
  acdCall          (0),
  priorityCall     (1),
  maintenanceCall  (2),
  directAgent      (3),
  assistCall       (4),
  voiceUnitCall    (5)}

CallQualifyingData ::= OCTET STRING (SIZE(0..32))

ConsultOptions ::= ENUMERATED
{
  unrestricted     (0),
  consultOnly      (1),
  transferOnly     (2),
  conferenceOnly   (3) }

NetworkCapability ::= SEQUENCE
{
  networkType ENUMERATED
  {
    iSDNPublic      (0),
    nonISDNPublic   (1),
    iSDNPrivate     (2),
    nonISDNPrivate  (3),
    other           (4) },
  eventsProvided BIT STRING
  {
    bridged          (0),
    callCleared      (1),
    conferenced      (2),
    connectionCleared (3),
    delivered         (4),
    digitsDialed      (5),
    diverted          (6),
    established       (7),
    failed            (8),
    held              (9),
    networkCapabilitiesChange (10),
    networkReached    (11),
    offered           (12),
    originated        (13),
    queued            (14),
    retrieved         (15),
    serviceInitiated  (16),

```

```

        transferred                                (17)    } OPTIONAL}

ProgressIndicator ::= SEQUENCE
{
    progressLocation ENUMERATED
    {
        user (0),
        privateNetServingLocal (1),
        publicNetServingLocal (2),
        transitNetwork (3),
        publicNetServingRemote (4),
        privateNetServingRemote (5),
        localInterface (6),
        internationalNetwork (7),
        networkBeyondInterwk (8),
        other (9)
    },
    progressDescription ENUMERATED
    {
        iSDNProgressDesc (0),
        qSIGProgressDesc (1),
        other (2)
    }
}

ServicesPermitted ::= SEQUENCE
{
    callControlServices      CallControlServices,
    callAssociatedServices    CallAssociatedServices,
    mediaAttachmentServices  MediaAttachmentServices,
    routingServices          RoutingServices,
    voiceUnitServices        VoiceUnitServices
}

CallLinkageDataList ::= SEQUENCE
{
    newCallLinkageData      CallLinkageData,
    oldCallLinkageData      CallLinkageData
}

CallLinkageData ::= SEQUENCE
{
    globalCallData          GlobalCallData,
    threadData              ThreadData
    OPTIONAL
}

GlobalCallData ::= SEQUENCE
{
    globalCallSwitchingSubDomainName  IA5String (SIZE(1..64))    OPTIONAL,
    globalCallLinkageID               GlobalCallLinkageID,
    callLinkageIDTimestamp            TimeInfo                  OPTIONAL
}

GlobalCallLinkageID ::= CHOICE
{
    subDomainCallLinkageID            [0] IMPLICIT OCTET STRING (SIZE(1..8)),
    globallyUniqueCallLinkageID       [1] IMPLICIT OCTET STRING (SIZE(1..16))
}

ThreadData ::= SEQUENCE
{
    threadSwitchingSubDomainName      IA5String (SIZE(1..64))    OPTIONAL,
    threadLinkageID                  ThreadLinkageID,
    threadIDTimeStamp                 TimeInfo                  OPTIONAL
}

ThreadLinkageID ::= CHOICE
{
    subDomainThreadID                [0] IMPLICIT OCTET STRING (SIZE(1..8)),
    globallyUniqueThreadID           [1] IMPLICIT OCTET STRING (SIZE(1..16))
}

END -- of CSTA-call-control

```

9.10 Capability Exchange

```

CSTA-capability-exchange
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) capability-exchange( 131) }

DEFINITIONS ::=
BEGIN

EXPORTS
CapExchangeServList, SystemStatusServList, MonitoringServList, SnapshotServList,
CallControlServList, CallControlEvtsList, CallAssociatedServList,
CallAssociatedEvtsList, MediaServList, MediaEvtsList, RouteingServList, PhysDevEvtsList,
PhysDevServList, LogicalEvtsList, LogicalServList, DeviceMaintEvtsList, IOServicesServList,
DataCollectionServList, VoiceUnitServList, VoiceUnitEvtsList, CDRServList,
VendorSpecificServList, VendorSpecificEvtsList, DeviceIDFormat, SwDomainFeatures,
SwAppearanceAddressability, SwAppearanceTypes, IgnoreUnsupportedParameters, PauseTime,
TimeStampMode, MiscMonitorCaps, MaxLengthParameters, FilterThreshold, ServiceCrossRefID,
DeviceCategory, GroupDeviceAttributes, NamedDeviceTypes, ACDModels, AgentLogOnModels,
AppearanceType, TransAndConfSetup, MediaServiceCapsList, LogDevServList, LogDevEvtsList;

IMPORTS
MediaServiceType, MediaServiceInstanceID, ConnectionMode, ConnectionModeBMap
FROM CSTA-media-services
    { iso( 1) identified-organization( 3) icd-ecma( 12)
      standard( 0) csta3( 285) media-services( 136) };

ServiceCrossRefID ::= OCTET STRING (SIZE(0..4))

-- Capability Bitmaps

LogDevServList ::= SEQUENCE
{
    callControlServList      [0] IMPLICIT CallControlServList      OPTIONAL,
    callAssociatedServList   [1] IMPLICIT CallAssociatedServList   OPTIONAL,
    logicalServList          [2] IMPLICIT LogicalServList          OPTIONAL,
    mediaServList            [3] IMPLICIT MediaServList            OPTIONAL,
    iOServicesServList       [4] IMPLICIT IOServicesServList       OPTIONAL,
    dataCollectionServList   [5] IMPLICIT DataCollectionServList   OPTIONAL,
    voiceUnitServList        [6] IMPLICIT VoiceUnitServList        OPTIONAL}

LogDevEvtsList ::= SEQUENCE
{
    callControlEvtsList      [0] IMPLICIT CallControlEvtsList      OPTIONAL,
    callAssociatedEvtsList   [1] IMPLICIT CallAssociatedEvtsList   OPTIONAL,
    logicalEvtsList          [2] IMPLICIT LogicalEvtsList          OPTIONAL,
    mediaEvtsList            [3] IMPLICIT MediaEvtsList            OPTIONAL,
    voiceUnitEvtsList        [4] IMPLICIT VoiceUnitEvtsList        OPTIONAL}

CapExchangeServList ::= SEQUENCE
{
    getLogicalDeviceInformation [0] IMPLICIT GetLogicalDeviceInformation OPTIONAL,
    getPhysicalDeviceInformation [1] IMPLICIT GetPhysicalDeviceInformation OPTIONAL,
    getSwitchingFunctionCaps    NULL,
    getSwitchingFunctionDevices [2] IMPLICIT GetSwitchingFunctionDevices OPTIONAL,
    switchingFunctionDevices     [3] IMPLICIT SwitchingFunctionDevices OPTIONAL}

GetLogicalDeviceInformation ::= BIT STRING
{
    privateDataInReq          ( 0),
    namedDeviceTypeInAck      ( 1),
    shortFormDeviceIDInAck    ( 2),
    miscMonitorCapsInAck      ( 3),
    maxCallBacksInAck         ( 4),
    maxAutoAnswerRingsInAck   ( 5),
    maxActiveCallsInAck       ( 6),
    maxHeldCallsInAck         ( 7),
    maxFwdSettingsInAck       ( 8),
    maxDevicesInConfInAck     ( 9),
    transAndConfSetupParameter (10),
    transAndConfSetupConsultationInAck (11),

```

```

transAndConfSetupHoldMakeInAck          (12),
transAndConfSetupAlternateInAck          (13),
transAndConfSetupTwoCallsHoldInAck       (14),
transAndConfSetupTwoCallsConnectedInAck  (15),
mediaClassSupportInAck                   (16),
connectionRateListInAck                  (17),
delayToleranceListInAck                  (18),
numberOfChannelsInAck                    (19),
maxChannelBindInAck                      (20),
privateDataInAck                         (21)}

GetPhysicalDeviceInformation ::= BIT STRING
{
    privateDataInReq          ( 0),
    namedDeviceTypesInAck     ( 1),
    otherLogicalDeviceListInAck ( 2),
    deviceModelNameInAck      ( 3),
    maxDisplaysInAck          ( 4),
    maxButtonsInAck           ( 5),
    maxLampsInAck             ( 6),
    maxRingPatternsInAck      ( 7),
    privateDataInAck          ( 8)}

GetSwitchingFunctionDevices ::= BIT STRING
{
    requestedDeviceID          ( 0),
    requestedDeviceCategoryACD  ( 1),
    requestedDeviceCategoryACDGroup ( 2),
    requestedDeviceCategoryHuntGroup ( 3),
    requestedDeviceCategoryPickGroup ( 4),
    requestedDeviceCategoryOtherGroup ( 5),
    requestedDeviceCategoryNetwInterface ( 6),
    requestedDeviceCategoryPark ( 7),
    requestedDeviceCategoryRouteingDevice ( 8),
    requestedDeviceCategoryStation ( 9),
    requestedDeviceCategoryVoiceUnit (10),
    requestedDeviceCategoryOther (11),
    privateData                (12),
    privateDataInAck           (13)}

SwitchingFunctionDevices ::= BIT STRING
{
    segmentID          ( 0),
    deviceListDeviceCategory ( 1),
    deviceListNamedDeviceTypes ( 2),
    deviceListDeviceAttributes ( 3),
    deviceListDeviceModelName ( 4),
    privateData          ( 5)}

SystemStatusServList ::= SEQUENCE
{
    changeSystemStatusFilter [ 0] IMPLICIT ChangeSystemStatusFilter OPTIONAL,
    systemRegister           [ 1] IMPLICIT SystemRegister           OPTIONAL,
    systemStatusRegisterAbort [ 2] IMPLICIT SystemStatusRegisterAbort OPTIONAL,
    systemStatusRegisterCancel [ 3] IMPLICIT SystemStatusRegisterCancel OPTIONAL,
    requestSystemStatus       [ 4] IMPLICIT RequestSystemStatus      OPTIONAL,
    systemStatus              [ 5] IMPLICIT SystemStatus             OPTIONAL,
    swFunctionCapsChanged     [ 6] IMPLICIT SwFunctionCapsChanged    OPTIONAL,
    swFunctionDevicesChanged  [ 7] IMPLICIT SwFunctionDevicesChanged  OPTIONAL}

ChangeSystemStatusFilter ::= BIT STRING
{
    requestedStatusFilterInitializing ( 0), -- optional parameters
    requestedStatusFilterEnabled      ( 1), -- optional parameters
    requestedStatusFilterNormal       ( 2), -- optional parameters
    requestedStatusFilterMessageLost  ( 3), -- optional parameters
    requestedStatusFilterDisabled     ( 4), -- optional parameters
    requestedStatusFilterPartiallyDisabled ( 5), -- optional parameters
    requestedStatusFilterOverloadImminent ( 6), -- optional parameters
    requestedStatusFilterOverloadReached ( 7), -- optional parameters
    requestedStatusFilterOverloadRelieved ( 8), -- optional parameters
    privateData                      ( 9), -- optional parameters

```

```

        privateDataInAck                                (10)} -- optional parameters

SystemRegister ::= BIT STRING
{
    requestTypesSystemStatus                            ( 0), -- optional parameters
    requestTypesRequestSystemStatus                     ( 1), -- optional parameters
    requestTypesSwitchingFunctionCapsChanged           ( 2), -- optional parameters
    requestTypesSwitchingFunctionDevicesChanged         ( 3), -- optional parameters
    statusFilterInitializing                             ( 4), -- optional parameters
    statusFilterEnabled                                 ( 5), -- optional parameters
    statusFilterNormal                                   ( 6), -- optional parameters
    statusFilterMessageLost                             ( 7), -- optional parameters
    statusFilterDisabled                                ( 8), -- optional parameters
    statusFilterPartiallyDisabled                       ( 9), -- optional parameters
    statusFilterOverloadImminent                        (10), -- optional parameters
    statusFilterOverloadReached                         (11), -- optional parameters
    statusFilterOverloadRelieved                        (12), -- optional parameters
    privateDataOctetString                             (13), -- optional parameters
    privateData                                          (14), -- optional parameters
    privateDataOctetStringInAck                         (15), -- optional parameters
    privateDataInAck                                    (16)} -- optional parameters

SystemStatusRegisterCancel ::= BIT STRING
{
    privateData          (0), -- optional parameters
    privateDataInAck     (1)} -- optional parameters

SystemStatusRegisterAbort ::= BIT STRING
{
    privateData          (0), -- optional parameters
    privateDataInAck     (1)} -- optional parameters

RequestSystemStatus ::= BIT STRING
{
    privateDataInReq      ( 0), -- optional parameters
    systemStatusInitializing ( 1), -- optional parameters
    systemStatusEnabled   ( 2), -- optional parameters
    systemStatusNormal    ( 3), -- optional parameters
    systemStatusMessageLost ( 4), -- optional parameters
    systemStatusDisabled  ( 5), -- optional parameters
    systemStatusPartiallyDisabled ( 6), -- optional parameters
    systemStatusOverloadImminent ( 7), -- optional parameters
    systemStatusOverloadReached ( 8), -- optional parameters
    systemStatusOverloadRelieved ( 9), -- optional parameters
    privateDataInAck      (10), -- optional parameters
    switchingFunctionSupportsSending (11), -- misc characteristics
    switchingFunctionSupportsReceiving (12)} -- misc characteristics

SystemStatus ::= BIT STRING
{
    systemStatusInitializing ( 0), -- optional parameters
    systemStatusEnabled      ( 1), -- optional parameters
    systemStatusNormal       ( 2), -- optional parameters
    systemStatusMessageLost  ( 3), -- optional parameters
    systemStatusDisabled     ( 4), -- optional parameters
    systemStatusPartiallyDisabled ( 5), -- optional parameters
    systemStatusOverloadImminent ( 6), -- optional parameters
    systemStatusOverloadReached ( 7), -- optional parameters
    systemStatusOverloadRelieved ( 8), -- optional parameters
    privateData              ( 9), -- optional parameters
    privateDataInAck         (10), -- optional parameters
    switchingFunctionSupportsSending (11), -- misc characteristics
    switchingFunctionSupportsReceiving (12)} -- misc characteristics

SwFunctionCapsChanged ::= BIT STRING
{
    privateData          (0), -- optional parameters
    privateDataInAck     (1)} -- optional parameters

SwFunctionDevicesChanged ::= BIT STRING
{
    privateData          (0), -- optional parameters
    privateDataInAck     (1)} -- optional parameters

MonitoringServList ::= SEQUENCE

```

```

{      changeMonitorFilter      [ 0] IMPLICIT ChangeMonitorFilter      OPTIONAL,
      monitorStart              [ 1] IMPLICIT MonitorStart        OPTIONAL,
      monitorStop               [ 2] IMPLICIT MonitorStop         OPTIONAL}

ChangeMonitorFilter ::= BIT STRING
{      privateData              (0),      -- optional parameters
      privateDataInAck          (1)}      -- optional parameters

MonitorStart ::= BIT STRING
{      monitorObjectDevice      ( 0),      -- optional parameters
      monitorObjectCall         ( 1),      -- optional parameters
      requestedMonitorFilter     ( 2),      -- optional parameters
      monitorTypeCall           ( 3),      -- optional parameters
      monitorTypeDevice         ( 4),      -- optional parameters
      monitorMediaClassParameter ( 5),      -- optional parameters
      monitorMediaClassAudio     ( 6),      -- optional parameters
      monitorMediaClassData      ( 7),      -- optional parameters
      monitorMediaClassImage     ( 8),      -- optional parameters
      monitorMediaClassVoice     ( 9),      -- optional parameters
      monitorExistingCallsInAck  (10),      -- optional parameters
      privateDataInAck           (11),      -- optional parameters
      callIDOnly                (12),      -- miscellaneous characteristics
      swDomainDefaultForMonitorTypeIsDevice (13)} -- miscellaneous characteristics

MonitorStop ::= BIT STRING
{      privateData              ( 0),      -- optional parameters
      privateDataInAck          ( 1),      -- optional parameters
      switchingFunctionSupportsSending ( 2),      -- misc characteristics
      switchingFunctionSupportsReceiving ( 3)} -- misc characteristics

SnapshotServList ::= SEQUENCE
{      snapshotCall             [0] IMPLICIT SnapshotCall,
      snapshotDevice            [1] IMPLICIT SnapshotDevice,
      snapshotCallData          [2] IMPLICIT SnapshotCallData,
      snapshotDeviceData        [3] IMPLICIT SnapshotDeviceData}

SnapshotCall ::= BIT STRING
{      privateData              ( 0),      -- optional parameters
      locaConnectionStateInAck  ( 1),      -- optional parameters
      mediaServiceInfoListInAck ( 2),      -- optional parameters
      mediaServiceVersionInAck  ( 3),      -- optional parameters
      mediaServiceInstanceInAck ( 4),      -- optional parameters
      mediaStreamID             ( 5),      -- optional parameters
      connectionInformation     ( 6),      -- optional parameters
      mediaCallCharacteristicsInAck ( 7),      -- optional parameters
      callCharacteristicsInAck  ( 8),      -- optional parameters
      callingDeviceInAck        ( 9),      -- optional parameters
      calledDeviceInAck         (10),      -- optional parameters
      privateDataInAck          (11),      -- optional parameters
      callIDOnly                (12),      -- miscellaneous characteristics
      reportsWithSnapsCallData  (13)} -- miscellaneous characteristics

SnapshotDevice ::= BIT STRING
{      privateData              ( 0),      -- optional parameters
      localCallStateCompoundCallState ( 1),      -- optional parameters
      localCallStateSimpleCallState ( 2),      -- optional parameters
      localCallStateUnknown     ( 3),      -- optional parameters
      mediaServiceInfoListInAck ( 4),      -- optional parameters
      mediaServiceVersionInAck  ( 5),      -- optional parameters
      mediaServiceInstanceInAck ( 6),      -- optional parameters
      mediaStreamID             ( 7),      -- optional parameters
      connectionInformation     ( 8),      -- optional parameters
      mediaCallCharacteristicsInAck ( 9),      -- optional parameters
      privateDataInAck          (10),      -- optional parameters
      reportsWithSnapsDeviceData (11)} -- miscellaneous characteristics

SnapshotCallData ::= BIT STRING
{      segmentID                ( 0),      -- optional parameters

```

```

    localConnectionState      ( 1),  -- optional parameters
    mediaServiceInformationList ( 2),  -- optional parameters
    mediaServiceVersion       ( 3),  -- optional parameters
    mediaServiceInstance      ( 4),  -- optional parameters
    mediaStreamID             ( 5),  -- optional parameters
    connectionInformation      ( 6),  -- optional parameters
    privateData                ( 7)}  -- optional parameters

SnapshotDeviceData ::= BIT STRING
{
    segmentID                  ( 0),  -- optional parameters
    localCallStateCompoundCallState ( 1),  -- optional parameters
    localCallStateSimpleCallState ( 2),  -- optional parameters
    localCallStateUnknown      ( 3),  -- optional parameters
    mediaServiceInformationList ( 4),  -- optional parameters
    mediaServiceVersion        ( 5),  -- optional parameters
    mediaServiceInstance       ( 6),  -- optional parameters
    mediaStreamID              ( 7),  -- optional parameters
    connectionInformation      ( 8),  -- optional parameters
    mediaCallCharacteristics    ( 9),  -- optional parameters
    privateData                (10)}  -- optional parameters

CallControlServList ::= SEQUENCE
{
    acceptCall      [0] IMPLICIT AcceptCall      OPTIONAL,
    alternateCall   [1] IMPLICIT AlternateCall    OPTIONAL,
    answerCall      [2] IMPLICIT AnswerCall       OPTIONAL,
    callBack        [3] IMPLICIT CallBack         OPTIONAL,
    callBackMessage [4] IMPLICIT CallBackMessage  OPTIONAL,
    campOnCall      [5] IMPLICIT CampOnCall       OPTIONAL,
    clearCall       [6] IMPLICIT ClearCall        OPTIONAL,
    clearConnection [7] IMPLICIT ClearConnection  OPTIONAL,
    conferenceCall  [8] IMPLICIT ConferenceCall   OPTIONAL,
    consultationCall [9] IMPLICIT ConsultationCall OPTIONAL,
    deflectCall     [10] IMPLICIT DeflectCall     OPTIONAL,
    dialDigits      [11] IMPLICIT DialDigits     OPTIONAL,
    directedPickupCall [12] IMPLICIT DirectedPickupCall OPTIONAL,
    groupPickupCall [13] IMPLICIT GroupPickupCall  OPTIONAL,
    holdCall        [14] IMPLICIT HoldCall        OPTIONAL,
    intrudeCall     [15] IMPLICIT IntrudeCall     OPTIONAL,
    joinCall        [16] IMPLICIT JoinCall        OPTIONAL,
    makeCall        [17] IMPLICIT MakeCall        OPTIONAL,
    makePredictiveCall [18] IMPLICIT MakePredictiveCall OPTIONAL,
    parkCall        [19] IMPLICIT ParkCall        OPTIONAL,
    reconnectCall   [20] IMPLICIT ReconnectCall   OPTIONAL,
    retrieveCall     [21] IMPLICIT RetrieveCall    OPTIONAL,
    singleStepConference [22] IMPLICIT SingleStepConference OPTIONAL,
    singleStepTransfer [23] IMPLICIT SingleStepTransfer OPTIONAL,
    transferCall    [24] IMPLICIT TransferCall    OPTIONAL}

AcceptCall ::= BIT STRING
{
    correlatorData      ( 0),  -- optional parameters
    userData            ( 1),  -- optional parameters
    privateData         ( 2),  -- optional parameters
    privateDataInAck    ( 3),  -- optional parameters
    deviceIDOnly        ( 4),  -- misc characteristics
    ackModelMultiStep   ( 5)}  -- misc characteristics

AlternateCall ::= BIT STRING
{
    alerting          ( 0),  -- initial states heldCall
    hold              ( 1),  -- initial states heldCall
    queued            ( 2),  -- initial states heldCall
    connectionReservation ( 3),  -- optional parameters
    consultOptionConsultOnly ( 4),  -- optional parameters
    consultOptionTransferOnly ( 5),  -- optional parameters
    consultOptionConferenceOnly ( 6),  -- optional parameters
    consultOptionUnrestricted ( 7),  -- optional parameters
    privateData       ( 8),  -- optional parameters
    privateDataInAck  ( 9),  -- optional parameters
    deviceIDOnly      (10),  -- misc characteristics

```

```

        ackModelMultiStep          (11), -- misc characteristics
        supportsOfferedModeOfAlerting (12)} -- misc characteristics

AnswerCall ::= BIT STRING
{
    alerting          ( 0), -- initial states
    initiated         ( 1), -- initial states
    queued            ( 2), -- initial states
    correlatorData    ( 3), -- optional parameters
    userData          ( 4), -- optional parameters
    privateData       ( 5), -- optional parameters
    privateDataInAck  ( 6), -- optional parameters
    deviceIDOnly      ( 7), -- misc characteristics
    supportsOfferedModeOfAlerting ( 8), -- misc characteristics
    ackModelMultiStep ( 9)} -- misc characteristics

CallBack ::= BIT STRING
{
    alerting          ( 0), -- initial states
    null              ( 1), -- initial states
    failed            ( 2), -- initial states
    queued            ( 3), -- initial states
    callCharacteristics ( 4), -- optional parameters
    privateData       ( 5), -- optional parameters
    targetDeviceInAck ( 6), -- optional parameters
    privateDataInAck  ( 7), -- optional parameters
    deviceIDOnly      ( 8), -- misc characteristics
    moreCallBacksNegAck ( 9), -- misc characteristics
    ackModelMultiStep (10)} -- misc characteristics

CallBackMessage ::= BIT STRING
{
    alerting          ( 0), -- initial states
    null              ( 1), -- initial states
    failed            ( 2), -- initial states
    queued            ( 3), -- initial states
    privateData       ( 4), -- optional parameters
    targetDeviceInAck ( 5), -- optional parameters
    privateDataInAck  ( 6), -- optional parameters
    deviceIDOnly      ( 7), -- misc characteristics
    moreCallBacksNegAck ( 8), -- misc characteristics
    ackModelMultiStep ( 9)} -- misc characteristics

CampOnCall ::= BIT STRING
{
    privateData          ( 0), -- optional parameters
    privateDataInAck     ( 1), -- optional parameters
    deviceIDOnly         ( 2), -- misc characteristics
    ackModelMultiStep    ( 3)} -- misc characteristics

ClearCall ::= BIT STRING
{
    alerting          ( 0), -- initial states
    connected         ( 1), -- initial states
    failed            ( 2), -- initial states
    queued            ( 3), -- initial states
    initiated         ( 4), -- initial states
    hold              ( 5), -- initial states
    userData          ( 6), -- optional parameters
    privateData       ( 7), -- optional parameters
    privateDataInAck  ( 8), -- optional parameters
    deviceIDOnly      ( 9), -- misc characteristics
    callIDOnly        (10), -- misc characteristics
    ackModelMultiStep (11)} -- misc characteristics

ClearConnection ::= BIT STRING
{
    alerting          ( 0), -- initial states
    connected         ( 1), -- initial states
    fail              ( 2), -- initial states
    queued            ( 3), -- initial states
    initiated         ( 4), -- initial states
    hold              ( 5), -- initial states
    correlatorData    ( 6), -- optional parameters

```

```

        userData                ( 7),    -- optional parameters
        privateData             ( 8),    -- optional parameters
        privateDataInAck        ( 9),    -- optional parameters
        deviceIDOnly            (10),    -- misc characteristics
        ackModelMultiStep       (11)}    -- misc characteristics

ConferenceCall ::= BIT STRING
{
    activeCallConnected         ( 0),    -- initial states
    activeCallHold              ( 1),    -- initial states
    heldCallConnected           ( 2),    -- initial states
    heldCallHold                ( 3),    -- initial states
    privateData                 ( 4),    -- optional parameters
    connectionParameterInAck    ( 5),    -- optional parameters
    endpointDeviceID            ( 6),    -- optional parameters
    resultingConnectionInfo     ( 7),    -- optional parameters
    conferenceCallOnfoInAck     ( 8),    -- optional parameters
    privateDataInAck            ( 9),    -- optional parameters
    protectedAgainstClearing    (10),    -- misc characteristics
    deviceIDOnly                (11),    -- misc characteristics
    ackModelMultiStep           (12)}    -- misc characteristics

ConsultationCall ::= BIT STRING
{
    connectionReservation       ( 0),    -- optional parameters
    accountCode                 ( 1),    -- optional parameters
    authCode                    ( 2),    -- optional parameters
    correlatorData              ( 3),    -- optional parameters
    userData                    ( 4),    -- optional parameters
    callCharacteristics          ( 5),    -- optional parameters
    callCharacteristicsACDCall   ( 6),    -- optional parameters
    callCharacteristicsPriorityCall ( 7),    -- optional parameters
    callCharacteristicsMaintenanceCall ( 8),    -- optional parameters
    callCharacteristicsDirectAgent ( 9),    -- optional parameters
    callCharacteristicsAssistCall (10),    -- optional parameters
    callCharacteristicsVoiceUnitCall (11),    -- optional parameters
    mediaCallCharacteristics     (12),    -- optional parameters
    callingConnectionInfo        (13),    -- optional parameters
    flowDirectionTransmit        (14),    -- optional parameters
    flowDirectionReceive         (15),    -- optional parameters
    flowDirectionTransmitAndReceive (16),    -- optional parameters
    numberOfChannels             (17),    -- optional parameters
    consultOptionConsultOnly      (18),    -- optional parameters
    consultOptionTransferOnly     (19),    -- optional parameters
    consultOptionConferenceOnly   (20),    -- optional parameters
    consultOptionUnrestricted     (21),    -- optional parameters
    privateData                  (22),    -- optional parameters
    initiatedCallInfoInAck        (23),    -- optional parameters
    privateDataInAck             (24),    -- optional parameters
    deviceIDOnly                 (25),    -- misc characteristics
    multiStage                   (26),    -- misc characteristics
    supportsAdjustmentOfMediaCharacteristics (27),    -- misc characteristics
    ackModelMultiStep            (28)}    -- misc characteristics

DeflectCall ::= BIT STRING
{
    alerting                    ( 0),    -- initial states
    connected                   ( 1),    -- initial states
    failed                      ( 2),    -- initial states
    hold                        ( 3),    -- initial states
    queued                     ( 4),    -- initial states
    correlatorData              ( 5),    -- optional parameters
    userData                    ( 6),    -- optional parameters
    privateData                 ( 7),    -- optional parameters
    privateDataInAck            ( 8),    -- optional parameters
    deviceIDOnly                ( 9),    -- misc characteristics
    ackModelMultiStep           (10)}    -- misc characteristics

DialDigits ::= BIT STRING
{
    correlatorData              ( 0),    -- optional parameters
    privateData                 ( 1),    -- optional parameters

```

```

privateDataInAck          ( 2), -- optional parameters
deviceIDOnly              ( 3), -- misc characteristics
ackModelMultiStep         ( 4)} -- misc characteristics

DirectedPickupCall ::= BIT STRING
{
    alerting                ( 0), -- initial states
    hold                    ( 1), -- initial states
    queued                  ( 2), -- initial states
    connected               ( 3), -- initial states
    correlatorData          ( 4), -- optional parameters
    userData                ( 5), -- optional parameters
    privateData             ( 6), -- optional parameters
    pickedCallInAck         ( 7), -- optional parameters
    pickedCallInfoInAck     ( 8), -- optional parameters
    privateDataInAck        ( 9), -- optional parameters
    deviceIDOnly            (10), -- misc characteristics
    supportsOfferedModeOfAlerting (11), -- misc characteristics
    supportsPrompting       (12), -- misc characteristics
    promptingMode           (13), -- misc characteristics
    ackModelMultiStep       (14)} -- misc characteristics

GroupPickupCall ::= BIT STRING
{
    alerting                ( 0), -- initial states
    connected               ( 1), -- initial states
    hold                    ( 2), -- initial states
    queued                  ( 3), -- initial states
    pickGroup               ( 4), -- optional parameters
    correlatorData          ( 5), -- optional parameters
    userData                ( 6), -- optional parameters
    privateData             ( 7), -- optional parameters
    pickedCallInAck         ( 8), -- optional parameters
    pickedCallInfoInAck     ( 9), -- optional parameters
    privateDataInAck        (10), -- optional parameters
    supportsPrompting       (11), -- misc characteristics
    promptingMode           (12), -- misc characteristics
    supportsOfferedModeOfAlerting (13), -- misc characteristics
    ackModelMultiStep       (14)} -- misc characteristics

HoldCall ::= BIT STRING
{
    connectionReservation   ( 0), -- optional parameters
    privateData             ( 1), -- optional parameters
    privateDataInAck        ( 2), -- optional parameters
    deviceIDOnly            ( 3), -- misc characteristics
    ackModelMultiStep       ( 4)} -- misc characteristics

IntrudeCall ::= BIT STRING
{
    participationTypeSilent ( 0), -- optional parameters
    participationTypeActive ( 1), -- optional parameters
    userData                ( 2), -- optional parameters
    privateData             ( 3), -- optional parameters
    conferencedCallInfoInAck ( 4), -- optional parameters
    privateDataInAck        ( 5), -- optional parameters
    deviceIDOnly            ( 6), -- misc characteristics
    supportsConference      ( 7), -- misc characteristics
    supportsAlternate       ( 8), -- misc characteristics
    ackModelMultiStep       ( 9)} -- misc characteristics

JoinCall ::= BIT STRING
{
    autoOriginatePrompt     ( 0), -- optional parameters
    autoOriginateDoNotPrompt ( 1), -- optional parameters
    participationTypeSilent ( 2), -- optional parameters
    participationTypeActive ( 3), -- optional parameters
    accountCode             ( 4), -- optional parameters
    authCode                ( 5), -- optional parameters
    correlatorData          ( 6), -- optional parameters
    userData                ( 7), -- optional parameters
    conferencedCallInAck    ( 8), -- optional parameters
    conferencedCallInfoInAck ( 9), -- optional parameters

```

```

privateDataInAck          (10),  -- optional parameters
deviceIDOnly              (11),  -- misc characteristics
supportsPrompting         (12),  -- misc characteristics
promptingMode             (13),  -- misc characteristics
ackModelMultiStep         (14)}  -- misc characteristics

MakeCall ::= BIT STRING
{
    initiated              ( 0),  -- initials states
    null                  ( 1),  -- initials states
    accountCode            ( 2),  -- optional parameters
    authCode              ( 3),  -- optional parameters
    autoOriginatePrompt    ( 4),  -- optional parameters
    autoOriginateDoNotPrompt ( 5),  -- optional parameters
    correlatorData        ( 6),  -- optional parameters
    userData              ( 7),  -- optional parameters
    callCharacteristics    ( 8),  -- optional parameters
    callCharacteristicsACDCall ( 9),  -- optional parameters
    callCharacteristicsPriorityCall (10),  -- optional parameters
    callCharacteristicsMaintenanceCall (11),  -- optional parameters
    callCharacteristicsDirectAgent (12),  -- optional parameters
    callCharacteristicsAssistCall (13),  -- optional parameters
    callCharacteristicsVoiceUnitCall (14),  -- optional parameters
    mediaCallCharacteristics (15),  -- optional parameters
    callingConnectionInfo  (16),  -- optional parameters
    privateData            (17),  -- optional parameters
    initiatedCallInfoInAck (18),  -- optional parameters
    privateDataInAck       (19),  -- optional parameters
    multiStage             (20),  -- misc characteristics
    supportsPrompting       (21),  -- misc characteristics
    promptingMode          (22),  -- misc characteristics
    offHook                (23),  -- misc characteristics
    mediaCharacteristicsAdjustable (24),  -- misc characteristics
    ackModelMultiStep      (25)}  -- misc characteristics

MakePredictiveCall ::= BIT STRING
{
    signallingDetection          ( 0),  -- optional parameters
    signallingConditionCallDelivered ( 1),  -- optional parameters
    signallingConditionCallEstablished ( 2),  -- optional parameters
    signallingConditionActionDestinationDetection ( 3),  -- optional parameters
    signallingConditionActionRemainConnected ( 4),  -- optional parameters
    destinationDetection        ( 5),  -- optional parameters
    destinationConditionHumanVoice ( 6),  -- optional parameters
    destinationConditionAnsweringMachine ( 7),  -- optional parameters
    destinationConditionFax      ( 8),  -- optional parameters
    destinationActionClearConnection ( 9),  -- optional parameters
    destinationActionRemainConnected (10),  -- optional parameters
    defaultActionClearConnection (11),  -- optional parameters
    defaultActionRemainConnected (12),  -- optional parameters
    accountCode                 (13),  -- optional parameters
    authCode                    (14),  -- optional parameters
    autoOriginatePrompt         (15),  -- optional parameters
    autoOriginateDoNotPrompt    (16),  -- optional parameters
    alertTime                   (17),  -- optional parameters
    correlatorData              (18),  -- optional parameters
    callCharacteristics          (19),  -- optional parameters
    callCharacteristicsACDCall   (20),  -- optional parameters
    callCharacteristicsPriorityCall (21),  -- optional parameters
    callCharacteristicsMaintenanceCall (22),  -- optional parameters
    callCharacteristicsDirectAgent (23),  -- optional parameters
    callCharacteristicsAssistCall (24),  -- optional parameters
    callCharacteristicsVoiceUnitCall (25),  -- optional parameters
    userData                    (26),  -- optional parameters
    privateData                 (27),  -- optional parameters
    initiatedCallInfoInAck      (28),  -- optional parameters
    privateDataInAck            (29),  -- optional parameters
    deviceIDOnly                (30),  -- misc characteristics
    supportsPrompting           (31),  -- misc characteristics
    promptingMode               (32),  -- misc characteristics

```

```

        reservesCallingDevice                (33),  -- misc characteristics
        ackModelMultiStep                    (34)}  -- misc characteristics

ParkCall ::= BIT STRING
{
    hold                ( 0),  -- initial states
    connected            ( 1),  -- initial states
    correlatorData       ( 2),  -- optional parameters
    privateData          ( 3),  -- optional parameters
    parkedToInAck        ( 4),  -- optional parameters
    privateDataInAck      ( 5),  -- optional parameters
    deviceIDOnly         ( 6),  -- misc characteristics
    ackModelMultiStep    ( 7)}  -- misc characteristics

ReconnectCall ::= BIT STRING
{
    alerting            ( 0),  -- initial states
    connected            ( 1),  -- initial states
    fail                ( 2),  -- initial states
    initiated            ( 3),  -- initial states
    queued              ( 4),  -- initial states
    privateData          ( 5),  -- optional parameters
    privateDataInAck     ( 6),  -- optional parameters
    deviceIDOnly         ( 7),  -- misc characteristics
    ackModelMultiStep    ( 8)}  -- misc characteristics

RetrieveCall ::= BIT STRING
{
    privateData          ( 0),  -- optional parameters
    privateDataInAck     ( 1),  -- optional parameters
    deviceIDOnly         ( 2),  -- misc characteristics
    ackModelMultiStep    ( 3)}  -- misc characteristics

SingleStepConference ::= BIT STRING
{
    participationTypeActive    ( 0),  -- optional parameters
    participationTypeSilent    ( 1),  -- optional parameters
    accountCode               ( 2),  -- optional parameters
    authCode                  ( 3),  -- optional parameters
    correlatorData            ( 4),  -- optional parameters
    userData                  ( 5),  -- optional parameters
    privateData               ( 6),  -- optional parameters
    conferencedCallInfoInAck  ( 7),  -- optional parameters
    privateDataInAck          ( 8),  -- optional parameters
    deviceIDOnly              ( 9),  -- misc characteristics
    ackModelMultiStep         (10)}  -- misc characteristics

SingleStepTransfer ::= BIT STRING
{
    accountCode              ( 0),  -- optional parameters
    authCode                 ( 1),  -- optional parameters
    correlatorData           ( 2),  -- optional parameters
    userData                 ( 3),  -- optional parameters
    privateData              ( 4),  -- optional parameters
    connectionParameterInAck ( 5),  -- optional parameters
    endpointDeviceID         ( 6),  -- optional parameters
    resultionConnectionInformation ( 7),  -- optional parameters
    transferredCollInfoInAck ( 8),  -- optional parameters
    privateDataInAck         ( 9),  -- optional parameters
    deviceIDOnly             (10),  -- misc characteristics
    multipleDevices          (11),  -- misc characteristics
    ackModelMultiStep        (12)}  -- misc characteristics

TransferCall ::= BIT STRING
{
    activeCallConnected      ( 0),  -- initial states
    activeCallHold           ( 1),  -- initial states
    heldCallConnected        ( 2),  -- initial states
    heldCallHold             ( 3),  -- initial states
    privateData              ( 4),  -- optional parameters
    connectionsParameterInAck ( 5),  -- optional parameters
    endpointDeviceID         ( 6),  -- optional parameters
    resultingConnectionInformation ( 7),  -- optional parameters
    transferredCallInfoInAck ( 8),  -- optional parameters

```

```

privateDataInAck          ( 9),    -- optional parameters
deviceIDOnly              (10),    -- misc characteristics
multipleDevices           (11),    -- misc characteristics
ackModelMultiStep         (12)}    -- misc characteristics

CallControlEvtsList ::= SEQUENCE
{
    bridged                [0] IMPLICIT Bridged          OPTIONAL,
    callCleared             [1] IMPLICIT CallCleared      OPTIONAL,
    conferenced             [2] IMPLICIT Conferenced      OPTIONAL,
    connectionCleared       [3] IMPLICIT ConnectionCleared OPTIONAL,
    delivered               [4] IMPLICIT Delivered        OPTIONAL,
    digitsDialed            [5] IMPLICIT DigitsDialed     OPTIONAL,
    diverted                [6] IMPLICIT Diverted         OPTIONAL,
    established             [7] IMPLICIT Established      OPTIONAL,
    failed                  [8] IMPLICIT Failed           OPTIONAL,
    held                    [9] IMPLICIT Held             OPTIONAL,
    netwCapsChanged         [10] IMPLICIT NetwCapsChanged  OPTIONAL,
    netwReached             [11] IMPLICIT NetwReached     OPTIONAL,
    offered                 [12] IMPLICIT Offered         OPTIONAL,
    originated              [13] IMPLICIT Originated      OPTIONAL,
    queued                  [14] IMPLICIT Queued          OPTIONAL,
    retrieved               [15] IMPLICIT Retrieved       OPTIONAL,
    serviceInitiated        [16] IMPLICIT ServiceInitiated OPTIONAL,
    transferred             [17] IMPLICIT Transferred     OPTIONAL }

Bridged ::= BIT STRING
{
    correlatorData          ( 0),    -- optional parameters
    userData                ( 1),    -- optional parameters
    servicesPermitted        ( 2),    -- optional parameters
    mediaCallCharacteristics ( 3),    -- optional parameters
    callCharacteristics      ( 4),    -- optional parameters
    bridgedConnectionInfo    ( 5),    -- optional parameters
    callLinkageData          ( 7),    -- optional parameters
    privateData              ( 6)}    -- optional parameters

CallCleared ::= BIT STRING
{
    correlatorData          ( 0),    -- optional parameters
    userData                ( 1),    -- optional parameters
    mediaCallCharacteristics ( 2),    -- optional parameters
    callCharacteristics      ( 3),    -- optional parameters
    callLinkageData          ( 6),    -- optional parameters
    privateData              ( 4),    -- optional parameters
    callIDOnly              ( 5)}    -- miscellaneous characteristics

Conferenced ::= BIT STRING
{
    confereceConnectionsEndpointDeviceID ( 0),    -- optional parameters
    confereceConnectionsresultingConnectionInfo ( 1),    -- optional parameters
    userData                  ( 2),    -- optional parameters
    servicesPermitted          ( 3),    -- optional parameters
    mediaCallCharacteristics    ( 4),    -- optional parameters
    callCharacteristics         ( 5),    -- optional parameters
    bridgedConnectionInfo      ( 6),    -- optional parameters
    privateData                ( 7)}    -- optional parameters

ConnectionCleared ::= BIT STRING
{
    correlatorData          ( 0),    -- optional parameters
    userData                ( 1),    -- optional parameters
    chargingInfo            ( 2),    -- optional parameters
    numberUnitsNumberOfChargingUnits ( 3),    -- optional parameters
    numberUnitsTypeOfUnits   ( 4),    -- optional parameters
    numberUnitsNumberOfCurrencyUnits ( 5),    -- optional parameters
    typeOfChargingInfoSubTotal ( 6),    -- optional parameters
    typeOfChargingInfoTotal   ( 7),    -- optional parameters
    chargingMultiplierATHousandth ( 8),    -- optional parameters
    chargingMultiplierAHundredth ( 9),    -- optional parameters
    chargingMultiplierATenth   (10),    -- optional parameters
    chargingMultiplierOne      (11),    -- optional parameters

```

```

chargingMultiplierTen          (12), -- optional parameters
chargingMultiplierHundred      (13), -- optional parameters
chargingMultiplierThousand      (14), -- optional parameters
servicesPermitted               (15), -- optional parameters
mediaCallCharacteristics        (16), -- optional parameters
callCharacteristics              (17), -- optional parameters
droppedConnectionInfo           (18), -- optional parameters
callLinkageData                 (20), -- optional parameters
privateData                     (19)} -- optional parameters

Delivered ::= BIT STRING
{
    originatingNIDConnection      ( 0), -- optional parameters
    userData                      ( 1), -- optional parameters
    servicesPermitted              ( 2), -- optional parameters
    netwCallingDevice              ( 3), -- optional parameters
    netwCalledDevice               ( 4), -- optional parameters
    mediaCallCharacteristics        ( 5), -- optional parameters
    callCharacteristics             ( 6), -- optional parameters
    connectionInfo                 ( 7), -- optional parameters
    privateData                    ( 8)} -- optional parameters

DigitsDialed ::= BIT STRING
{
    servicesPermitted              ( 0), -- optional parameters
    netwCallingDevice              ( 1), -- optional parameters
    netwCalledDevice               ( 2), -- optional parameters
    diallingConnectionInfo         ( 3), -- optional parameters
    callCharacteristics             ( 4), -- optional parameters
    privateData                    ( 5)} -- optional parameters

Diverted ::= BIT STRING
{
    callingDevice                  ( 0), -- optional parameters
    calledDevice                   ( 1), -- optional parameters
    userData                      ( 2), -- optional parameters
    servicesPermitted              ( 3), -- optional parameters
    mediaCallCharacteristics        ( 4), -- optional parameters
    callCharacteristics             ( 5), -- optional parameters
    connectionInfo                 ( 6), -- optional parameters
    netwCallingDevice              ( 7), -- optional parameters
    netwCalledDevice               ( 8), -- optional parameters
    privateData                    ( 9), -- optional parameters
    sendsDivertedToAll             (10)} -- optional parameters

Established ::= BIT STRING
{
    originatingNIDConnection      ( 0), -- optional parameters
    userData                      ( 1), -- optional parameters
    servicesPermitted              ( 2), -- optional parameters
    netwCallingDevice              ( 3), -- optional parameters
    netwCalledDevice               ( 4), -- optional parameters
    mediaCallCharacteristics        ( 5), -- optional parameters
    callCharacteristics             ( 6), -- optional parameters
    establishedConnectionInfo       ( 7), -- optional parameters
    privateData                    ( 8)} -- optional parameters

Failed ::= BIT STRING
{
    originatingNIDConnection      ( 0), -- optional parameters
    userData                      ( 1), -- optional parameters
    servicesPermitted              ( 2), -- optional parameters
    netwCallingDevice              ( 3), -- optional parameters
    netwCalledDevice               ( 4), -- optional parameters
    mediaCallCharacteristics        ( 5), -- optional parameters
    callCharacteristics             ( 6), -- optional parameters
    failedConnectionInfo           ( 7), -- optional parameters
    privateData                    ( 8), -- optional parameters
    callIDOnly                     ( 9)} -- miscellaneous characteristics

Held ::= BIT STRING
{
    correlatorData                 ( 0), -- optional parameters
    servicesPermitted              ( 1), -- optional parameters

```

```

mediaCallCharacteristics      ( 2),  -- optional parameters
callCharacteristics           ( 3),  -- optional parameters
heldConnectionInfo           ( 4),  -- optional parameters
callLinkageData              ( 6),  -- optional parameters
privateData                   ( 5)}  -- optional parameters

NetwCapsChanged ::= BIT STRING
{
    progressLocationUser      ( 0),  -- optional parameters
    progressLocationPrivateServLocal ( 1),  -- optional parameters
    progressLocationPublicServLocal ( 2),  -- optional parameters
    progressLocationTransit   ( 3),  -- optional parameters
    progressLocationPublicServRemote ( 4),  -- optional parameters
    progressLocationPrivateServRemote ( 5),  -- optional parameters
    progressLocationLocal     ( 6),  -- optional parameters
    progressLocationInternational ( 7),  -- optional parameters
    progressLocationNetwBeyondInterworking ( 8),  -- optional parameters
    progressLocationOther     ( 9),  -- optional parameters
    progressDescriptionISDN   (10),  -- optional parameters
    progressDescriptionQSIG   (11),  -- optional parameters
    progressDescriptionOther  (12),  -- optional parameters
    userData                  (13),  -- optional parameters
    typeOfNetworkISDNPublic   (14),  -- optional parameters
    typeOfNetworkNonISDNPublic (15),  -- optional parameters
    typeOfNetworkISDNPrivate  (16),  -- optional parameters
    typeOfNetworkNonISDNPrivate (17),  -- optional parameters
    typeOfNetworkOther       (18),  -- optional parameters
    eventsProvidedParameter   (19),  -- optional parameters
    eventsProvidedBridged     (20),  -- optional parameters
    eventsProvidedCallCleared (21),  -- optional parameters
    eventsProvidedConferenced (22),  -- optional parameters
    eventsProvidedConnectionCleared (23),  -- optional parameters
    eventsProvidedDelivered   (24),  -- optional parameters
    eventsProvidedDigitsDialed (25),  -- optional parameters
    eventsProvidedDiverted    (26),  -- optional parameters
    eventsProvidedEstablished (27),  -- optional parameters
    eventsProvidedFailed      (28),  -- optional parameters
    eventsProvidedHeld        (29),  -- optional parameters
    eventsProvidedNetwCapsChanged (30),  -- optional parameters
    eventsProvidedNetwReached (31),  -- optional parameters
    eventsProvidedOffered     (32),  -- optional parameters
    eventsProvidedOriginated  (33),  -- optional parameters
    eventsProvidedQueued      (34),  -- optional parameters
    eventsProvidedRetrieved    (35),  -- optional parameters
    eventsProvidedServiceInitiated (36),  -- optional parameters
    eventsProvidedTransferred (37),  -- optional parameters
    servicesPermitted         (38),  -- optional parameters
    mediaCallCharacteristics  (39),  -- optional parameters
    callCharacteristics       (40),  -- optional parameters
    outboundConnectionInfo    (41),  -- optional parameters
    privateData               (42)}  -- optional parameters

NetwReached ::= BIT STRING
{
    originatingNIDConnection ( 0),  -- optional parameters
    userData                 ( 1),  -- optional parameters
    typeOfNetworkISDNPublic   ( 2),  -- optional parameters
    typeOfNetworkNonISDNPublic ( 3),  -- optional parameters
    typeOfNetworkISDNPrivate  ( 4),  -- optional parameters
    typeOfNetworkNonISDNPrivate ( 5),  -- optional parameters
    typeOfNetworkOther       ( 6),  -- optional parameters
    eventsProvidedParameter   ( 7),  -- optional parameters
    eventsProvidedBridged     ( 8),  -- optional parameters
    eventsProvidedCallCleared ( 9),  -- optional parameters
    eventsProvidedConferenced (10),  -- optional parameters
    eventsProvidedConnectionCleared (11),  -- optional parameters
    eventsProvidedDelivered   (12),  -- optional parameters
    eventsProvidedDigitsDialed (13),  -- optional parameters
    eventsProvidedDiverted    (14),  -- optional parameters
    eventsProvidedEstablished (15),  -- optional parameters

```

```

eventsProvidedFailed          (16),      -- optional parameters
eventsProvidedHeld            (17),      -- optional parameters
eventsProvidedNetwCapsChanged (18),      -- optional parameters
eventsProvidedNetwReached     (19),      -- optional parameters
eventsProvidedOffered         (20),      -- optional parameters
eventsProvidedOriginated      (21),      -- optional parameters
eventsProvidedQueued          (22),      -- optional parameters
eventsProvidedRetrieved       (23),      -- optional parameters
eventsProvidedServiceInitiated (24),      -- optional parameters
eventsProvidedTransferred     (25),      -- optional parameters
servicesPermitted             (26),      -- optional parameters
mediaCallCharacteristics      (27),      -- optional parameters
callCharacteristics           (28),      -- optional parameters
outboundConnectionInfo        (29),      -- optional parameters
netwCallingDevice             (30),      -- optional parameters
netwCalledDevice              (31),      -- optional parameters
privateData                   (32)}      -- optional parameters

Offered ::= BIT STRING
{
    originatingNIDConnection    ( 0),      -- optional parameters
    userData                    ( 1),      -- optional parameters
    servicesPermitted           ( 2),      -- optional parameters
    netwCallingDevice           ( 3),      -- optional parameters
    netwCalledDevice            ( 4),      -- optional parameters
    mediaCallCharacteristics     ( 5),      -- optional parameters
    callCharacteristics          ( 6),      -- optional parameters
    offeredConnectionInfo        ( 7),      -- optional parameters
    privateData                  ( 8)}      -- optional parameters

Originated ::= BIT STRING
{
    originatingDevice           ( 0),      -- optional parameters
    servicesPermitted           ( 1),      -- optional parameters
    netwCallingDevice           ( 2),      -- optional parameters
    netwCalledDevice            ( 3),      -- optional parameters
    mediaCallCharacteristics     ( 4),      -- optional parameters
    callCharacteristics          ( 5),      -- optional parameters
    originatedConnectionInfo     ( 6),      -- optional parameters
    privateData                  ( 7)}      -- optional parameters

Queued ::= BIT STRING
{
    numberQueued                ( 0),      -- optional parameters
    callsInFront                ( 1),      -- optional parameters
    userData                    ( 2),      -- optional parameters
    servicesPermitted           ( 3),      -- optional parameters
    netwCallingDevice           ( 4),      -- optional parameters
    netwCalledDevice            ( 5),      -- optional parameters
    mediaCallCharacteristics     ( 6),      -- optional parameters
    callCharacteristics          ( 7),      -- optional parameters
    queuedConnectionInfo        ( 8),      -- optional parameters
    privateData                  ( 9)}      -- optional parameters

Retrieved ::= BIT STRING
{
    correlatorData              ( 0),      -- optional parameters
    servicesPermitted           ( 1),      -- optional parameters
    mediaCallCharacteristics     ( 2),      -- optional parameters
    callCharacteristics          ( 3),      -- optional parameters
    retrievedConnectionInfo      ( 4),      -- optional parameters
    callLinkageData              ( 6),      -- optional parameters
    privateData                  ( 5)}      -- optional parameters

ServiceInitiated ::= BIT STRING
{
    servicesPermitted           ( 0),      -- optional parameters
    mediaCallCharacteristics     ( 1),      -- optional parameters
    callCharacteristics          ( 2),      -- optional parameters
    initiatedConnectionInfo      ( 3),      -- optional parameters
    netwCallingDevice           ( 4),      -- optional parameters
    netwCalledDevice            ( 5),      -- optional parameters
    privateData                  ( 6)}      -- optional parameters

```

```

Transferred ::= BIT STRING
{
    transferredConnectionsEndpointDeviceID      ( 0),  -- optional parameters
    transferredConnectionsResultingConnectionInfo ( 1),  -- optional parameters
    userData                                    ( 2),  -- optional parameters
    chargingInfo                                ( 3),  -- optional parameters
    numberUnitsNumberOfChargingUnits            ( 4),  -- optional parameters
    numberUnitsTypeOfUnits                      ( 5),  -- optional parameters
    numberUnitsNumberOfCurrencyUnits            ( 6),  -- optional parameters
    typeOfChargingInfoSubTotal                  ( 7),  -- optional parameters
    typeOfChargingInfoTotal                    ( 8),  -- optional parameters
    chargingMultiplierAThousandth              ( 9),  -- optional parameters
    chargingMultiplierAHundredth               (10),  -- optional parameters
    chargingMultiplierATenth                   (11),  -- optional parameters
    chargingMultiplierOne                      (12),  -- optional parameters
    chargingMultiplierTen                      (13),  -- optional parameters
    chargingMultiplierHundred                  (14),  -- optional parameters
    chargingMultiplierThousand                 (15),  -- optional parameters
    servicesPermitted                          (16),  -- optional parameters
    mediaCallCharacteristics                   (17),  -- optional parameters
    callCharacteristics                        (18),  -- optional parameters
    connectionInfo                            (19),  -- optional parameters
    privateData                                (20)}  -- optional parameters

CallAssociatedServList ::= SEQUENCE
{
    associateData          [0] IMPLICIT AssociateData          OPTIONAL,
    cancelTelephonyTones  [1] IMPLICIT CancelTelephonyTones   OPTIONAL,
    generateDigits         [2] IMPLICIT GenerateDigits         OPTIONAL,
    generateTelephonyTones [3] IMPLICIT GenerateTelephonyTones OPTIONAL,
    sendUserInformation    [4] IMPLICIT SendUserInformation    OPTIONAL}

AssociateData ::= BIT STRING
{
    accountCode      ( 0),  -- optional parameters
    authCode         ( 1),  -- optional parameters
    correlatorData   ( 2),  -- optional parameters
    callQualifyingData ( 3),  -- optional parameters
    privateData      ( 4),  -- optional parameters
    privateDataInAck ( 5),  -- optional parameters
    deviceIDOnly     ( 6),  -- misc characteristics
    rejectsRequestsWithOldConnectionID ( 7),  -- misc characteristics
    ackModelMultiStep ( 8)}  -- misc characteristics

CancelTelephonyTones ::= BIT STRING
{
    privateData      ( 0),  -- optional parameters
    privateDataInAck ( 1),  -- optional parameters
    ackModelMultiStep ( 2)}  -- misc characteristics

GenerateDigits ::= BIT STRING
{
    digitModeDTMF      ( 0),  -- optional parameters
    digitModePulse     ( 1),  -- optional parameters
    toneDuration       ( 2),  -- optional parameters
    pulseRate          ( 3),  -- optional parameters
    pauseDuration      ( 4),  -- optional parameters
    privateData        ( 5),  -- optional parameters
    privateDataInAck   ( 6),  -- optional parameters
    deviceIDOnly       ( 7),  -- misc characteristics
    supportsDTMFTonesABCD ( 8),  -- misc characteristics
    supportsPauseToneChar ( 9),  -- misc characteristics
    ackModelMultiStep  (10)}  -- misc characteristics

GenerateTelephonyTones ::= BIT STRING
{
    toneToSendBeep      ( 0),  -- optional parameters
    toneToSendBilling    ( 1),  -- optional parameters
    toneToSendBusy       ( 2),  -- optional parameters
    toneToSendCarrier    ( 3),  -- optional parameters
    toneToSendConfirmation ( 4),  -- optional parameters

```