
**Information technology —
Telecommunications and information
exchange between systems — ASN.1 for
Computer Supported
Telecommunications Applications (CSTA)
Phase III**

*Technologies de l'information — Téléinformatique — ASN.1 pour
applications en télécommunications supportées par ordinateur (CSTA)
en phase III*



Reference number
ISO/IEC 18052:2012(E)

AVA Documents
Document Sharing Hub
© ISO/IEC 2012



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2012

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.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword	x
Introduction.....	xi
1 Scope	1
2 Conformance	1
2.1 Static Requirements	1
2.2 Dynamic Requirements.....	1
2.3 PICS Requirement	1
3 Normative References.....	2
4 Terms and definitions	2
5 CSTA Service Definition Model.....	2
5.1 CSTA Application Layer Structure	2
5.2 Remote Operations	3
5.3 The CSTA Service Response	3
5.4 Cross Referencing of Event Reports.....	3
5.5 Handling of Private Data	4
6 Interconnection service boundary.....	4
7 Security	4
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	5
8.2.2 Encoding of application context information	5
9 CSTA parameter types	10
9.1 Switching function objects.....	10
9.2 Device Identifiers	11
9.3 Call and connection identifiers	13
9.4 Connection states	14
9.5 Status reporting.....	15
9.6 Device and feature types and other parameters	18
9.7 Security	24
9.8 Common extensions	25
9.9 Call control	26
9.10 Capability Exchange	30
9.11 Call Detail Record.....	95
9.12 Charge information	98
9.13 Data call types	99
9.14 Escape types.....	100
9.15 Media services	101
9.16 Physical device features.....	104
9.17 Data Collection	106
9.18 Event Cause	107
9.19 Error Value	110
10 Event Report Service	116
11 Capability exchange services	122
11.1 Services	122
11.1.1 Get CSTA-Features.....	122

11.1.2	Get logical device information	124
11.1.3	Get physical device information	126
11.1.4	Get switching function capabilities	127
11.1.5	Get switching function devices.....	131
11.1.6	Switching function devices	133
12	System services.....	134
12.1	Registration services	134
12.1.1	Change system status filter	134
12.1.2	System register	135
12.1.3	System register abort.....	136
12.1.4	System register cancel.....	137
12.2	Services	138
12.2.1	Request system status.....	138
12.2.2	System status.....	139
12.2.3	Switching function capabilities changed	140
12.2.4	Switching function devices changed.....	141
12.2.5	Get Registrations	142
12.2.6	Registration Info	143
13	Monitoring services	144
13.1	Services	144
13.1.1	Change monitor filter	144
13.1.2	Monitor start.....	145
13.1.3	Monitor stop	146
13.1.4	Get Monitor.....	147
13.1.5	Monitor Info	148
14	Snapshot services	149
14.1	Services	149
14.1.1	Snapshot call.....	149
14.1.2	Snapshot device	151
14.1.3	Snapshot calldata	152
14.1.4	Snapshot devicedata	153
15	Call control services and events.....	154
15.1	Services	154
15.1.1	Accept call	154
15.1.2	Alternate call	155
15.1.3	Answer call.....	156
15.1.4	Call back call-related	157
15.1.5	Call back message call-related.....	158
15.1.6	Camp on call.....	159
15.1.7	Clear call	160
15.1.8	Clear Connection	161
15.1.9	Conference call	162
15.1.10	Consultation call	163
15.1.11	Deflect call	165
15.1.12	Dial digits.....	167
15.1.13	Directed pickup call	168
15.1.14	Group pickup call.....	169
15.1.15	Hold call	170
15.1.16	Intrude call.....	171
15.1.17	Join call.....	172
15.1.18	Make call	173
15.1.19	Make connection.....	175
15.1.20	Make predictive call	177
15.1.21	Park call	179
15.1.22	Reconnect call.....	180
15.1.23	Retrieve call.....	181
15.1.24	Single step conference call	182
15.1.25	Send Message	184

15.1.26	Single step transfer call	186
15.1.27	Transfer call	188
15.2	Events	189
15.2.1	Bridged	189
15.2.2	Call cleared	191
15.2.3	Conferenced	192
15.2.4	Connection cleared	194
15.2.5	Delivered	196
15.2.6	Digits dialed	198
15.2.7	Diverted	200
15.2.8	Established	202
15.2.9	Failed	204
15.2.10	Held	206
15.2.11	Network capabilities changed	207
15.2.12	Network reached	209
15.2.13	Offered	211
15.2.14	Originated	213
15.2.15	Queued	215
15.2.16	Retrieved	217
15.2.17	Service initiated	218
15.2.18	Transferred	220
16	Call associated features	222
16.1	Services	222
16.1.1	Associate data	222
16.1.2	Cancel telephony tones	224
16.1.3	Change Connection Information	225
16.1.4	Generate digits	226
16.1.5	Generate telephony tones	227
16.1.6	Send user information	228
16.2	Events	229
16.2.1	Call information	229
16.2.2	Charging	230
16.2.3	Digits generated	231
16.2.4	Telephony tones generated	232
16.2.5	Service completion failure	233
17	Media attachment services and events	234
17.1	Services	234
17.1.1	Attach media service	234
17.1.2	Detach media service	236
17.2	Events	237
17.2.1	Media attached	237
17.2.2	Media detached	238
18	Routeing services	239
18.1	Registration services	239
18.1.1	Route register	239
18.1.2	Route register abort	240
18.1.3	Route register cancel	241
18.2	Services	242
18.2.1	Re-Route	242
18.2.2	Route end	243
18.2.3	Route reject	244
18.2.4	Route request	245
18.2.5	Route select	247
18.2.6	Route used	248
19	Physical device features	249
19.1	Services	249
19.1.1	Button press	249
19.1.2	Get auditory apparatus information	250

19.1.3	Get button information	251
19.1.4	Get display	252
19.1.5	Get hookswitch status	253
19.1.6	Get lamp information	254
19.1.7	Get lamp mode	255
19.1.8	Get message waiting indicator	256
19.1.9	Get microphone gain	257
19.1.10	Get microphone mute	258
19.1.11	Get ringer status	259
19.1.12	Get speaker mute	260
19.1.13	Get speaker volume	261
19.1.14	Set button information	262
19.1.15	Set display	263
19.1.16	Set hookswitch status	264
19.1.17	Set lamp mode	265
19.1.18	Set message waiting indicator	266
19.1.19	Set microphone gain	267
19.1.20	Set microphone mute	268
19.1.21	Set ringer status	269
19.1.22	Set speaker mute	270
19.1.23	Set speaker volume	271
19.2	Events	272
19.2.1	Button information	272
19.2.2	Button press	273
19.2.3	Display updated	274
19.2.4	Hookswitch	275
19.2.5	Lamp mode	276
19.2.6	Message waiting	277
19.2.7	Microphone gain	278
19.2.8	Microphone mute	279
19.2.9	Ringer status	280
19.2.10	Speaker mute	281
19.2.11	Speaker volume	282
20	Logical device features	283
20.1	Services	283
20.1.1	Call back non-call-related	283
20.1.2	Call back message non-call-related	284
20.1.3	Cancel call back	285
20.1.4	Cancel call back message	286
20.1.5	Get agent state	287
20.1.6	Get auto answer	288
20.1.7	Get auto work mode	289
20.1.8	Get Call Back	290
20.1.9	Get caller id status	291
20.1.10	Get do not disturb	292
20.1.11	Get forwarding	293
20.1.12	Get last number dialed	294
20.1.13	Get presence state	295
20.1.14	Get routeing mode	296
20.1.15	Set agent state	297
20.1.16	Set auto answer	298
20.1.17	Set auto work mode	299
20.1.18	Set caller id status	300
20.1.19	Set do not disturb	301
20.1.20	Set forwarding	302
20.1.21	Set presence state	303
20.1.22	Set routeing mode	304
20.2	Events	305
20.2.1	Agent busy	305

20.2.2	Agent logged off	306
20.2.3	Agent logged on	307
20.2.4	Agent not ready	308
20.2.5	Agent ready	309
20.2.6	Agent working after call.....	310
20.2.7	Auto answer	311
20.2.8	Auto work mode	312
20.2.9	Call back	313
20.2.10	Call back message	314
20.2.11	Caller id status	315
20.2.12	Do not disturb	316
20.2.13	Forwarding	317
20.2.14	Presence state	318
20.2.15	Routeing mode	319
21	Device maintenance events	320
21.1	Events	320
21.1.1	Back in service	320
21.1.2	Device capabilities changed	321
21.1.3	Out of service.....	322
21.1.4	Partially in Service.....	323
22	I/O services	324
22.1	Registration services	324
22.1.1	I/O register.....	324
22.1.2	I/O register abort.....	325
22.1.3	I/O register cancel	326
22.2	Services	327
22.2.1	Data path resumed	327
22.2.2	Data path suspended	328
22.2.3	Fast data	329
22.2.4	Resume data path.....	330
22.2.5	Send broadcast data	331
22.2.6	Send data	332
22.2.7	Send multicast data.....	333
22.2.8	Start data path	334
22.2.9	Stop data path.....	335
22.2.10	Suspend data path	336
23	Data Collection Services.....	337
23.1	Services	337
23.1.1	Data Collected.....	337
23.1.2	Data Collection Resumed	339
23.1.3	Data Collection Suspended	340
23.1.4	Resume Data Collection	341
23.1.5	Start Data Collection	342
23.1.6	Stop Data Collection	343
23.1.7	Suspend Data Collection	344
24	Voice unit services and events	345
24.1	Services	345
24.1.1	Activate.....	345
24.1.2	Clear	346
24.1.3	Concatenate message.....	347
24.1.4	Deactivate.....	348
24.1.5	Delete message	349
24.1.6	Play message	350
24.1.7	Query voice attribute.....	351
24.1.8	Queue.....	353
24.1.9	Record message.....	354
24.1.10	Reposition	355
24.1.11	Resume.....	356

24.1.12	Review	357
24.1.13	Set voice attribute	358
24.1.14	Start	360
24.1.15	Stop	361
24.1.16	Suspend	362
24.1.17	Synthesize message	363
24.2	Events	364
24.2.1	Bookmark Reached	364
24.2.2	Completed	365
24.2.3	DTMF Detected	366
24.2.4	Emptied	367
24.2.5	Interruption Detected	368
24.2.6	Not Recognized	369
24.2.7	Play	370
24.2.8	Recognized	371
24.2.9	Record	372
24.2.10	Review	373
24.2.11	Silence Timeout Expired	374
24.2.12	Speech Detected	375
24.2.13	Started	376
24.2.14	Stop	377
24.2.15	Suspend play	378
24.2.16	Suspend record	379
24.2.17	Voice attribute changed	380
24.2.18	VoiceErrorOccurred	382
25	Call detail record services	383
25.1	Services	383
25.1.1	Call detail records notification	383
25.1.2	Call detail records report	384
25.1.3	Send stored call detail records	385
25.1.4	Start call detail records transmission	386
25.1.5	Stop call detail records transmission	387
26	Location Services	388
26.1	Services	388
26.1.1	Get Location Information	388
26.1.2	Set Location Information	389
26.1.3	Location tracking session resumed	390
26.1.4	Location tracking session suspended	391
26.1.5	Resume Location tracking session	392
26.1.6	Location Information Report	393
26.1.7	Start Location Tracking Session	394
26.1.8	Stop Location Tracking Session	396
26.1.9	Suspend Location Tracking Session	397
26.1.10	Get Location Tracking Capabilities	398
26.1.11	Get Location Tracking Sessions	400
26.1.12	Location Session Info	401
27	Vendor specific extensions services and events	402
27.1	Registration services	402
27.1.1	Escape register	402
27.1.2	Escape register abort	403
27.1.3	Escape register cancel	404
27.2	Services	405
27.2.1	Escape	405
27.2.2	Private data version selection	406
27.3	Events	407
27.3.1	Private event	407
Annex A	(normative) Protocol Implementation Conformance Statement (PICS) Proforma	408

Bibliography.....578

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. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. 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 document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 18052 was prepared by Ecma International (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.

This third edition cancels and replaces the second edition (ISO/IEC 18052:2011), which has been technically revised.

Introduction

This International Standard specifies the ASN.1 for Phase III of Services for Computer Supported Telecommunications Applications (CSTA), ISO/IEC 18051 6th edition. This International Standard is an alternative to the XML Schema Definitions (XSD) in ECMA-323.

ASN.1 Tools Acknowledgement

The ASN.1 specified in this International Standard has been checked for conformance with the ASN.1 Standard by the OSS Nokalva, Inc. ASN.1 Tools.

Information technology — Telecommunications and information exchange between systems — ASN.1 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, 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 27 inclusive contains CSTA-specific message templates in ASN.1.

The APDUs are exchanged in the context of an application association.

2 Conformance

Implementations may include any ASN.1 message definition part specified in this International Standard, as long as it satisfies the minimum conformance requirements as specified in Clause 2 of ISO/IEC 18051.

As specified and required in ISO/IEC 18051 a Protocol Implementation Conformance Statement (PICS) shall be made; for implementations of this standard its edition shall be declared as the protocol version in the PICS.

2.1 Static Requirements

To conform to this International Standard, a system shall support the transfer syntax (derived from the encoding rules specified in ISO/IEC 8825-1) named {joint-iso-ccitt asn1(1) basic-encoding(1) }; CSTA APDUs shall be interpreted according to ISO/IEC 8824-1.

2.2 Dynamic Requirements

To conform to this International Standard, a system shall:

- a. follow the procedures as specified in this 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.

2.3 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.

3 Normative References

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 18051:2012, *Information technology — Telecommunications and information exchange between systems — Services for Computer Supported Telecommunications Applications (CSTA) Phase III (ECMA-269)*

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 (ECMA TR/72)*

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 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 13712-2:1995, *Information technology — Remote Operations: OSI realisations — Remote Operations Service Element (ROSE) service definition (this corresponds to ITU-T Rec. X.881, 1994)*

ISO/IEC 13712-3:1995, *Information technology — Remote Operations: OSI realisations — Remote Operations Service Element (ROSE) protocol specification (this corresponds to ITU-T Rec. X.882, 1994)*

ISO/IEC 8824-1:2008, *Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation (this corresponds to ITU-T X.680, 2008)*

ISO/IEC 8825-1:2008, *Information technology — ASN.1 encoding rules — Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER) (this corresponds to ITU-T Rec. ITU-T X.690, 2008)*

RFC 4119, *A Presence-based GEOPRIV Location Object Format*

4 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC TR 18053 apply. In addition, the following terms defined in other International Standards 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. The protocol specifications in ISO/IEC 18051 determine the class of the operation.

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, implementations shall use 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, implementations shall use 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, implementations shall use the operations Class 5, defined in ISO/IEC 13712-2 as:

- Asynchronous, outcome not reported.

Implementations 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:

- a. Specific services may have an unconfirmed mode where responses to correct requests are not returned.
- b. 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. a CSTA Object ID (Call ID or Device ID),
- b. an earlier Monitor request; or
- c. 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 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 APDUs in this Standard are exchanged in the context of an association and by using the Remote Operations protocol defined in ISO/IEC 13712-3, which assumes that certain services are provided by the underlying layers.

7 Security

The authentication mechanism of ACSE may be used.

8 Association Management

The APDUs in this Standard are exchanged 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.

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

ACSEUserInformationForCSTA ::= CHOICE
{
    oldDefinition                OldACSEUserInformationForCSTA,
    newDefinition [0] IMPLICIT   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), -- ECMA-180
    versionTwo      ( 1), -- ECMA-218
    versionThree    ( 2), -- ECMA-285, 1st Edition
    versionFour     ( 3), -- ISO/IEC 18052
    versionFive     ( 4), -- ECMA-285, 2nd Edition
    versionSix      ( 5), -- ECMA-323
    versionSeven    ( 6), -- ECMA-323, 2nd Edition
    versionEight    ( 7), -- ECMA-323, 3rd Edition
    versionNine     ( 8), -- ECMA-323, 4th Edition, Tag option
    versionTen      ( 9), -- ECMA-323, 4th Edition, short Tag option
    versionEleven   (10), -- ECMA-323, 4th Edition, dynamic transformation option
    versionTwelve   (11), -- ECMA-323, 5th Edition, Tag option
    versionThirteen (12), -- ECMA-323, 5th Edition, short Tag option
    versionFourteen (13), -- ECMA-323, 5th Edition, dynamic transformation option
    versionFifteen  (14), -- ECMA-285, 3rd Edition
    versionSixteen  (15), -- ECMA-323, 6th Edition, Tag option
    versionSeventeen (16), -- ECMA-323, 6th Edition, short Tag option
    versionEighteen (17), -- ECMA-323, 6th Edition, dynamic transformation option
    versionNineteen (18), -- ECMA-285, 4th Edition
    versionTwenty   (19), -- Reserved for future use
    versionTwentyone (20), -- Reserved for future use
    versionTwentytwo (21), -- Reserved for future use
    versionTwentythree (22), -- Reserved for future use
    versionTwentyfour (23) } -- 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),
    makeConnection      (25),
    makePredictiveCall  (18),
    parkCall           (19),
    reconnectCall       (20),
    retrieveCall        (21),
    sendMessage        (26),
    singleStepConference (22),
    singleStepTransfer  (23),
    transferCall        (24) }

```

```

CallAssociatedServices ::= BIT STRING
{
    associateData      ( 0),
    cancelTelephonyTones ( 1),
    changeConnectionInformation ( 5),
    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
{
    activate                                (12),
    clear                                  (13),
    concatenateMessage                     ( 0),
    deactivate                             (14),
    deleteMessage                         ( 1),
    playMessage                            ( 2),
    queryVoiceAttribute                    ( 3),
    recordMessage                          ( 4),
    reposition                             ( 5),
    resume                                 ( 6),
    review                                 ( 7),
    setVoiceAttribute                      ( 8),
    stop                                   ( 9),
    suspend                                (10),
    synthesizeMessage                      (11) }

LocationServices ::= BIT STRING
{
    getLocationInformation                  ( 0),
    setLocationInformation                  ( 1),
    locationTrackingSessionResumed          ( 2),
    locationTrackingSessionSuspended        ( 3),
    resumeLocationTrackingSession           ( 4),
    locationInformationReport                ( 5),
    startLocationTrackingSession             ( 6),
    stopLocationTrackingSession              ( 7),
    suspendLocationTrackingCapabilities      ( 8),
    locationSessionInfo                     ( 9) }

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),
    agentLoggedIn                            ( 1),
    agentLoggedOff                           ( 2),
    agentNotReady                            ( 3),
    agentReady                               ( 4),
    agentWorkingAfterCall                    ( 5),
    autoAnswer                               ( 6),
    autoWorkMode                             ( 7),
    callBack                                 ( 8),
    callBackMessage                          ( 9),
    callerIDStatus                           (10),
    doNotDisturb                             (11),
    forwarding                               (12),
    presenceState                            (14),
    routeingMode                             (13) }

DeviceMaintenanceEvents ::= BIT STRING
{
    backInService                            ( 0),
    deviceCapabilityChanged                   ( 2),
    outOfService                             ( 1),
    partiallyInService                       ( 3) }

VoiceUnitEvents ::= BIT STRING
{
    bookmarkReached                          ( 7),
    completed                                ( 8),
    dtmfDetected                             ( 9),
    emptied                                  (10),
    interruptionDetected                     (11),

```

```

notRecognized          (12),
play                   ( 1),
recognized              (13),
record                  ( 3),
review                  ( 5),
started                 (14),
silenceTimeoutExpired  (15),
speechDetected          (16),
stop                    ( 0),
suspendPlay             ( 2),
suspendRecord           ( 4),
voiceAttributesChange   ( 6),
voiceErrorOccured       (17) }

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

END -- of CSTA-application-context-information-csta3

```

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) };

ExtendedDeviceID ::= [APPLICATION 9] CHOICE
{
    deviceIdentifier      DeviceID,
    notKnown              [ 7] IMPLICIT  NULL,
    restricted            [ 8] IMPLICIT  NULL }

CallingDeviceID ::= [APPLICATION 1] CHOICE
{
    deviceIdentifier      DeviceID,
    notKnown              [ 7] IMPLICIT  NULL,
    restricted            [ 8] IMPLICIT  NULL }

CallingDevice ::= CallingDeviceID

CalledDeviceID ::= [APPLICATION 2] CHOICE
{
    deviceIdentifier      DeviceID,
    notKnown              [ 7] IMPLICIT  NULL,
    restricted            [ 8] IMPLICIT  NULL }

CalledDevice ::= CalledDeviceID

SubjectDeviceID ::= [APPLICATION 3] CHOICE
{
    deviceIdentifier      DeviceID,
    notKnown              [ 7] IMPLICIT  NULL,
    restricted            [ 8] IMPLICIT  NULL }

RedirectionDeviceID ::= [APPLICATION 4] CHOICE
{
    numberdialed          DeviceID,
    notKnown              [ 7] IMPLICIT  NULL,
    notRequired           [ 8] IMPLICIT  NULL,
    notSpecified          [ 9] IMPLICIT  NULL,
    restricted            [10] IMPLICIT  NULL }

LastRedirectionDevice ::= RedirectionDeviceID

AssociatedCallingDeviceID ::= [APPLICATION 5] CHOICE
{
    deviceIdentifier      DeviceID,
    notKnown              [ 7] IMPLICIT  NULL,
    restricted            [ 8] IMPLICIT  NULL }

AssociatedCallingDevice ::= AssociatedCallingDeviceID

```

```

AssociatedCalledDeviceID ::= [APPLICATION 6] CHOICE
{
    deviceIdentifier          DeviceID,
    notKnown                  [ 7] IMPLICIT    NULL,
    restricted                  [ 8] IMPLICIT    NULL }

AssociatedCalledDevice ::= AssociatedCalledDeviceID

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

NetworkCallingDevice ::= NetworkCallingDeviceID

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

NetworkCalledDevice ::= NetworkCalledDeviceID

DeviceID ::= SEQUENCE
{
    deviceIdentifier CHOICE
        {
            dialingNumber      [ 0] IMPLICIT    NumberDigits,
            deviceNumber        [ 1] IMPLICIT    DeviceNumber,
            implicitPublic      [ 2] IMPLICIT    NumberDigits,
            explicitPublic      [ 3] PublicTON,
            implicitPrivate     [ 4] IMPLICIT    NumberDigits,
            explicitPrivate     [ 5] PrivateTON,
            other                [ 6] IMPLICIT    OtherPlan },
    privateNumber              [ 0] NumberDigits    OPTIONAL,
    privateName                [ 1] IA5String        OPTIONAL,
    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-identifier

```

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

LocalDeviceID ::= CHOICE
{
    staticID      DeviceID,
    dynamicID     [3] IMPLICIT      OCTET STRING }

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,
          restricted     [1] 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, RegistrationTypeList, RegistrationList,
MonitorList, LocationSessionInfoList;

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, LocationInfoList, CallLinkageData 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) }
LocCrossRefID FROM CSTA-device-feature-types
{ iso( 1) identified-organization( 3) icd-ecma( 12)
  standard( 0) csta3( 285) device-feature-types( 127) };

MonitorObject ::= CSTAObject

MonitorCrossRefID ::= [APPLICATION 21] IMPLICIT OCTET STRING

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) }

MonitorList ::= SEQUENCE OF MonitorListEntry

MonitorListEntry ::= SEQUENCE
{
    monitorCrossRefID    MonitorCrossRefID,
    monitorObject        DeviceID }

RegistrationTypeList ::= SEQUENCE OF RegistrationType

RegistrationType ::= ENUMERATED
{
    escapeRegister      (1),
    ioRegister          (2),
    routeRegister       (3),
    sysStatRegister     (4) }

RegistrationList ::= SEQUENCE OF RegistrationListEntry

RegistrationListEntry ::= SEQUENCE
{
    registrationType      RegistrationType,
    registrationID        IA5String,
    registrationObject    DeviceID                                OPTIONAL }

LocationSessionInfoList ::= SEQUENCE OF LocationSessionListEntry

LocationSessionListEntry ::= SEQUENCE
{
    locCrossRefID        LocCrossRefID,
    locDevice            DeviceID }

MonitorMediaClass ::= BIT STRING
{
    voice                ( 0),
    data                 ( 1),
    image                ( 2),
    audio                ( 4),
    other                ( 3),
    notKnown             ( 5),
    chat                 ( 6),
    email                ( 7),
    message              ( 8),
    im                   ( 9),
    sms                  (10),
    mms                  (11) }

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

SnapshotDeviceResponseInfo ::= SEQUENCE
{
    connectionIdentifier    ConnectionID,
    endpointDevice          [3] IMPLICIT SubjectDeviceID          OPTIONAL,
    localCallState          CallState,
    servicesPermitted       [0] IMPLICIT ServicesPermitted       OPTIONAL,
    mediaServiceInfoList    [1] IMPLICIT DeviceMediaInfoList      OPTIONAL,
    mediaCallCharacteristics [2] IMPLICIT MediaCallCharacteristics OPTIONAL,
    callLinkageData         [4] IMPLICIT CallLinkageData          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,
   locationInfo           [2] IMPLICIT LocationInfoList             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, DeviceHistory, SubjectOfCall, LanguagePreferences, MessageInfo,
LocReason, LocCrossRefID, ResourceID, PresenceState;

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) }
```

EventCause FROM CSTA-event-causes

```
{ iso( 1) identified-organization( 3) icd-ecma( 12)
    standard( 0) csta3( 285) event-causes( 121) };
```

AccountInfo ::= OCTET STRING

AgentID ::= OCTET STRING

AgentPassword ::= OCTET STRING

AgentState ::= ENUMERATED

```
{ agentNotReady          (0),
  agentNull              (1),
  agentReady             (2),
  agentBusy              (3),
  agentWorkingAfterCall  (4) }
```

AuthCode ::= OCTET STRING

CorrelatorData ::= OCTET STRING

IOCrossRefID ::= CHOICE

```
{ switchProvided          [0] OCTET STRING ,
  computerProvided        [1] OCTET STRING }
```

IORegisterReqID ::= OCTET STRING

MessageID ::= OCTET STRING

ParticipationType ::= ENUMERATED

```
{ silent                  (0),
  active                   (1),
  none                     (2) }
```

RouteRegisterReqID ::= OCTET STRING

RouteingCrossRefID ::= [APPLICATION 24] IMPLICIT OCTET STRING

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

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,
  ringDuration   [0] IMPLICIT INTEGER 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) }

LocReason ::= ENUMERATED
{   connectionToDeviceOrNetworkRestored ( 0),
    performanceConditionRelieved        ( 1) }

LocCrossRefID ::= IA5String

CurrentState ::= ENUMERATED
{   dtmfDetected        ( 6),
    formed              ( 7),
    play                ( 1),
    playing              ( 8),
    ready               ( 9),
    record              ( 2),
    review              ( 5),
    speechDetected      (10),
    started             (11),
    stop                ( 0),
    suspendPlay         ( 3),
    suspendRecord       ( 4) }

ResourceID ::= IA5String

```

```

MessageInfo ::= SEQUENCE
{
    contents                IA5String,
    contentTypeAndSubtype   IA5String OPTIONAL,
    contentsLength           INTEGER    OPTIONAL }

DeviceHistory ::= SEQUENCE
{
    oldDeviceID              DeviceID,
    eventCause               [1] IMPLICIT EventCause OPTIONAL,
    oldConnectionID          [2] IMPLICIT ConnectionID OPTIONAL }

SubjectOfCall ::= IA5String

LanguagePreferences ::= IA5String

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

UserData ::= [APPLICATION 29] OCTET STRING

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) }

```

```

PresenceState ::= BIT STRING

```

```

{   available          ( 0),
    away                ( 1),
    busy                ( 2),
    calling             ( 3),
    dnd                 ( 4),
    meeting             ( 5),
    offline             ( 6),
    unknown             ( 7),
    vacation            ( 8),
    applicationSpec0    ( 9),
    applicationSpec1    (10),
    applicationSpec2    (11),
    applicationSpec3    (12),
    applicationSpec4    (13),
    applicationSpec5    (14),
    applicationSpec6    (15),
    applicationSpec7    (16),
    applicationSpec8    (17),
    applicationSpec9    (18),
    applicationSpec10   (19) }

```

```

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, LocationInfo, LocationInfoList,
ProgressIndicator, ServicesPermitted;

IMPORTS
TimeInfo FROM CSTA-security
{ iso( 1) identified-organization( 3) icd-ecma( 12)
  standard( 0) csta3( 285) security( 128) }
CallControlServices, CallAssociatedServices, MediaAttachmentServices, RouteingServices,
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),
  lowPriorityCall         (11),
  priorityCall           ( 1),
  highPriorityCall        (12),
  maintainanceCall       ( 2),
  directAgent            ( 3),
  assistCall             ( 4),
  voiceUnitCall          ( 5),
  privateCall            ( 6),
  personalCall           ( 7),
  sensitiveCall          ( 8),
  confidentialCall       ( 9),
  encryptedCall          (10) }

CallQualifyingData ::= OCTET STRING

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),
    allCleared           ( 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)) }

Method ::= ENUMERATED -- http://www.iana.org/assignments/method-tokens/method-tokens.xhtml
{
    a-GPS          ( 0), -- "Method" defines, what location tracking system has been used
    aOA            ( 1), -- to allocate the presented Location Data
    cell           ( 2),
    derived        ( 3),
    deviceAssistedAGps ( 4),
    deviceBasedAGps ( 5),
    dHCP           ( 6),
    gPS            ( 7),
    handsetAflt    ( 8),
    handsetEflt    ( 9),
    hybridAGps     (10),
    ipdl           (11),
    lldpMed        (12),
    manual         (13),
    mpl            (14),
    nmr            (15),
    otdoa          (16),
    rfid           (17),
    rssi           (18),
    rssiRtt        (19),
    rtt            (20),
    ta             (21),
    taNmr          (22),
    triangulation  (23),
    utdoa          (24),
    wiremap        (25),
    locMet80211    (26) }

NenaCompanyIDType ::= IA5String -- NenaCompanyIds at http://www.nena.org/nena-company-id

TelUri247Type ::= IA5String -- must be a RFC-3966 conform Tel URI

ProvidedBy ::= SEQUENCE -- This DataType contains information about the organization
{
    -- which provided the allocated location data
    dataproviderID NenaCompanyIDType,
    telUri          TelUri247Type, -- Standard Telephone URI,
    uRL             IA5String } -- must be a emergency number, available 24/7

UsageRules ::= SEQUENCE
{
    retransmissionAllowed BOOLEAN,
    retentionExpires       TimeInfo, -- TimeInfo Already defined in CSTA-security
    rulesetReference       IA5String, -- rulesetReference must be a URI
    noteWell               IA5String OPTIONAL }

GpLocationInfo ::= SEQUENCE
{
    civicLocGml SEQUENCE
    {
        gmlFormat OCTET STRING, -- must conform xsd schemas of GML 3.0
        -- http://www.opengeospatial.org/standards/gml#downloads
    }
    civicLoc SEQUENCE
    {
        country IA5String (SIZE(0..2)), -- must conform the ISO 3166 country code
        al      [ 0] IMPLICIT IA5String OPTIONAL, -- NationalSubdivisions
    }
}

```

```

a2      [ 1] IMPLICIT IA5String OPTIONAL,      -- CountryOrParishOrDistrict
a3      [ 2] IMPLICIT IA5String OPTIONAL,      -- City
a4      [ 3] IMPLICIT IA5String OPTIONAL,      -- CityDivision
a5      [ 4] IMPLICIT IA5String OPTIONAL,      -- NeighborhoodOrBlock
a6      [ 5] IMPLICIT IA5String OPTIONAL,      -- Street
prd     [ 6] IMPLICIT IA5String OPTIONAL,      -- LeadingStreetDirection
pod     [ 7] IMPLICIT IA5String OPTIONAL,      -- TrailingStreetSuffix
sts     [ 8] IMPLICIT IA5String OPTIONAL,      -- StreetSuffix
hno     [ 9] IMPLICIT IA5String OPTIONAL,      -- HouseNumber
hns     [10] IMPLICIT IA5String OPTIONAL,      -- HouseNumberSuffix
lmk     [11] IMPLICIT IA5String OPTIONAL,      -- LandmarkVanity
loc     [12] IMPLICIT IA5String OPTIONAL,      -- AdditionalLocationInformation
flr     [13] IMPLICIT IA5String OPTIONAL,      -- Floor
nam     [14] IMPLICIT IA5String OPTIONAL,      -- ResidenceOrOfficeName
pc      [15] IMPLICIT IA5String OPTIONAL } OPTIONAL -- PostalCode
}

GpLocationInfoList ::= SEQUENCE OF GpLocationInfo

--GeoPriv has been defined, regarding the ISO/IEC 18051 respective Definition of LocationInfo
GeoPriv ::= SEQUENCE --Conforming RFC 4119
{
    gpLocationInfo      GpLocationInfoList,
    usageRules           [0] IMPLICIT UsageRules OPTIONAL,
    method               [1] IMPLICIT Method OPTIONAL,
    providedBy           [2] IMPLICIT ProvidedBy OPTIONAL }

LocationInfo ::= SEQUENCE OF GeoPriv

LocationInfoList ::= SEQUENCE OF LocationInfo

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, MaxLengthParametersContinued,
FilterThreshold, ServiceCrossRefID, DeviceCategory, GroupDeviceAttributes, NamedDeviceTypes,
ACDModels, AgentLogOnModels, AppearanceType, TransAndConfSetup, MediaServiceCapsList, LogDevServList,
LogDevEvtsList, BookmarkReached, GetLocationTrackingSessions, GetLocationTrackingCapabilities,
LocationSessionInfo, LocationServicesServList;

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

-- 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,
	locationServicesServList	[7] IMPLICIT LocationServicesServList	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

{	getCSTAFeatures	[5] IMPLICIT GetCSTAFeatures	OPTIONAL,
	getLogicalDeviceInformation	[0] IMPLICIT GetLogicalDeviceInformation	OPTIONAL,
	getPhysicalDeviceInformation	[1] IMPLICIT GetPhysicalDeviceInformation	OPTIONAL,
	getSwitchingFunctionCaps	[4] IMPLICIT GetSwitchingFunctionCapabilities	OPTIONAL,
	getSwitchingFunctionDevices	[2] IMPLICIT GetSwitchingFunctionDevices	OPTIONAL,
	switchingFunctionDevices	[3] IMPLICIT SwitchingFunctionList	OPTIONAL }

```

GetCSTAFeatures ::= BIT STRING
{
    privateDataInReq          ( 0),
    privateDataInAck          ( 1) }

```

```

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),
    requestedDeviceCategoryUserGroup (14),
    requestedDeviceCategoryOtherGroup ( 5),
    requestedDeviceCategoryNetwInterface ( 6),
    requestedDeviceCategoryPark ( 7),
    requestedDeviceCategoryRouteingDevice ( 8),
    requestedDeviceCategoryStation ( 9),
    requestedDeviceCategoryVoiceUnit (10),
    requestedDeviceCategoryGenericIV (15),
    requestedDeviceCategoryListenerIV (16),
    requestedDeviceCategoryVDTMFIV (17),
    requestedDeviceCategoryPromptIV (18),
    requestedDeviceCategoryPromptQueueIV (19),
    requestedDeviceCategoryMessageIV (20),
    requestedDeviceCategoryConference (21),
    requestedDeviceCategoryOther (11),

```

```
privateData          (12),
privateDataInAck     (13) }
```

GetSwitchingFunctionCapabilities ::= BIT STRING

```
{
    privateData          ( 0),
    mediaClassSupport    ( 1),
    numberOfChannels     ( 2),
    maxChannelBind       ( 3),
    miscMediaCallCharacteristics ( 4),
    connectionRateList   ( 5),
    delayToleranceRateList ( 6),
    pauseTime            ( 7),
    currentTime          ( 8),
    transAndConfSetupParameter ( 9),
    transAndConfSetupConsultation (10),
    transAndConfSetupHoldMake (11),
    transAndConfSetupAlternate (12),
    transAndConfSetupTwoCalls (13),
    transAndConfSetupTwoCallsConnected (14),
    miscMonitor          (15),
    correlatorDataSupported (16),
    dynamicFeatureSupported (17),
    callLinkageOptions    (18),
    acdModels            (19),
    agentLogOnModels      (20),
    agentStateModels      (21),
    privateDataVersionList (22),
    simpleThreshold       (23),
    filterThreshold       (24),
    maxDeviceHistoryEntries (25) }
```

SwitchingFunctionDevices ::= BIT STRING

```
{
    segmentID          ( 0),
    deviceListDeviceCategory ( 1),
    deviceListNamedDeviceTypes ( 2),
    deviceListDeviceAttributes ( 3),
    deviceListDeviceModelName ( 4),
    deviceListNidGroup      ( 6),
    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,
    getRegistrations         [ 8] IMPLICIT GetRegistrations         OPTIONAL,
    registrationInfo         [ 9] IMPLICIT RegistrationInfo         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

GetRegistrations ::= BIT STRING
{
    requestedReqTypes      ( 0),
    privateData            ( 1),
    privateDataInAck        ( 2),
    segmentedResponseProvided ( 3) }

RegistrationInfo ::= BIT STRING
{
    segmentID              ( 0),
    privateData             ( 1) }

MonitoringServList ::= SEQUENCE
{
    changeMonitorFilter      [ 0] IMPLICIT ChangeMonitorFilter    OPTIONAL,
    monitorStart             [ 1] IMPLICIT MonitorStart           OPTIONAL,
    monitorStop              [ 2] IMPLICIT MonitorStop            OPTIONAL,
    getMonitors              [ 3] IMPLICIT GetMonitors            OPTIONAL,
    monitorInfo              [ 4] IMPLICIT MonitorInfo            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
    monitorMediaClassChat     (15),    -- optional parameters
    monitorMediaClassEmail    (16),    -- optional parameters
    monitorMediaClassMessage  (17),    -- optional parameters
    monitorMediaClassIM       (18),    -- optional parameters
    monitorMediaClassSMS      (19),    -- optional parameters
    monitorMediaClassMMS      (20),    -- optional parameters
    monitorExistingCallsInAck (10),    -- optional parameters
    privateDataInAck          (11),    -- optional parameters
    callIDOnly                (12),    -- miscellaneous characteristics
    swDomainDefaultForMonitorTypeIsDevice (13),    -- miscellaneous characteristics
    acceptsMonitorStartWhileOOS (14) }    -- miscellaneous characteristics

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

```

```

GetMonitors ::= BIT STRING
{
    privateData                ( 0),    -- optional parameters
    privateDataInAck           ( 1),    -- optional parameters
    segmentedResponseProvided  ( 2) }   -- optional parameters

MonitorInfo ::= BIT STRING
{
    segmentID                  ( 0),    -- optional parameters
    privateData                ( 1) }   -- optional parameters

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
    subjectOfCallInAck           (14),    -- optional parameters
    messageInfoInAck             (15),    -- optional parameters
    languagePreferencesInAck     (16),    -- optional parameters
    deviceHistoryInAck           (17),    -- optional parameters
    privateDataInAck             (11),    -- optional parameters
    callIDOnly                   (12),    -- miscellaneous characteristics
    reportsWithSnapsCallData     (13),    -- optional parameters
    locationInfoList             (18) }   -- 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
    locationInfoList             ( 8) }   -- 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
    endpointDeviceID              (11),    -- 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,
    makeConnection            [25] IMPLICIT MakeConnection     OPTIONAL,
    makePredictiveCall        [18] IMPLICIT MakePredictiveCall OPTIONAL,
    parkCall                  [19] IMPLICIT ParkCall           OPTIONAL,
    reconnectCall             [20] IMPLICIT ReconnectCall      OPTIONAL,
    retrieveCall              [21] IMPLICIT RetrieveCall       OPTIONAL,
    sendMessage               [26] IMPLICIT SendMessage        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
    subjectOfCall           (11),    -- optional parameters
    languagePreferences     (12),    -- 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
reason                    (12) }   -- optional parameters

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
    reason                  (12),    -- 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
    callCharacteristicsLowPriorityCall (29), -- optional parameters
    callCharacteristicsPriorityCall ( 7), -- optional parameters
    callCharacteristicsHighPriorityCall (30), -- optional parameters
    callCharacteristicsMaintenanceCall ( 8), -- optional parameters
    callCharacteristicsDirectAgent ( 9), -- optional parameters
    callCharacteristicsAssistCall (10), -- optional parameters
    callCharacteristicsVoiceUnitCall (11), -- optional parameters
    callCharacteristicsPrivateCall (31), -- optional parameters
    callCharacteristicsPersonalCall (32), -- optional parameters
    callCharacteristicsSensitiveCall (33), -- optional parameters
    callCharacteristicsConfidentialCall (34), -- optional parameters
    callCharacteristicsEncryptedCall (35), -- optional parameters
    mediaCallCharacteristics (12),    -- optional parameters
    callingConnectionInfo   (13),    -- optional parameters
    flowDirectionTransmit   (14),    -- optional parameters
    flowDirectionReceive    (15),    -- optional parameters

```