Standard ECMA-269

3rd Edition - December 1998

ECMA

Standardizing Information and Communication Systems

Services for Computer Supported Telecommunications Applications (CSTA) Phase III

Volume 1

Standard ECMA-269

3rd Edition - December 1998

ECMA

Standardizing Information and Communication Systems

Services for Computer Supported Telecommunications Applications (CSTA) Phase III

Volume 1

Phone: +41 22 849.60.00 - **Fax:** +41 22 849.60.01 - **URL:**http://www.ecma.ch - **Internet:** helpdesk@ecma.ch

Brief History

This Standard ECMA-269 defines Phase III of Services for Computer Supported Telecommunications Applications (CSTA) for OSI Layer 7 communication between a computing network and a telecommunications network. This Standard is part of a Suite of Standards and Technical Reports for Phase III of CSTA. All of the Standards and Technical Reports in the Suite are based on practical experience of ECMA member companies and each one represents a pragmatic and widely-based consensus.

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

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

- New categories of services and events such as capabilities exchange, charging, media attachment services, call data recording (CDR), etc.
- Additional services and events for call and device control.
- Enhancement to existing services and events.
- Organization of services and events to reflect a grouping based on function (call control, device control, etc.).
- Use of a consistent template for services and events that includes initial/final connection state, connection state transitions, event monitoring sequences, etc.

The First Edition of Standard ECMA-269 was published in December 1997 and the Second Edition was published in June 1998.

This edition completes the planned Services for CSTA Phase III by extending the Second Edition in the following areas: ACD and ACD Agent Modeling, Call Associated Features, Call Detail Recording services, Capability Exchange services, Data Collection services, I/O Services, Logical Device Feature services, Physical Device Feature services, Media Attachment services, Maintenance events, Vendor Specific Extensions, and Voice services.

This ECMA Standard is contributed to ISO/IEC JTC1 under the terms of the fast-track procedure, for adoption as an ISO/IEC International Standard.

Adopted as 3rd Edition of Standard ECMA-269 by the General Assembly of December 1998.

Table of Contents

1 Scope

Services and Events Reports supported by Computer-Supported Telecommunications Applications, Phase III (CSTA) are defined in this Standard.

This Standard is focused on providing application service interfaces to a Switching Function, Computing Function and a Special Resource Function. A CSTA application interface is disassociated from the various user-network interfaces and network-network interfaces CSTA applications may serve, observe or manipulate. Because CSTA operates with existing telecommunications interfaces indirectly, it operates generically, so that differences among various existing interfaces are hidden from CSTA applications. Support of user-to-network interfaces is outside the scope of CSTA.

Although most terminal equipment (TE) are suitable for use with CSTA there will be instances of TE that will not be suitable in certain circumstances. Examples are:

- FAX terminals and modems that are unable to adjust their transmission modes to prevent carrier conflict when both parties are alerted via CSTA during call establishment;
- Functional terminals that perform telecommunication functions outside the control of the Switching Function.

Services defined in this Standard allow functional integration between a computing network and a telecommunications network. Computing platforms (i.e., Application Programming Interfaces - APIs) that support such functionally-integrated applications are outside the scope of this Standard.

Communication between the computing and switching (i.e., telecommunications) networks may take place via intervening networks ranging from simple point-to-point connections to local- or wide-area telecommunications networks.

This Standard is part of a suite of CSTA Standards and Technical Reports that provide a comprehensive description of the architectural and practical issues involved in applying, implementing, and utilizing CSTA-based CTI applications.

2 Conformance

This Clause specifies the conformance requirements for a Switching Function, Special Resource Function, and a Computing Function.

Conformance requirements specify the parts of this Standard that a CSTA conformant implementation shall support.

This Standard specifies an operational model (Clause 6, “CSTA Operational Model” and Clause 9, “Generic Service Requirements”) that defines a collection of objects (e.g. domains and sub-domains, logical and physical elements, calls) and the relationships between these objects.

The behaviours of CSTA-conformant services, features, and event reports are determined by this model.

2.1 Switching Function

In order to conform to this Standard a switching function shall support the following as a minimum:

1. the requirements pertaining to CSTA features as specified in Clause 6, “CSTA Operational Model”.
2. the requirements as specified in Clause 9, “Generic Service Requirements”.
3. the Get Switching Function Capability service.
4. at least one of the profiles as specified in 2.1.3, “CSTA Profiles”.

2.1.1 Conformant Services

In order to conform to a specific CSTA *service* an implementation shall support the following as a minimum:

1. the requirements of the service as specified by its service description, service request parameters, service response parameters, and operational model including connection state transitions, monitoring event sequences, and functional requirements.

2. the requirements associated with each parameter used in the service as specified by its parameter description, format, and functional requirements in Clause 12, “Parameter Types”.
3. all of the events that are associated with its service completion criteria, as documented in its event monitoring tables.
4. service requests that contain a device identifier parameter, an implementation shall support, at a minimum, the Diallable Digits format as specified in 10.1.1, “Diallable Digits”.

2.1.2 Conformant Events

In order to conform to a specific CSTA *event* an implementation shall support the following as a minimum:

1. the requirements of the event as specified by its event description, event parameters, event causes, and functional requirements.
2. the requirements associated with each parameter used in the event as specified by its parameter description, format, and functional requirements in Clause 12, “Parameter Types”.
3. events that contain a device identifier parameter, an implementation shall support, at a minimum, the Switching Function Representation format as specified in 10.1.2, “Switching Function Representation”.

2.1.3 CSTA Profiles

Some CSTA services and events are grouped together as profiles.

2.1.3.1 Basic Telephony Profile

This profile includes the following:

1. CSTA Services: Answer Call, Clear Connection, Make Call, Monitor Start (with the monitorType of device-type), and Monitor Stop.
2. CSTA Events: Connection Cleared, Delivered, Established, Failed, Network Reached, Originated, and Service Initiated.

Other CSTA services and events may be provided in any combination in addition to this set.

2.1.3.2 Routeing Profile

If the switching function supports Routeing Services as specified in Clause 20, “Routeing Services”, it shall support a minimum set of Routeing Services that includes: Route Request, Route Select, and Route End (from the switching function only).

Other Routeing services may be provided in any combination in addition to this set.

If a switching function supports the routeing for digital data calls, then the Route Register and CSTA Route Register Cancel shall also be included in the minimum set.

2.1.4 Support of Service Requests And Manual Mode

A conformant switching function may support a given service defined in this Standard through the CSTA service boundary but is not required to support the equivalent service in a manual mode.

A conformant switching function may support a feature associated with an equivalent CSTA service defined in this Standard through manual mode but is not required to support the equivalent service through the service boundary.

2.2 Special Resource Function Conformance

In order to conform to this Standard a special resource function shall support the following as a minimum:

1. the requirements pertaining to CSTA features as specified in Clause 6, “CSTA Operational Model”.
2. the requirements as specified in Clause 9, “Generic Service Requirements”.
3. for a supported service, the special resource function shall not reject as unsupported all of the events specified in the monitoring event sequences associated with the service.
4. the atomic service request acknowledgment model as specified in 9.2, “Service Response (Acknowledgements)”.

2.2.1 Conformant Services

In order to conform to a specific CSTA *service* an implementation shall support the following as a minimum:

1. the requirements of the service as specified by its service description, service request parameters, service response parameters, and operational model including connection state transitions, monitoring event sequences, and functional requirements.
2. the requirements associated with each parameter used in the service as specified by its parameter description, format, and functional requirements in Clause 12, “Parameter Types”.
3. all of the events that are associated with its service completion criteria, as documented in its event monitoring tables.

2.2.2 Conformant Events

In order to conform to a specific CSTA *event* an implementation shall support the following as a minimum:

1. the requirements of the event as specified by its event description, event parameters, event causes, and functional requirements.
2. the requirements associated with each parameter used in the event as specified by its parameter description, format, and functional requirements in Clause 12, “Parameter Types”.

2.2.3 Support of Service Requests And Manual Mode

A conformant special resource function may support a given service defined in this Standard through the CSTA service boundary but is not required to support the equivalent service in a manual mode.

A conformant special resource function may support a feature associated with an equivalent CSTA service defined in this Standard through manual mode but is not required to support the equivalent service through the service boundary.

2.3 Computing Function Conformance

In order to conform to this Standard a computing function shall support the following as a minimum:

1. the requirements pertaining to CSTA features as specified in Clause 6, “CSTA Operational Model”.
2. the requirements as specified in Clause 9, “Generic Service Requirements”.
3. for a supported service, the computing function shall not reject as unsupported all of the events specified in the monitoring event sequences associated with the service.
4. the “Single Physical and Logical Element” and “Logical Element Only” device configurations as specified in 6.1.3.3, “Device Configurations”.
5. for service requests, the Diallable Digits format of Device Identifiers as specified in 10.1.1, “Diallable Digits”.
6. for events, all formats of Device Identifiers as specified in 10.1, “Device Identifier Formats”.
7. the “No Appearance Addressability” and the “Individual Appearance Addressability” of referencing device elements as specified in 6.1.7, “Referencing Devices, Elements, Appearances and Device Configurations”.
8. both types of service request acknowledgment models (e.g., Atomic and Multi-Step) as specified in 9.2, “Service Response (Acknowledgements)”.
9. all failure models as specified in 6.8.2, “Connection Failure”.
10. both switching function options of handling unsupported parameters in service requests as specified in the capability exchange services.
11. both the fixed and local view of the primaryOldCall and the secondaryOldCall parameters in the Conferenced and the Transferred events.
12. all bi-directional services for which it registered, whether explicitly (i.e., via a service registration service such as System Status Register) or implicitly (i.e., the switching function does not support registration but does

support (as indicated through the capabilities exchange services) a particular bi-directional service and therefore may issue a service request to the computing function).

3 References

The references used in this Standard are defined in the following sections.

3.1 ECMA References

ECMA-143	Private Integrated Services Network (PISN) - Circuit mode bearer services - Inter-Exchange Signalling Procedures and Protocol (QSIG-BC), 3rd edition (June 1997)
ECMA-155	Private Integrated Services Network (PISN) - Addressing, 2nd edition (June 1997)
ECMA TR/72	Glossary of Definitions and Terminology for Computer Supported Telecommunications Applications (CSTA) Phase III, 2nd edition (December 1998)

3.2 ISO References

ISO 8649:1988	Information Processing Systems - Open Systems Interconnection - Service definition for the Association Control Service Element.

3.3 ITU-T References

E.131	Subscriber Control Procedures for Supplementary Telephone Services (7) (1988)
E.160	Definitions Relating to National and International Numbering Plans (6) (1993)
E.161	Arrangement of Digits, Letters and Symbols on Telephones and Other Devices that can be used for Gaining Access to a Telephone Network (7) (1993)
E.164	The International Public Telecommunication Numbering Plan (7) (1997)
Q.931	Digital Subscriber Signalling System No. 1 (DSS 1) - ISDN User-Network Interface layer 3 Specification for Basic Call Control (9) (1993)

4 Definitions and Abbreviations

The definitions and abbreviations used in this Standard are defined in *Glossary of Definitions and Terminology for Computer Supported Telecommunications Applications (CSTA) Phase III,* ECMA TR/72.

5 Functional Architecture

The objective of CSTA Architecture is to define the inter working mechanisms among Computing, Switching and Special Resource Functions independently from their physical implementations.

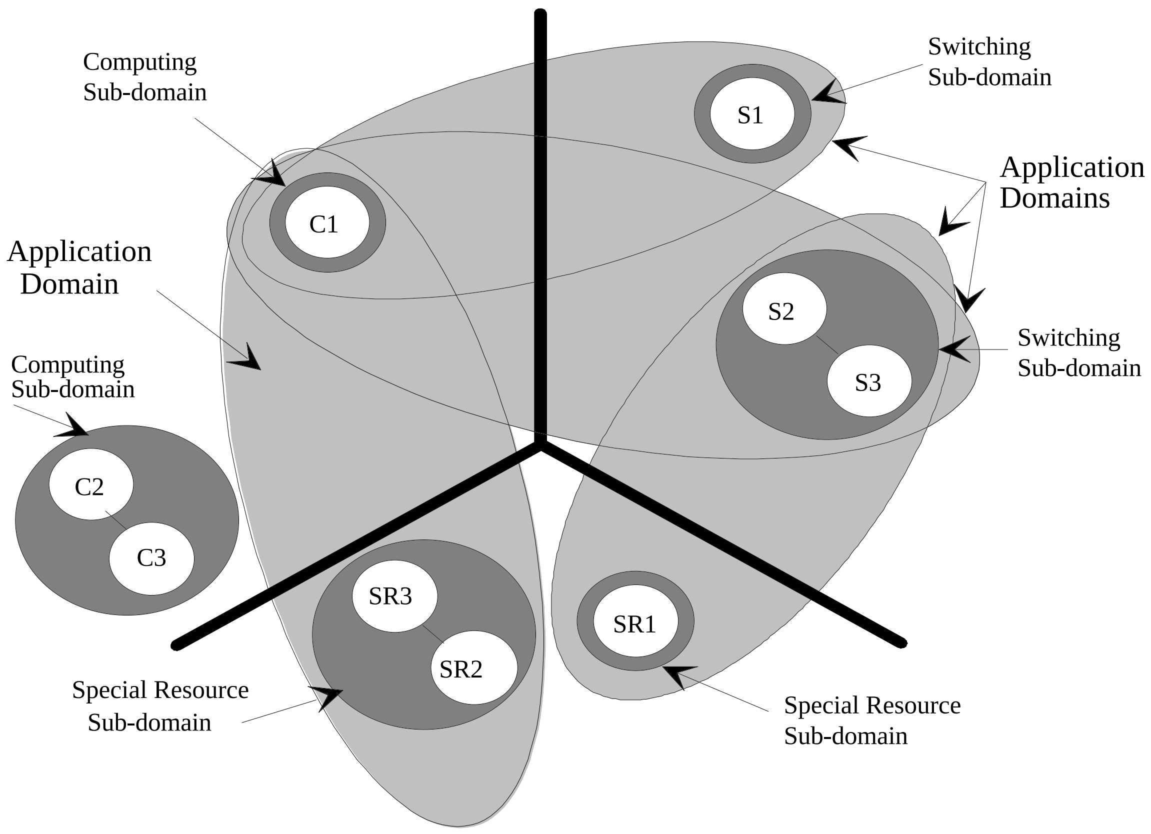
The concepts of: distribution of Computing, Switching and Special Resource Functions, CSTA Service, client server model, and CSTA objects as abstracted at a CSTA Service Boundary, are specified in another part of the CSTA Phase III Suite.

6 CSTA Operational Model

The operational model considered for CSTA is summarized in this clause.

The set of accessible Computing, Switching and Special Resource Functions from which an application might receive service defines a CSTA domain. An example of a CSTA domain is shown in the next figure. The CSTA domain contains switching, computing and special resource domains that are divided in the figure by the heavy lines. The special resource, switching and computing domains comprise Computing Functions (C1, C2 and C3), Switching Functions (S1, S2 and S3), and Special Resource Functions (SR1, SR2 and SR3). Each function can provide to a CSTA application, a view of the domain in which the function resides. Each such view defines a sub-domain. If one or more functions provides an identical view, then these functions are part of the same sub-domain. CSTA applications encompass at least two different sub-domains, and are represented in the next figure as application domains.

Figure 6-1 Domains and Sub-Domains



Note that a function may provide a view to an application that includes not only the objects within its sub-domain, but also the objects it can view in another (presumably related) sub-domain. For example (in Figure 6-1), a computing domain {C1} may receive a view of a switching sub-domain from a switching domain {S2+S3}. That switching sub-domain may receive a view of a special resource sub-domain from a special resource domain {SR1}, and relay that view, in addition to the view of its sub-domain, to the Computing Function. This relay may preserve two views of separate switching and special resource sub-domains, or it may provide a combined view of a switching/special-resource sub-domain. As shown in the figure, {C1} also may have its own, direct view of a special resource sub-domain {SR2+SR3}. Finally, {C2+C3} represent a computing domain that is potentially, but not yet, involved in CSTA transactions with other sub-domains because an association has not yet been established between any other sub-domain and {C2+C3}.

6.1 Switching Sub-Domain Model

The tools needed to provide an abstract view of the Switching Function are defined by the switching sub-domain model. This model allows an application to conceptualize the Switching Function’s operation. To provide this abstract view, CSTA defines several CSTA switching sub-domain model Objects that can be observed and acted upon by the Switching Function on behalf of the Computing Function. Those objects include CSTA Devices, Calls, and Connections.

The concepts discussed in this section have also been introduced in another part of the CSTA Phase III Suite. Refer to this reference for an introduction to these concepts.

6.1.1 Switching Sub-Domain Name

The switching function is identified by a unique name within the switching domain known at association time. This name may be used by the computing function to identify that different CSTA applications resulting from different associations are operating in the same switching sub-domain.

6.1.2 Application Working Domain

The application working domain is the subset of devices (and the calls and connections associated with those devices) inside a switching sub-domain that are controllable and/or monitorable over a CSTA Service Boundary. This subset is known at association time. The scope of this working domain may result from considerations like the application's design, licensing policy, security constraints, etc., and is under administration of the switching sub-domain. Different CSTA applications operating in the same switching sub-domain can have different working domains or share totally or partially the same working domain.

6.1.3 Device

CSTA enables manipulation and observation of devices that allow users to access telecommunications services.

NOTE

It is not claimed that this Standard alone supports ISDN (or any other) devices because, for example, of the additional information required to support such devices in PISNs. CSTA only provides a facility for passing ISDN (or other) specific information to allow, for example, a selection among ISDN devices sharing the same directory number (bearer capability, subaddress, etc.). Another example, that applies generally to telecommunications networks (including ISDN and OSI), is specifying the originator for a call that is established via CSTA. With the current signalling support, each party in a call can act only as a called party because the "network" is acting to originate the call. This situation has implications for both the network-to-terminal signalling and any application-level signalling that is significant to the calling party (e.g., issuing A_Associate).

Devices that are visible or controllable via CSTA are known as *CSTA Devices.*

CSTA Devices can be either physical devices (such as buttons, lines, trunks, and stations) or logical devices (such as groups of devices, pilot numbers, and automatic call distribution groups). CSTA Devices have attributes that allow CSTA to monitor and manipulate them. The attributes of any CSTA Device shall be:

1. **Device Type** - differing types of CSTA Device can be used for various purposes and can be manipulated and observed differently within CSTA. CSTA Device Types are listed and defined in 6.1.3.4, "Device Categories".
2. **Media Characteristics** - CSTA devices have distinct capabilities and characteristics defined by their media features. CSTA represents these characteristics by the following attributes.
   - **Media Class:** A CSTA Device shall belong to at least one and may belong to more than one media class. The Media Class can be used in Call Control services to help select a device for a call or it can be used in call control events to report the media class associated with the call. The following media classes are defined in CSTA:
     - **Audio** - 3.1 KHz audio. Devices in this class are used to make audio calls excluding speech calls. It includes G3 FAX and facsimile machines.
     - **Data** - Devices in this class are used to make digital data calls (both circuit switched and packet switched). This class includes digital computer interfaces and G4 facsimile machines.
     - **Image** - Devices in this class are used to make digital data calls involving imaging, or high-speed, circuit-switched data in general. This class includes digital video telephones and CODECs.
     - **Voice** - Devices in this class are used to make speech calls. This class includes standard telephones.
     - **Other** - A class comprising devices not in the Data, Image, Audio or Voice classes.
   - **Media Stream Information**: The media stream associated with a CSTA Device has attributes such as **Connection Rate**, **Bit Rate**, and **Delay Tolerance**. This information can be used in CSTA services to help select the media stream information for a call or to report the media stream information associated with an existing call.

- **Protocol Specific Information**: Many protocols provide additional information beyond what is standardized in CSTA to help distinguish devices. CSTA provides a mechanism where protocol specific information can be passed in CSTA messages to help select a specific device for a call or to provide additional information about the protocol specific information associated with the call. This information consists of:
    - The type of call control information elements (ISDN, for example).
    - A character string that contains the protocol specific information elements. For example, in ISDN, the information may include Bearer Capability, Subaddress (for both calling and called devices), High Layer compatibility, and Low Layer compatibility as defined in IS 11572: 1993.

Refer to 12.2.15, “MediaCallCharacteristics”, for a description of the Media Characteristics that are used in Call Control services to select devices for a call and in Call Control events to report the media Characteristics associated with the devices involved with the call.

3. **CSTA Device Identifier** - Each device that can be observed and/or manipulated shall be referenced across the CSTA Service Boundary. To accomplish this, each device shall be identified using a Device Identifier.
    - Throughout this standard, the term *Device Identifier* shall always mean CSTA *Device Identifier.*
    - Device Identifiers may be static or, only when used in the context of a connection identifier, dynamically-assigned.
    - A static Device Identifier shall be stable over time. It shall remain constant and unique between calls, associations and within both the switching and computing functions. An example of a static Device Identifier is an ITU-T E.164 Directory Number.
    - It may be useful for the Switching Function to convert a Device Identifier to another static form for use in service interactions. An example, it might be useful to transform a Public Directory Number into a Private Directory Number. This transformation allows service interactions to be independent of the identification mechanism and allows reduction in the amount of data exchanged. This shortened form of Device Identifier is known as a CSTA Short Form Device Identifier.
    - A static Device Identifier may be used in conjunction with “MediaCallCharacteristics”, as specified in 12.2.15, “MediaCallCharacteristics”, on page 101, in order to distinguish among CSTA Devices that share a Device Identifier.
    - A dynamically-assigned Device Identifier is temporary (lasting for the duration of a call) and may be created at any appropriate time. Once a CSTA Device has been included in a call, it may be desirable to continue to refer to the particular instance of the CSTA Device associated with this call for manipulation or tracking. A static Device Identifier may not always be sufficient because it may not be available or because it is too long and cumbersome for efficient use. In these cases the Switching Function can dynamically assign a Device Identifier as a device reference or handle for the duration of the call. Management of the dynamically-assigned Device Identifier is discussed in 6.1.8.
    - The **Device Identifier Status** indicates if an actual Device Identifier is being provided in a parameter or the reason why it is not being provided. The set of possible values for the Device Identifier Status is:
        - *Provided* - A Device Identifier is present.
        - *Not Known* - Indicates that the switching function cannot provide the Device Identifier but knows that the device exists.
        - *Not Required* - Indicates that the device is not relevant in this case.
        - *Not Specified* - Indicates that the device cannot be specified.
    - The parameter type associated with a particular Device Identifier determines how it is interpreted, restrictions on its use, and the Device Identifier Statuses that are applicable. These parameter types (AssociatedCalledDeviceID, AssociatedCallingDeviceID, CallingDeviceID, CalledDeviceID, DeviceID,

RedirectionDeviceID, and SubjectDeviceID) are specified in 12.3, “Identifier Parameter Types”, on page 107.

- The format of a Device Identifier is specified in Clause 10, “CSTA Device Identifier Formats”.

4. **Device State** - A CSTA Device itself does not have a state or status directly associated with it. The elements, components and calls associated with the CSTA Device do have states and statuses associated with them. The following is the list of these states and statuses associated with a CSTA Device:

   - A connection state is the state of a CSTA Device’s logical element’s connection into a call. This state is associated with Call Control features/services. For more information on connection states, refer to Clause 6.1.5, “Connection”, beginning on page 31.
   - The status of the physical components associated with a physical element of a CSTA Device. (e.g., the hookswitch status). For more information, refer to Clause 21, “Physical Device Features”, beginning on page 383.
   - The status of the logical device features/services associated with a logical element of a CSTA Device. (e.g., the forwarding and do not disturb status). For more information, refer to Clause 22, “Logical Device Features”, beginning on page 431.

5. **Device Elements** - A CSTA Device represents various types of telephony endpoints in a switching sub-domain and allows access to telephony services. A CSTA Device can range from a single endpoint (e.g., station) to a set of associated endpoints that form a group. Each CSTA Device is represented by its attributes (e.g., identifier, state(s), type) as well as its features/services. These attributes/features/services are grouped into two categories which are referred to as *device elements*. A device element encompasses the control and observation of a specific set of CSTA Device attributes/features/services. The device elements are: *physical element* and *logical element*.

The logical element of a CSTA Device encompasses the set of attributes/features/services (e.g., Make Call, Set Forward) that have any association with the control and observation of a call at a CSTA Device (i.e., connection). The physical element of a CSTA Device encompasses the set of attributes/feature/services that have any association with physical components of the CSTA Device that would potentially make up the user interface of the device.

All addressable (i.e., has a single identifier) CSTA Devices consist of one of the following device element combinations.

Figure 6-2 Logical Element Only

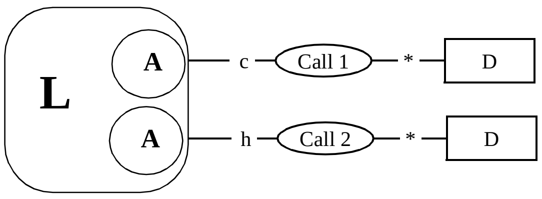


Figure 6-3 Physical Element Only

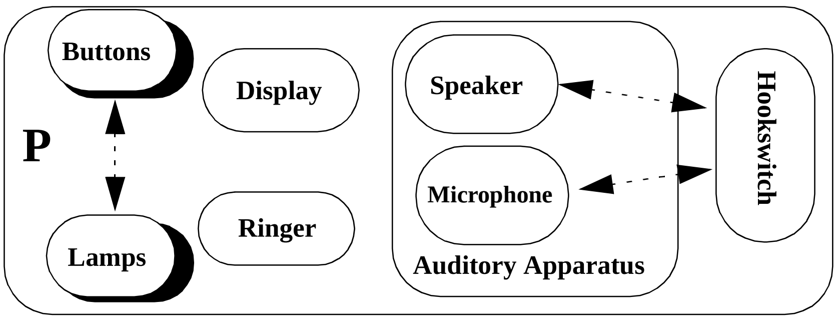
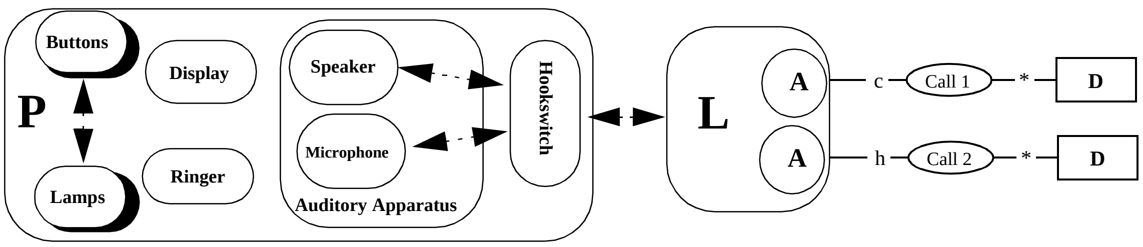


Figure 6-4 A Logical and Physical Element



D	represents another device
P	represents the identifiers for the physical elements
L	represents the identifiers for the logical elements
A	represents an appearance of a logical element

◄- - - - ►

indicates that there is an interaction and/or association between the elements or components of an element.

For example, a “plain old telephone set” (POTS) consists of a logical and physical element. A computing function learns about the devices, their elements, and their associated attributes\features\services in a switching sub-domain by using the capabilities exchange services (refer to 13.1 beginning on page 118). The following sections describes the device elements and the device attributes/features/services that are associated with the elements in detail.

6.1.3.1 Physical Element

The physical element represents the attributes of the physical components and their associated features/services that make up the user interface of a device (e.g., the components of a telephone set). A physical component at a CSTA Device can be a piece of hardware or a virtual (e.g., software) representation of a piece of hardware. For example, a set of buttons on a device (i.e., physical components of the device) could be comprised of a piece of hardware with 12 buttons and a switching function software representation of 12 more buttons. As a result, the device has a set of 24 buttons associated with its physical element.

The combinations of physical components associated with the physical element are switching function specific. The following features/services are controlled and observed through the physical element:

- The Physical Device features/services, such as Button Press, Get/Set Hookswitch Status, and Get/Set Speaker Volume.
- The I/O Services such as Start Data Path and Send Data (when applied to the physical element of a device).

Note that these features/services also include the associated events and monitoring of these events.

The physical element of a device is observed and/or controlled within the switching function through an assigned Device Identifier.

Note that if the device is a combination of logical and physical elements, the assigned Device Identifier is the same for both elements.

In order for the physical components to interact or be associated with calls at the device, the device shall have a logical device element (for more details, see 6.1.3.2, "Logical Element") and/or some association(s) with a logical element(s) from another device(s) (for more details, see 6.1.3.3, "Device Configurations"). The physical components interact with calls through these logical device element(s) but it is device and switching function specific as to how these components actually interact and are associated with the calls.

The following sections describe the physical components that can be associated with the physical element of the device. All physical components shall be controlled and observed in conjunction with the physical element (i.e., associated with the physical element's Device Identifier).

6.1.3.1.1 Auditory Apparatus

An auditory apparatus is a component which is used to convert electronic signals into voice/speech (i.e., a speaker) and/or convert voice/speech into electronic signals (i.e., a microphone) but at a minimum shall have either a speaker or a microphone. A physical element can have several auditory apparatuses associated with it. Each auditory apparatus can be used independent of each other. An auditory apparatus has several attributes which can be controlled and observed by a computing function. The following are those attributes:

1. *Auditory Apparatus type* - There are several types of auditory apparatuses. Each type representing a different physical configuration and/or function. The following is the list of auditory apparatus types:
    a. *Handset* - An auditory apparatus that is held in a person's hand and contains a microphone and speaker.
    b. *Headset* - An auditory apparatus that is worn on a person's head and contains a microphone and speaker.
    c. *Speakerphone* - An auditory apparatus that does not require a person's body to come into contact with the apparatus and contains a microphone and speaker.
    d. *Speaker-only phone* - A Speakerphone without a microphone.
    e. *Microphone-only*- A type that provides only a microphone.
    f. *Other* - An auditory apparatus that is unique to the given switching function.
2. *Auditory Apparatus Identifier* - Each auditory apparatus that can be observed and/or controlled within the switching function is referenced using an assigned identifier. The auditory apparatus identifiers associated with a given physical element's Device Identifier are unique.This identifier is used to control and observe all auditory apparatus attributes except the hookswitch which is associated with the apparatus.
3. *Microphone* - The auditory apparatus may or may not have a microphone. If a microphone is present at the auditory apparatus, then there are two features of the microphone that may or may not be controlled (i.e., settable) and observed (i.e., readable).
    a. *Gain* - This is the level at which the microphone is generating the electronic signal. For more details, refer to the definition of the microphoneGainValue parameter in associated services and events.
    b. *Mute* - This is the capability to temporarily disable the microphone. For more details, refer to the definition of the microphoneMute parameter in associated services and events.

    Both of these features are controlled and observed using the auditory apparatus identifier.
4. *Speaker* - The auditory apparatus may or may not have a speaker. If a speaker is present at the auditory apparatus, then there are two features of the speaker that may or may not be controlled (i.e., settable) and observed (i.e., readable).
    a. *Volume* - This is the level at which the speaker is boosting the electronic signal when generating the associated voice sound waves. For more details, refer to the definition of the speakerVolumeValue parameter in associated services and events.

b. *Mute* - This is the capability to temporarily disable the speaker. For more details, refer to the definition of the speakerMute parameter in associated services and events.

Both of these features are control and observed using the auditory apparatus identifier.

5. *Hookswitch association* - This identifies the particular hookswitch that is used to activate (i.e., put into use) and deactivate (i.e., remove from use) the auditory apparatus. It also indicates, if the particular hookswitch can be controlled (i.e., settable) and observed. For more details on hookswitches, refer to associated services and events.

6.1.3.1.2 Hookswitch

A hookswitch is a component which is used to activate (i.e., put into use or off-hook) or deactivate (i.e., remove from use or on-hook) an auditory apparatus(es). When a hookswitch is off-hook, it enables the auditory apparatus(es) to transmit and receive the electronic signals associated with sound, and when it is on-hook, this capability is disabled. A physical element can have several hookswitches associated with it. Each hookswitch can be used independently of each other. A hookswitch has one attribute which can be controlled and observed by a computing function. This attribute is the hookswitch identifier. This identifier is assigned by the switching function. The hookswitch identifiers associated with a given physical element's Device Identifier are unique. This identifier is used to control and observe the status of the particular hookswitch (i.e., on-hook, off-hook).

6.1.3.1.3 Button

A button is a component which executes a specific feature/service that is assigned to it. The most common implementation of this component, is a piece of hardware that is pressed and released, thereby executing the feature/service assigned to it (e.g., each of the number buttons on a station). However, an implementation can use any component that can produce a similar behaviour. A button can also have the capability of toggling between two settings of a feature/service (e.g., enabling and disabling Do Not Disturb, that is you press the button once, it enables the Do Not Disturb feature/service, and press again, it disables the feature/service). The button can also have the capability of looping through a series of features/services. Almost any feature/service or set of features/services can be assigned to a button, but generally a switching function makes visible buttons only with features/services that are not available through the components/attributes/features/services defined in this Standard. A button can also be used to represent another physical component (e.g., a hookswitch). A physical element can have many buttons associated with it. Each button can be used independently of each other. A button has the following attributes which can be controlled and observed by the computing function:

1. *Button Identifier* - Each button that can be observed and/or controlled within the switching function is referenced using an assigned identifier. The button identifiers associated with a given physical element's Device Identifier are unique. This identifier is used to control and observe all other button attributes.

2. *Button Label* - This is a label by which people interacting with the physical device refer to a given button. This label is a character string which is retained by the switching function. This attribute can also be changed (if supported) by the computing function. The meaning of a Button Label is specific to the users of a particular device and changing it does not change the function of the button.

3. *Button Function* - This is a feature which can be assigned by the switching function to describe the function associated with a given button. The switching function may reassign the functionality of a button and change this attribute as required (in response to other button presses, for example) but it may not be changed directly by the computing function.

4. *Button Associated Number* - This is a diallable digits format Device Identifier which is associated with the feature/service assigned to the button. This Device Identifier is used by the feature/service when it is executed by the button being pressed. This attribute is optional and only applies to buttons that have some form of associated number. This Device Identifier is initially assigned by the switching function and can be changed (if supported) at any time by either the switching or computing function.

5. *Button Press Indicator* - This indicates if the button can be pressed via the Button Press service.

6.1.3.1.4 Lamp

A lamp is a component that represents (i.e., indicates), for example, the status of, for example, a feature/service, physical component, logical device element or other CSTA device. The most common implementation of this component is a piece of hardware that emits light. However, an implementation can use any component that can produce a similar behaviour (e.g,. an icon presented on a display). A physical element can have many lamps associated with it. Each lamp can be used independently of each other. A lamp has several attributes which can be controlled and observed by the computing function. The following are those attributes:

1. *Lamp Identifier* - Each lamp that can be observed and/or controlled within the switching function is referenced using an assigned identifier. The lamp identifiers associated with a given physical element's Device Identifier are unique.This identifier is used to control and observe all other lamp attributes.
2. *Lamp Label* - This is a character string which is assigned by the switching function to describe the feature or service's status associated with this lamp. This attribute cannot be changed by the computing function.The meaning of the Lamp Label attribute is switching function specific.
3. *Lamp Mode* - This is the output of the lamp which is used to indicate the status of the (feature/service) or physical component. The output values are represented by the various ways light can be produced from a lamp. This output can be changed at any time by either the switching or computing function. For more details, refer to definition of the lampMode parameter.
4. *Lamp Brightness* - This attribute indicates the visible brightness of the lamp when it is on. This attribute can be changed by the switching function or by the computing function. For more details, refer to the definition of the lampBrightness parameter.
5. *Lamp Color* - This attribute is an additional characteristic of the lamp which helps distinguish it from other lamps. This attribute can only be changed by the switching function. For more details, refer to the definition of the lampColor parameter.
6. *Button Association* - This identifies a button that is associated with the lamp. The lamp can be used to represent either the status of the feature/service associated with the button or to represent the status of the toggle sequence.

6.1.3.1.5 Ringer

A *ringer* is a component associated with the physical element that provides indication that a device is being rung. There may be one or more ringers associated with a device.

A ringer has attributes that can be controlled and observed by a computing function. The following is a list of those attributes:

1. *Ringer Identifier* - Each ringer that can be observed and/or controlled within the switching function is referenced using an assigned identifier. The ringer identifiers associated with a given physical element's Device Identifier are unique. This identifier is used to control and observe all ringer attributes associated with the device.
2. *Ring Mode* - This attribute describes if the ringer is engaged in a ringing cycle. It will remain *ringing* for the entire ringing cycle (e.g. across consecutive instances of a ringing pattern). Typically only one ringer on a physical element can be rung at one time.
3. *Ring Count* - This attribute describes the number of ring cycles (instances of ring pattern) that the ringer has completed. This attribute is set to 0 immediately before the first ring cycle starts. Note that this is used to query the most recent ring count even after ringing has ceased.
4. *Ring Pattern* - This attribute describes the type of Ring Pattern associated with a ringer. Each individual Ring Pattern cycle may consist of zero or more periods of audible ringing followed by a silent phase. The Ring pattern may be used to help audibly distinguish the types of calls at a device or to uniquely identify a ringer. The meaning of the Ring Pattern is switching function specific.
5. *Ring Volume* - This is the level at which the ringer is set to ring. This information is associated with the ringer until it is reset by the switching function or until it is changed via the Set Ringer Status service.

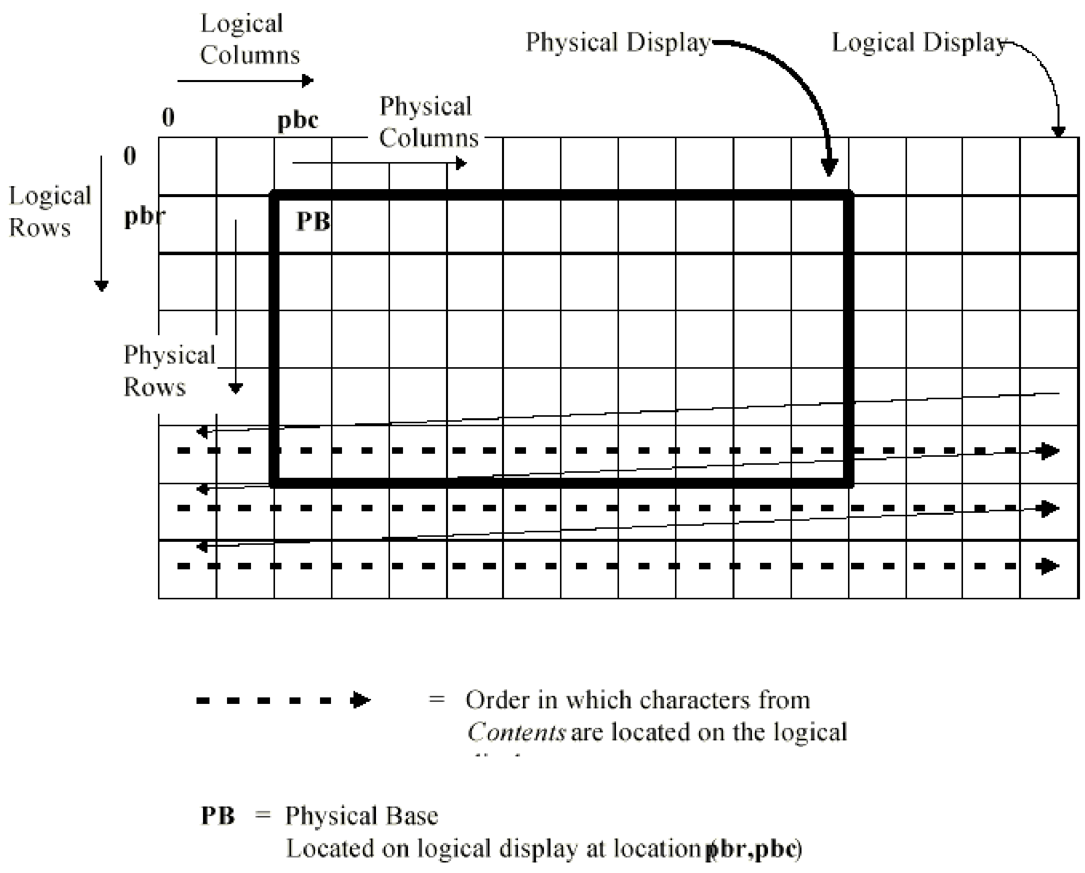
6.1.3.1.6 Display

A *display* is a component which presents a two dimensional array of characters associated with the physical element. A physical element can have several displays. A display may be real or virtual; that is it may or may not actually be present on the physical device itself. Displays have eight attributes as visualized in Figure 6-5, "Display Attributes":

1. *Display Identifier* - To identify a specific display on a physical device.
2. *Logical Rows* - The number of rows on the logical display.
3. *Logical Columns* - The number of columns on the logical display.
4. *Physical Rows* - The number of rows on the physical display. This number is always smaller or equal to *Logical Rows*.
5. *Physical Columns* - The number of columns on the physical display. This number is always smaller or equal to *Logical Columns*.
6. *Physical Base* - The location of the first character of the physical display expressed as (LogicalRowNbr,LogicalColumnNbr) and identified in Figure 6-5 as (**pbr,pbc**). Note that the top-left most position in the logical display is defined as (**0,0**).
7. *Character Set* - Normally ASCII, but may be also be Unicode or a proprietary character set used by the switching function. This attribute is fixed for a given display.
8. *Contents* - A character string which represents the contents of the logical display. Spaces are always present so the size of this string is always the product of the number of logical rows and logical columns. The *contents* can be observed and/or set (if supported) by the computing function. When setting the value of the *Contents*, an offset (starting point in logical display) and a length field (number of characters to be set) shall be given).

The following figure visualizes these attributes further:

Figure 6-5 Display Attributes


Logical Columns
Physical Display
Logical Display
0
pbc
Physical Columns
0
Logical Rows
pbr
PB
Physical Rows
= Order in which characters from *Contents* are located on the logical
PB = Physical Base
Located on logical display at location (pbr,pbc)

6.1.3.2 Logical Element

A logical element is the part of a device that is used to manage and interact with calls at a device. This element represents the isochronous media stream channels (e.g., ISDN bearer channels) and associated call handling facilities that are used by the device when involved in a call (i.e.,via a connection). If a device also has a physical element, the logical element may interact with the physical element's components in order to convey call information (e.g., via lamps) to the user of the device, to provide/manage the media stream data of the call (e.g., via an auditory apparatus) for the user of the device, and to allow the user of the device to manage the calls (e.g., via buttons). The implementation of this interaction is device and switching function specific. The following are the call and call-related features/services that are controlled and observed through the logical element itself:

- Logical Device features/services which are used to indirectly control calls at the device such as Get/Set Forwarding, Get/Set Do Not Disturb and Get/Set Auto Answer.

Note that these features/services also include the associated events and monitoring of these events.

The logical element of a device is observed and/or controlled within the switching function through an assigned Device Identifier.

Note that if the device is a combination of logical and physical elements, the assigned Device Identifier is the same for both elements.

The following sections describe the attributes and components of the logical element. These attributes/component shall be controlled and observed in conjunction with the logical element (i.e., associated with the logical element's Device Identifier).

6.1.3.2.1 Appearance

An appearance is a receptor which is used to connect with at most a single call at the device. A logical element consists of one or more appearances. Appearances are also sometimes called *call appearances*. The number of appearances that a logical element can have is switching function and device specific. Changes in the number of addressable appearances for a logical element are reflected by the capabilities exchange services (13.1 beginning on page 118). Each appearance can be used independently. The following are the call and call-related features/services that are controlled and observed through an appearance for a particular call:

- Call Control features/services such as Make Call, Deflect Call, Answer Call.
- Call Associated features/services such as Associate Data, Generate DTMF, Generate Telephony Tones.
- Routeing Services such as Route Request, Route Select and Route End.
- Media Stream Access such as Attach Media Service, and Detach Media Service.
- I/O Services such as Start Data Path and Send Data (when applied to logical elements of a device.)

Note that these features/services also include the associated events and monitoring of these events.

An appearance has several attributes. The following are those attributes:

1. *Addressability* - The addressability of an appearance refers to whether or not the switching function is explicitly representing the appearance to the computing function.
   a. *Addressable* - An appearance is addressable if it can be explicitly referenced by the computing function at any time with or without the involvement in a call, through a CSTA static device identifier. Refer to Clause 10, "CSTA Device Identifier Formats" for a description of how addressable appearances are referenced.
   b. *Non-addressable* - An appearance is non-addressable if it can only be referenced, when it is involved with a call, through a CSTA connection identifier. In this case, the logical element dynamically creates and destroys appearances based on the call activity, call capabilities, and features/services of the device. Once the appearance is created (i.e., associated with a call), the corresponding Connection Identifier shall be used to control and observe the appearance. For example, when a call is presented to the device, the logical element creates an appearance to handle the call.

2. *Appearance Type* - The type of appearance is based on its relationship with other devices. The type of appearance determines the functionality and behaviour associated with the logical element of the device. There are two types of appearances: Standard and Bridged Appearances.

Refer to Annex A for a complete description of the types of appearances and their associated behaviour.

6.1.3.3 Device Configurations

A *device configuration* describes the arrangement of the various elements and appearances that can be directly associated with a given device. Multiple device configurations may be formed from the possible combinations of physical elements, logical elements, and different appearance types.

Device configurations are described in terms of a specific device configuration for a particular device:

1. *Device's element combination* - This indicates whether the device has a physical element only, a logical element only or both a logical and physical element.
2. *Other devices using the physical element* - This indicates the list of devices (i.e., their logical elements) that are using the physical element of the base device.
3. *Other devices using the logical element* - This indicates the list of devices (i.e., their physical elements) that are using the logical element of the base device.
4. *The logical element's appearance addressability* - This is an attribute of the appearances of the logical element of the base device (if the logical element is present).
5. *The logical element's appearance type* - This is an attribute of the appearances of the logical element of the base device (if the logical element is present).
6. *The number of appearances associated with the logical element* - This is an attribute of the logical element of the base device (if the logical element is present).

As a set, these attributes describe the device configuration for a specific device. The following sections illustrate typical examples of device configurations that can exist in a switching sub-domain.

Note that in the following examples, where physical and logical elements form part of the same device, the application of a suffix number to the identifying letter identifies that they are parts of the same device (e.g. L1, P1 are a single device; L1, P2 are elements from different devices).

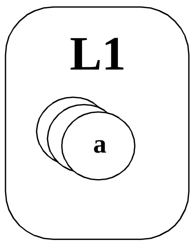
6.1.3.3.1 Logical Element Only

This device configuration has only one logical element (e.g., some Park devices). The following identify the attributes of this device configuration:

- *Device's element combination* - logical element only (L1)
- *Other devices using the physical element* - None
- *Other devices using the logical element* - None
- *The logical element's appearance addressability* - Non-addressable
- *The logical element's appearance type* - Selected-standard
- *The number of appearances associated with the logical element* - unlimited (switching function based limits)

Figure 6-6 is a diagram of a logical element only device configuration.

Figure 6-6 Logical Element Only Device Configuration



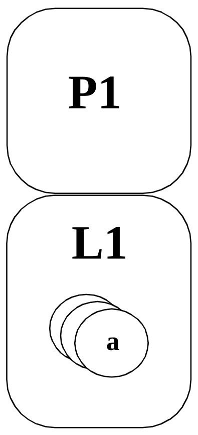
6.1.3.3.2 Single Physical and Logical Element

A *Single Physical and Logical Element* device configuration consists of a single physical element of the device associated with a single logical element of the device that contains non-addressable standard appearances. This device configuration could be used to model a basic telephone station device (e.g., a Plain Old Telephone Service (POTS) telephone or a featured telephone). The following identify the attributes of this device configuration:

- *Device's element combination* - both a logical and physical element (L1/P1)
- *Other devices using the physical element* - None
- *Other devices using the logical element* - None
- *The logical element's appearance addressability* - Non-addressable
- *The logical element's appearance type* - Selected-standard
- *The number of appearances associated with the logical element* - unlimited (switching function based limits)

Figure 6-7 is a diagram of this device configuration.

Figure 6-7 Single Physical and Logical Element Device Configuration (One Device)



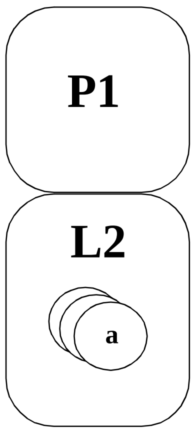
In this figure, the labels "L" and "P" represent the logical and physical elements of the device respectively.

Another variation of the single logical and physical element device configuration involves two different devices - one with a logical element only and one with a physical element only - that are associated with each other. From the perspective of the physical device element P1, the device configuration in this example can be represented as follows:

- *Device's element combination* - physical element only (P1)
- *Other devices using the physical element* - one device (L2)
- *Other devices using the logical element* - None
- *The logical element's appearance addressability* - N/A
- *The logical element's appearance type* - N/A
- *The number of appearances associated with the logical element* - N/A

Figure 6-8 is a diagram of this device configuration.

Figure 6-8 Single Physical and Logical Device Configuration (two devices)



6.1.3.3.3 Multiple Logical Elements

A *multiple logical elements* device configuration consists of a single physical element associated with multiple logical elements containing standard appearances. A multi-line telephone station could be modeled using this device configuration. The following identify the attributes of this device configuration:

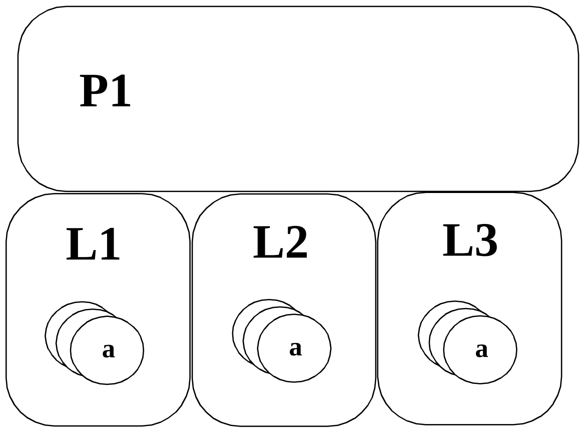
- *Device's element combination* - physical and logical element combination (L1/P1)
- *Other devices using the physical element* - two devices (L2,L3)
- *Other devices using the logical element* - None
- *The logical element's appearance addressability* - non-addressable
- *The logical element's appearance type* - Selected-standard
- *The number of appearances associated with the logical element* - unlimited (switching function based limits)

None of the logical elements in this device configuration need to be part of the same device as the physical element.

Multiple logical elements device configuration represents a single physical element (a telephone set) in a telephone system that supports only one appearance per logical element but has access to multiple calls simultaneously.

Figure 6-9 is a diagram of a multiple logical elements device configuration.

Figure 6-9 Multiple Logical Elements Device Configuration



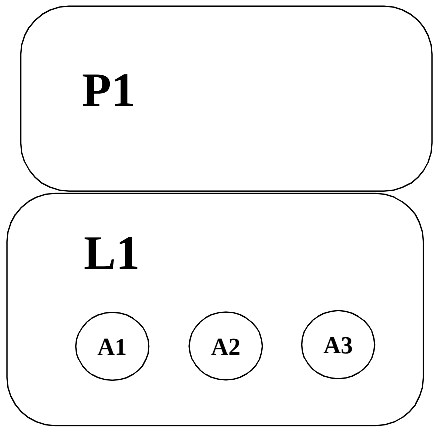
6.1.3.3.4 Multiple Appearance

A *multiple appearance* device configuration consists of single physical element and a single logical element containing two or more addressable appearances. Multiple appearance device configurations are another way to represent a single telephone set that has access to multiple calls simultaneously. This approach could be used in a telephone system that supports addressable standard appearances. This device configuration is sometimes called a *call appearance station*. The following identify the attributes of this device configuration:

- *Device's element combination* - physical and logical element combination (L1/P1)
- *Other devices using the physical element* - None
- *Other devices using the logical element* - None
- *The logical element's appearance addressability* - addressable
- *The logical element's appearance type* - Selected-standard
- *The number of appearances associated with the logical element* - 3 (A1, A2, A3)

Figure 6-10 is a diagram of a Multiple Appearance Device Configuration

Figure 6-10 Multiple Appearance Device Configuration



6.1.3.3.5 Bridged

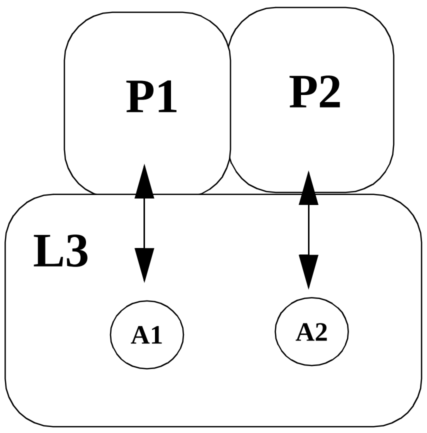
A *bridged* device configuration involves bridged appearances. The characteristics of a bridged device configuration depends upon whether the device configuration is for a physical or logical element.

In the example presented in Figure 6-11, the device configuration shown is for logical element L3 which has bridged appearances. The following identify the attributes of this device configuration:

- *Device's element combination* - logical element only (L3)
- *Other devices using the physical element* - None
- *Other devices using the logical element* - two devices (P1,P2)
- *The logical element's appearance addressability* - addressable
- *The logical element's appearance type* - Independent-shared-bridged
- *The number of appearances associated with the logical element* - 2 (A1 for P1 and A2 for P2)

Figure 6-11 is a diagram of a Bridged Device Configuration

Figure 6-11 Bridged Device Configuration

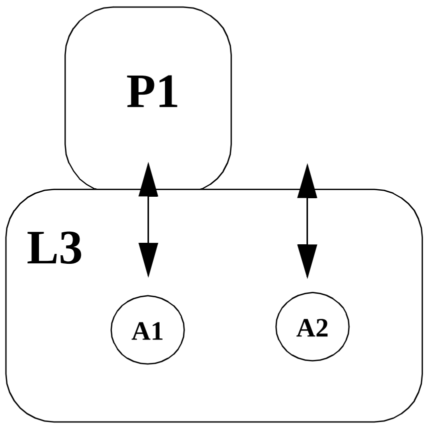


The device configuration for the physical element P1 in this example, is shown in Figure 6-12. In P1's device configuration there are only two device elements rather than three; one is the physical element (P1) and one is the logical element which has two addressable bridged appearances. The following identify the attributes of this device configuration:

- *Device's element combination* - physical element only (P1)
- *Other devices using the physical element* - one device (L3 using appearance A1)
- *Other devices using the logical element* - None
- *The logical element's appearance addressability* - N/A
- *The logical element's appearance type* - N/A
- *The number of appearances associated with the logical element* - N/A

Figure 6-12 is a diagram of a Bridged Device Configuration.

Figure 6-12 Bridged Device Configuration



6.1.3.3.6 Hybrid

A physical element associated with multiple logical elements that each have different types of appearances has a *hybrid* device configuration.

An arbitrary example of a hybrid device configuration is shown in Figure 6-13. This example consists of one physical element and three logical elements.

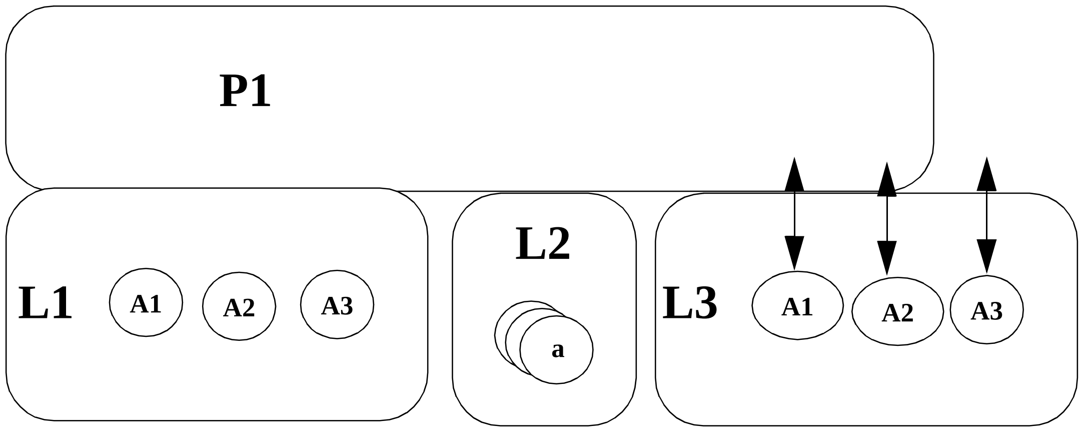
- device 1 has both a physical element (P1) and a logical element (L1) containing two addressable standard appearances
- device 2 has only a logical element (L2) with non-addressable standard appearances
- device 3 has only a logical element (L3) with three addressable bridged appearances

The following identify the attributes of this device configuration:

- *Device's element combination* - logical and physical element (L1/P1)
- *Other devices using the physical element* - two devices (L2, L3 using appearances A1 andA2)
- *Other devices using the logical element* - None
- *The logical element's appearance addressability* - addressable
- *The logical element's appearance type* - Selected-standard
- *The number of appearances associated with the logical element* - 3 (A1, A2, A3)

Figure 6-13 is a diagram of a Hybrid Device Configuration.

Figure 6-13 Hybrid Device Configuration



6.1.3.4 Device Categories

The device category of a particular device provides a generic indication of the device's behaviour and configuration. The computing function should use the device category along with other information provided by the capabilities exchange services to model a given device.

6.1.3.4.1 Station Device Category

This category of device can range from a basic "Plain Old Telephone Set" (POTS) device to a very complex feature telephone device. Station devices can be represented by any single device configuration type, or more commonly, as a hybrid of two or more different device configuration types. The physical component, if present, may have any combination of components. The logical element(s) may have any number of appearances appropriate for the type of device configuration.

6.1.3.4.2 Network Interface Device Category

A *Network Interface Device* is a category of device which is within the switching sub-domain and is connected to another telephone network.

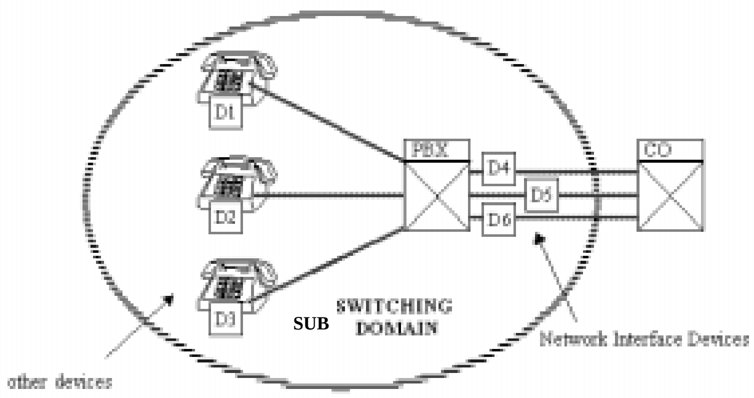
A given switching sub-domain is connected to another telephone network(s) (which may or may not be thought of as other switching sub-domains) through one or more Network Interface Devices.

To indicate when a given call involves a Network Interface Device and an external device, the switching function provides a Network Reached event to the computing function specifying what Network Interface device is being used (if known), and what subsequent call-related information is subject to the capabilities of the network being used.

The following are examples which illustrate the use of Network Interface Devices.

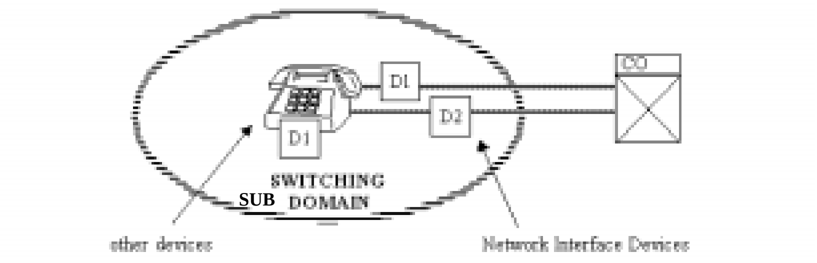
In Figure 6-14, the switching sub-domain is centered on a PBX. The Network Interface Devices are distinct devices and are commonly referred to as "trunks."

Figure 6-14 Trunks As Network Interface Devices



In Figure 6-15, the switching sub-domain is an individual telephone station connected directly to the public telephone network. In this example, the two Network Interface Devices represent two Central Office lines.

Figure 6-15 Central Office Lines As Network Interface Devices



6.1.3.4.3 ACD Device Category

An ACD (Automatic Call Distributor) is a device that distributes calls. An ACD device only consists of the distribution mechanism and may be associated with the devices to which the mechanism distributes calls.

A dynamic process, *Agent Log On* allows an association between an ACD device and a distributed-to device to be created, removed and changed at anytime. Refer to 6.1.3.7, “Agent”, on page 24 for a description of Agent and the Agent Log On Process.

An ACD device is represented as a logical element only device configuration. When a call is presented to an ACD device, a connection in the entering distribution mode of the alerting state is created. Calls that are presented to the ACD device may be queued before they are distributed. The conditions under which the call is queued are switching function specific. Many calls can be simultaneously enqueued, pending their distribution. The ACD device can, for example:

- distribute a call to an agent that had logged on to the ACD device
- distribute a call to an agent that had logged on to an ACD Group
- distribute a call to another ACD device (as well as any other type of internal or external device)
- queue a call to an ACD Group, or to resources that provide message playing, message prompting, voice response interaction, etc.

There are two ACD device models - visible ACD-related devices and non-visible ACD-related devices (note that an ACD device can support one or both models):

- *Visible ACD-Related Devices* - In this model, the ACD device, and the devices (e.g., voice announcement units, devices the call may queue to, other ACD devices, ACD groups, etc.) that can interact with the ACD device while the call is being handled by the ACD device, can be monitored/ controlled and are represented uniquely by the switching function with their own connection when associated with a call.

- *Non-Visible ACD-Related Devices* - In this model, the ACD device, and the devices (e.g., voice announcement units, devices the call may queue to, other ACD devices, ACD groups) that can interact with the ACD device while the call is being handled by the ACD device, are represented by the switching function using a single connection at the ACD device. Neither these devices, which interact with the ACD device, nor their connections to calls queued at the ACD device can be monitored or controlled. Note that when a single connection is used in this manner, all relationships between the call and the ACD device are terminated when the call is no longer part of the ACD device (diverted from the device, for example).

6.1.3.4.4 Park Device Category

A *park* device category is a device that is exclusively used by the switching function to park calls on behalf of other devices in the switching sub-domain. These calls, once parked, may be retrieved by a device in the switching sub-domain. The number of calls that can be parked at one of these devices is switching function specific. These devices can be represented by either a single physical and logical elements device configuration or a logical element only device configuration. The visibility of these devices within the switching sub-domain is switching function specific.

Note that calls may be parked at devices other than a Park device.

6.1.3.4.5 Other Device Category

A device in this category has characteristics (i.e., device configuration, capabilities, type of elements, element components) which are switching function specific.

6.1.3.5 Group Device Category

A *group* device models a relationship between CSTA devices that is characterized by the fact that these devices share a common device identifier. This relationship can be permanent or temporary. The group model differs from the bridged appearances model in that the relationship is between devices’ logical elements while for bridged appearances the relationship is between a logical element and a physical element.

The group device has a logical element that has relationships with the logical elements of the devices that are members of this group. The group device is referenced by the group device identifier.

A group device may have a distribution mechanism. The distribution mechanism may support queuing. When a group device has a distribution mechanism, the group device identifier represents both the distribution mechanism and the member devices.

6.1.3.5.1 Group Device Attributes

The attributes of a group device are (a group device may have more than one of these attributes (e.g., a hunt and pick group device):

- Hunt - The hunt attribute characterizes a group device (also called hunt group) that has the capability to distribute calls to the member devices according to different selection modes (e.g., cyclical, sequential, longest idle time). The association between the group device and the member devices is always fixed by the switching function. A hunt group may have the capability to queue these calls before they are distributed.

- Pick - The pick attribute characterizes a group device (also called pick group) that represents a collection of devices that can be used with the Group Pickup Call service. When a call is delivered to a device in the pick group, other devices in the group can be used to answer the particular call (i.e., picking the call). When the call is picked, it is diverted from the originally delivered device and connected to the device which picked the call. The details associated with the pick group feature are documented in the Group Pickup Call service. Only devices associated with the pick group device can pick a call from the given group unless the pick group device is addressable within the switching sub-domain and the pick group device is configured to allow devices outside the group to pick calls. The addressability of the pick device within the switching sub-domain is switching function specific. If the pick group device is addressable, its Device Identifier can only be used in conjunction with the Group Pickup Call service (i.e. the computing function has limited control and can not observe the pick group device). The number of devices that are associated with the pick group device is switching function and device specific. The association between the pick group device and the other devices is not visible within the switching sub-domain (i.e., does not have an explicit representation). The association between the pick group device and the other devices is fixed by the switching function. Changes in the number of devices associated with the pick group device are reflected by the capabilities exchange services.

- ACD - The ACD attribute characterizes a group device (also called an ACD group) that has a distribution mechanism similar to a hunt group. In addition, this group device represents an explicit association between the distribution mechanism (the ACD) and the distributed-to-devices. A dynamic process, *Agent Log On* allows an association between an ACD group and the distributed-to devices to be created, removed and changed at anytime. Refer to 6.1.3.7, “Agent”, on page 24 for a description of Agent and the Agent Log On Process.
- Other - The Other attribute characterizes a group device whose characteristics are switching function-dependent.

6.1.3.5.2 Group Device and Monitoring

Monitoring of a group device in which the group device includes a distribution mechanism may follow either the *group inclusive* or the *group exclusive* model. The switching function may support either model, or both models simultaneously (with individual group devices supporting either the inclusive or exclusive model). The model(s) supported by a switching function is indicated by the Get Switching Function Capabilities service.

In the group inclusive model, the scope of a monitor on the group device includes the distribution mechanism and all member devices. In the group exclusive model, the scope of the monitor on the group device includes only the distribution mechanism.

The events reported to the monitor of a group device are the same as those reported to monitors of the individual member devices, with the following exceptions:

- A single Monitor Start service request is set on the group device identifier.
- The events for the member devices, if they are in the scope of the monitor, are reported with the cross reference identifier of the monitor set on the group device.
- When an incoming call is received, the called device represents the group device and the subject device (e.g., alerting device) represents the member device.

In addition, if the group device supports a distribution mechanism, all events reported for the call when it is present at the distribution mechanism (e.g., queued), are reported with a connection identifier consisting of the call identifier and the group device identifier. When the call leaves the distribution mechanism to be delivered to a member device, a Diverted event is generated.

Some services applied to the group device are applied to all member devices of the group device (e.g., Set Agent State). This capability is indicated by the capabilities exchange services.

6.1.3.6 Named Device Types

Switching Function implementations may indicate that a device is of a particular device type. The following device types may be used for this purpose but the interpretation of a given device type is implementation specific.

- ACD
- ACD Group
- Button
- Button Group
- Conference Bridge
- Line
- Line Group
- Operator
- Operator Group
- Parking Device
- Station
- Station Group

- Trunk
- Trunk Group
- Other
- Other Group

6.1.3.7 Agent

An agent represents a device’s association and activities with one or more ACD devices or ACD groups.

An agent becomes associated with a specific ACD device or ACD group by the process of logging on. There are two Agent Log On Models that may be supported by a switching function (as indicated by the capability exchange services). Note that multiple models may be supported by a switching function and by a single agent.

- *Log On to an ACD device* - In this model an agent logs on to ACD device and becomes associated with the activities of the ACD device. There is no association with any ACD group. This may be achieved by using the Set Agent State service (loggedOn) without providing the ACD group parameter. Note that this capability cannot be simultaneously supported with the Log On to an ACD Group (implicit/one step) described below.
- *Log On to an ACD Group* - In this model an agent becomes associated with the activities of an ACD group. This may be accomplished by the following:
  - explicit/one step - by using the Set Agent State (loggedOn) service and supplying the ACD group parameter
  - explicit/two step - by first logging on to an ACD device using the Set Agent State (loggedOn) service and omitting the ACD group parameter (described above). Once logged on to an ACD device, a log on to a specific group is achieved by using another Set Agent State (loggedOn) service with the ACD group parameter provided.
  - implicit/one step - by logging on to an ACD Group using the Set Agent State (loggedOn) service and omitting the ACD group parameter. If supported by the switching function, the agent is automatically logged on to an ACD group. Note that this capability cannot be simultaneously supported with the Log On to an ACD device.

An agent has several attributes that can be controlled and observed by the computing function, as described below.

6.1.3.7.1 Agent Identifier

Each agent that can be observed and/or controlled within the switching function is referenced using an assigned identifier. This identifier is associated with the logical element’s Device Identifier of the device. Certain switching functions may not assign an agent identifier to the agent. In these cases, the logical element’s Device Identifier is used to represent the agent associated with the device. The format of the agent identifier is switching function specific.

There are two ways that an agent identifier may be provided:

- as a parameter in an event (via the agentID parameter in the Agent Logged Off event, for example) or
- as a sub-string in the Switching Function Representation format of a logical element’s Device Identifier, thereby making a unique identifier for the logical element and the associated agent.

When an agent is associated with more than one ACD group and the switching function assigns a different agent identifier for each ACD group that the agent is associated with, then the following applies:

- The agent identifier shall be supplied (via the agentID parameter) on the agent state events associated with this agent (e.g., Agent Ready, Agent Not Ready).
- This agent identifier may or may not be part of the Device Identifier parameters (i.e., the agent identifier sub-string in the Switching Function Representation format) in the call control events that are associated with the agent in an ACD call. Non-ACD calls are unaffected.
- This agent identifier shall be part of the Device Identifier parameter (i.e., the agent identifier sub-string in the Switching Function Representation format) on services when the computing function want to focus the service

at a particular ACD call or group, otherwise if not supplied, the switching function will choose which ACD call or group the service is focused.

When an agent is associated with one ACD group, the switching function may or may not assign an agent identifier to the agent. If the agent identifier is not assigned, the logical element's Device Identifier of the agent device is used to represent the agent in the ACD group. If the agent identifier is assigned, then either the logical element's Device Identifier of the agent device or the agent identifier may be used to represent the agent in the ACD group. This also applies to the case where the switching function has only one agent identifier (either assigned or not) for an agent that is associated with multiple ACD groups. When only one agent identifier is assigned to an agent the following applies:

- The agent identifier may be supplied (via the agentID parameter) on the agent status events associated with this agent (e.g., Agent Ready, Agent Not Ready).
- This agent identifier may be part of the Device Identifier parameters (i.e., the agent identifier sub-string in the Switching Function Representation format) in call control events that are associated with the agent in an ACD call. Non-ACD calls are unaffected.
- This agent identifier may be supplied as part of the Device Identifier parameter (i.e., the agent identifier sub-string in the Switching Function Representation format) on services.

Note that when the agent identifier is supplied as part of the Device Identifier and the agent identifier is not associated with the device, the service request shall be rejected with a negative acknowledgement.

6.1.3.7.2 Agent Password

The agent password authenticates the agent's association with a given device and ACD device or ACD group in the switching sub-domain. The agent password is used when the agent is associated with the device and made available to the ACD device or ACD group (i.e., log on) and when the agent is made unavailable to the ACD device or ACD group (i.e., log off). An agent password may be assigned to each agent. For more details, refer to the definition of the agentPassword parameter in associated services.

6.1.3.7.3 Agent Group Association

This identifies the ACD group that is associated with the agent. There can be zero or more ACD groups associated with the same agent. This is represented via the ACD group's logical element's Device Identifier.

6.1.3.7.4 Agent State

A state that an agent may take in relation to an ACD device or ACD group and the calls associated with the ACD device or ACD group.

The computing function is informed of agent state changes through agent state event reporting. These event reports are sent to the computing function when a monitor is started on either the ACD device or ACD group (if supported by the switching function) or the device associated with the agent or agents. The following are the agent states with respect to an agent and a particular ACD device or ACD group:

- *Agent Null* - The state where an agent is not logged-on to (i.e., logged off from) the ACD device or ACD group at a particular device. Logging-on and logging-off from an ACD device or ACD group shall cause the transitions from and to this state. The event report that represents the entry to this state is the Agent Logged Off event. Note that the presence/absence of the ACD Group parameter in this event determines if the agent has logged off of an ACD device or an ACD group
- *Agent Logged On* - The state where an agent is logged-on at a particular device to an ACD device or ACD group and is ready to contribute to the activities of the ACD device or ACD group. This state does not indicate that the agent is ready to accept ACD calls (refer to the Agent Ready event section). The event report that represents the entry to this state is the Agent Logged On event. Note that the presence/absence of the ACD Group parameter in this event determines if the agent has logged on to an ACD device or an ACD group.
- *Agent Not Ready* - The state where an agent is logged-on at a particular device to an ACD device or ACD group but is not prepared to handle calls that the ACD distributes. While in this state an agent may receive calls that are not ACD calls. The event report that represents the entry to this state is the Agent Not Ready event.

- *Agent Ready* - The state where an agent is logged-on at a particular device to an ACD device or ACD group and is prepared to handle ACD calls even though it may be involved with non-ACD calls. The event report that represents the entry to this state is the Agent Ready event.
- *Agent Busy* - The state where an agent is involved with an existing ACD call at the device, even if that call is on hold at that device. Calls between agents, calls between supervisors and agents and private calls may or may not cause this transition. The event report that represents the entry to this state is the Agent Busy event.
- *Agent Working After Call* - The state where an agent is no longer connected to an ACD call but is still occupied with work related to a previous ACD call. In this state, an agent cannot receive ACD calls but may be able to receive non-ACD calls. The agent may be performing administrative duties (e.g., updating a business order form) for a previous call, or may be involved with a non-ACD call. The event report that represents the entry to this state is the Agent Working After Call event.

Note that this Standard does not impose or restrict the transition of an agent state to another agent state. Any implementation restrictions shall be reflected by a negative acknowledgement to the Set Agent State service request.

6.1.3.7.5 Agent State Models

It is possible that an agent may have several agent states with respect to the different associated ACD devices or ACD groups. Alternatively, an agent may have a single state to describe its relationship to all associated ACD devices or ACD groups (if the state is, in fact, the same for all of them).

The following Multi-State Agent models that indicate how the switching function maintains the agent states for each ACD or ACD group that is associated with an agent. (The capability exchange services indicate which models are supported by a switching function.) The models are:

- *Agent Multi-State Model (Independent Group Working)* - In this model, as illustrated in Table 6-1, the switching function maintains an independent agent state model for each ACD device or ACD group associated with an agent. All agent state transitions are totally independent. The cause code of Forced does not appear since no state is forced. (Note that the numbers between parenthesis in the following figures represent the ACD group number).

Table 6-1 Agent Multi-State Model (Independent Group Working) Illustration

	Step 1	Step 2	Step 3	Step 4	Step 5
ACD Group 1 State	Not Ready (1)	Not Ready (1)	Not Ready (1)	Not Ready (1)	Ready (1)
ACD Group 2 State	Not Ready (2)	Ready (2)	Busy (2)	Working After Call (2)	Ready (2)
ACD Group 3 State	Not Ready (3)	Ready (3)	Ready (3)	Ready (3)	Busy (3)
ACD Device State	Ready	Busy	Busy	Busy	Ready

- *Agent Multi-State Model (Semi-Independent Linked)* - In this model, as illustrated in Table 6-2, the switching function maintains an agent state model for each ACD device or ACD group associated with an agent. However the agent state models for each ACD device or ACD group are linked together. For example, if an agent is associated with two ACD groups (both states are Ready), and the agent connects to an ACD call for the first ACD group, the agent state for the first group transitions to Busy while the agent state for the second group goes to Not Ready (with a cause code of Forced to indicate that the state transition was the result of another ACD device or ACD group activity).

Table 6-2 Agent Multi-State Model (Semi-Independent Linked) Illustration

	Step 1	Step 2	Step 3	Step 4	Step 5
ACD Group 1 State	Not Ready (1)	Not Ready (1)	Not Ready (1) - Forced cause	Not Ready (1) - Forced cause	Not Ready (1)
ACD Group 2 State	Not Ready (2)	Ready (2)	Busy (2)	Working After Call (2)	Ready (2)

Table 6-2 Agent Multi-State Model (Semi-Independent Linked) Illustration (continued)

	Step 1	Step 2	Step 3	Step 4	Step 5
ACD Group 3 State	Not Ready (3)	Ready (3)	Not Ready (3) - Forced cause	Not Ready (3) - Forced cause	Busy (3)
ACD Device State	Ready	Ready	Not Ready - Forced cause	Not Ready - Forced cause	Ready

- *Agent Orientated Model* - In this model, as illustrated in Table 6-3, the switching function maintains one state for the agent, no matter how many ACD devices or ACD groups the agent is associated with. When the ACD Group parameter is included in an agent event, it indicates the ACD group that has caused the agent state transition. When the ACD Group parameter is not included in an agent event, it indicates that non-ACD group activity has occurred.

Table 6-3 Agent Orientated Model Illustration

	Step 1	Step 2	Step 3	Step 4	Step 5
ACD Group 1 State	Not Ready	Ready	Busy	Working After Call	Ready
ACD Group2 State	Not Ready	Ready	Busy	Working After Call	Ready
ACD Group 3 State	Not Ready	Ready	Busy	Working After Call	Ready
ACD Device State	Not Ready	Ready	Busy	Working After Call	Ready
Single Event	Not Ready (2)	Ready (2)	Busy(2)	Working After Call (2)	Ready (2)

6.1.4 Call

Calls are communication relationships between one or more CSTA Devices. A call's behaviour can be observed and manipulated across the CSTA service boundary (also called service boundary in this Standard). During some call phases (e.g., establishment and release) the call is not completely formed and there may be only a single CSTA Device involved (for example, the CSTA Device on whose behalf the call was initiated). In some call control operations, such as a conference and transfer, one CSTA Device in a call is replaced with another CSTA Device, or two calls are merged into a single call.

The CSTA Call attributes, which are described in detail in the following sub-clauses, shall be:

- Call Identifier
- Correlator Data
- User Data
- Media Call Characteristics
- Account Information
- Authorisation Code
- Charging Information

6.1.4.1 Call Identifier

A CSTA *Call Identifier* (also called *Call Identifier* in this Standard) is a reference associated with a call whereby the call is known to the switching, computing and special resource functions through the call's life. A Call Identifier shall be allocated to each call by the Switching Function, at the latest, when the call first becomes visible across a CSTA Service Boundary. It shall be unique within a switching sub-domain and shall be the same for all CSTA Devices in the call. A Call Identifier can be assigned to a call before the call is fully established. For example, an incoming call may be assigned a Call Identifier when the called CSTA Device is alerting and before the call has been answered. A Call Identifier shall not only reference the entire call within the sub-domain but shall also infer a reference to that part of the call that is outside the sub-domain.

A call can pass through various stages involving many different CSTA Devices before it finally terminates. Some of these stages cause a call to change identifiers. Examples of services that cause this are Transfer and Conference. During the operation of these services, or as a result of manual intervention, the Call Identifier may change as a result of two calls being merged by the switching function but the call shall continue as a CSTA object. This merger results in the Call Identifiers for both old calls changing to a new identifier for the resulting calls in which the CSTA Devices are involved. The respective Conferenced or Transferred event specifies the transition from the old Call Identifiers to the new Call Identifiers indicating the old invalid Call Identifiers. Management of Call Identifiers is covered in section 6.1.8, "Management of Dynamically-Assigned Identifiers".

6.1.4.2 Call State

The term *Call State* means the collective set of Connection states for all the Connections comprising a call. Call state is returned only by the Snapshot Device Service for CSTA Devices that have calls. Connection states are further described in 6.1.5, "Connection", on page 31. Call states are described in more detail in 6.1.6, "Call State Definitions", on page 35.

6.1.4.3 Correlator Data

Correlator Data is computing function specific data which has been attached to a call that a computing function is controlling or observing. This information, for example, might be a key to a database entry, an application command sequence, file name, etc. Once Correlator Data is associated with a call, it remains with the call for its entire duration (at least one CSTA Device is actively involved in the call), or until the computing function overwrites the data with new data. In order to remove data, the computing function shall overwrite the existing data with null data. Correlator Data enters the switching sub-domain in two ways:

1. A computing function provides Correlator Data on a service request (e.g. Call Control and Call Associated services).
2. Correlator Data arrives from an external network connection with a call (for example, Correlator Data may be used as a key to pop a screen for the call). Correlator Data is delivered through an external network via the format described in Annex B.

Permitting a computing function to associate its own information with a call allows multiple computing functions to share data on calls which they are all controlling or observing. This feature is useful when calls are moving from one computing function to another in a distributed computer network or from one switching sub-domain to another. For more information on Correlator Data, refer to 12.2.9, "CorrelatorData", on page 87.

Correlator Data is provided by the Computing Function and associated with the call for its entire duration or until overwritten with new data. This data survives Conference and Transfer and can be provided on various events. An application may remove the Correlator Data by overwriting existing data with null data.

When Correlator Data is associated with a call, call events that indicate that a CSTA Device has become part of a call (such as Delivered, Established and Queued events, for example) shall include the Correlator Data (if this parameter is supported in the event being reported). Subsequent call events also may contain the Correlator Data.

Note that "null Correlator Data" means Correlator Data information with zero length.

When a consultation call is transferred or conferenced and null or non-null Correlator Data is associated with either (or both) the primary or secondary call, the Correlator Data in the resulting call shall always be the same Correlator Data that was associated with the secondary call (even if only the primary call had non-null Correlator Data). In that case the Correlator Data (if any) associated with the primary call is discarded. If the secondary call contains no Correlator Data, the Correlator Data associated with the resulting call is that which was associated with the primary call.

Table 6-4 Inheritance rules for Correlator Data in Conference and Transfer

Secondary Call	**Primary Call**		
	No Correlator Data	**Null Correlator Data**	**Correlator Data 1**
No Correlator Data	No Correlator Data	Null Correlator Data	Correlator Data 1
Null Correlator Data	Null Correlator Data	Null Correlator Data	Null Correlator Data

Table 6-4 Inheritance rules for Correlator Data in Conference and Transfer (continued)

Secondary Call	Primary Call		
	No Correlator Data	**Null Correlator Data**	**Correlator Data 1**
Correlator Data 2	Correlator Data 2	Correlator Data 2	Correlator Data 2

When Correlator Data is associated with a call via the Associate Data service, the Call Information event is provided by the switching function. If the data is changed by any other service, the switching function does not provide the Call Information event.

6.1.4.4 User Data

User Data is call-related computing function-to-computing function information that, unlike Correlator Data, is not associated with a call for the life of the call. The switching function receives User Data in two ways:

- A computing function sends User Data in a service request.
- User Data arrives from an external network connection with a call (for example, User Data may be used as a key to pop a screen for the call).

Both a computing function and the network may send User Data in two ways:

1. *With Call Control Activity* - Call control service requests (and network signalling for call control) permit User Data as an optional parameter. The switching function reflects the delivery of User Data in the first call control event that results from the switching function or network carrying out the call control activity. When the computing function provides User Data in a Call Control service request, the User Data is delivered to other parties if and only if the call control service successfully completes. If the switching function does not generate the call control event that corresponds to the call control activity because the computing function has set an event filter that filters out the relevant event, then the User Data is not propagated to subsequent events and the User Data information will be lost. Refer to the description of the individual Call Control services for more details on the events that will contain User Data for those services.
2. *Independent of Call Control Activity* - The computing function may use the Send User Information service to pass User Data at any time. Some networks provide an independent signalling mechanism for sending User Data. The switching function generates a Call Information event with the userData parameter containing the received User Data to reflect the delivery of the User Data. Independent of call control activity, this event is generated for all computing functions monitoring the call and all computing functions monitoring any CSTA Devices that have a connection to that call.

This standard defines a mechanism for delivering both User Data and Correlator Data through an external network at the same time. This mechanism is described in Annex B.

When a computing function uses a service to send User Data, the switching function sends that User Data only after the switching function sends a positive acknowledgement to the service request.

User Data is described further in 12.2.22, “UserData”, on page 106.

6.1.4.5 Other Call Related Information

There is additional information associated with calls such as:

- *Account Information* - A computing function or business-specific piece of information that is assigned to a call for accounting purposes. For more information, see 12.2.1, “AccountInfo”, on page 84.
- *Authorisation Code* - A code provided to the switching function that is used to check if a computing function user is authorised to perform a given service. For more information see 12.2.3, “AuthCode”, on page 84.
- *Charging Information* - An amount charged to a device for a call in which the device was involved. For more information, see 12.2.6, “ChargingInfo”, on page 85.

6.1.4.6 Media Call Characteristics

This Standard covers the control and observation of calls within either a voice or digital data switching sub-domain (e.g., network) or a switching sub-domain that is a combination of both (voice and digital data). Within the set of possible digital data switching sub-domains, this Standard is limited to a sub-set of these switching sub-domains with the following characteristics:

- connection-oriented
- circuit or cell-based packet switching
- point-to-point, and multi-point topology
- A connection in a call may represent either one or many data channels within the switching sub-domain.
- A CSTA Device can have multiple calls but the calls are totally unrelated to one another (i.e., they can not be conferenced, transferred, join, etc.).

Examples of these digital data switching sub-domains that support these characteristics are;

- T1
- ISO-Ethernet (TDM part of the protocol)
- ISDN
- Switched 56
- RSVP
- ATM (B-ISDN)

As a result, another attribute of a call is the Media Class (i.e., Voice or Data). Voice calls and the sub-set of digital data calls as described above use the same model for control and observation but with some additional unique characteristics for digital data. The following is the list of characteristics:

1. *Media Class* - Describes the type of call. It may consist of one or more of the following classes.
   - **Data** - These types of calls involve digital data calls (both circuit switched and packet switched). Devices that may be involved with these types of calls are digital computer interfaces and G4 facsimile machines.
   - **Image** - Digital data calls involving imaging, or high-speed, circuit-switched data in general. Devices that may be involved with these types of calls include digital video telephones and CODECs.
   - **Voice** - Devices in this class are used to make speech calls. This class includes standard telephones.
   - **Audio** - 3.1 KHz audio. Devices in this class are used to make audio calls excluding speech calls. It includes G3 fax and facsimile machines.
   - **Other** - A class comprising devices not in the Data, Image, Audio or Voice classes.
2. *Connection Rate* - This characteristic reflects the amount of bandwidth which is needed or allocated for a digital data call. This characteristic is specified in kilobits per second. This characteristic also represents the amount of bandwidth for both directions of data flow. The computing function can learn about a switching function’s supported rates through the capability exchange services.
3. *Bit Rate* - This characteristic specifies if the bit rate is or needs to be a constant rate (i.e., dedicated bandwidth and constant rate of media stream delivery) or variable rate. A constant rate is used for media streams like audio or video. A variable rate is used for media stream like computer-generated data transfer.
4. *Delay Tolerance* - This characteristic specifies the maximum amount of media stream delivery delay that will be toleranced for the call. If the bit rate is constant, then this value will indicate the actual amount of media stream delivery delay for the life of the call. Where as if the bit rate is variable, it will be the maximum delay allowed during the life of the call.
5. *Switching Function Call Control Information Elements* - This characteristic provides a mechanism which enables the use of the switching function private call control information elements to be used during the life of a digital data call (e.g., elements used during call setup). The format, meaning and behaviour of these information elements are specific to the given switching function. This standard allows the following types of switching function private call control information elements:

- *ISDN* - All information elements associated with call control from the ITU Q.931 standard
- *ATM* - All information elements associated with call control from the ITU B-ISDN Q.2931 standard
- *RSVP* - All information elements associated with the call control from the IETF RSVP functional specification
- *ISO-Ethernet (TDM part of the protocol)* - All information elements associated with call control from the 802.9 standard (a modified version of Q.933).
- Private ISDN - All information elements associated with call control and supplementary service control from ISO/IEC 11572 and ISO/IEC 11582.
- *Other* - All information elements associated with call control from the particular switching function specification

For details on the specific information elements of these types, see the appropriate standards or switching function specific documentation. If a given information element overlaps with an attributes or characteristic that is defined by this Standard, the information element should not be used and the attribute or characteristic in this Standard should be used. If both are supplied, then they shall contain the same value. In addition, any information element that deals with device addressing shall not be used (e.g., calling party number, calling party subaddress). The CSTA Device addressing in this Standard shall be used.

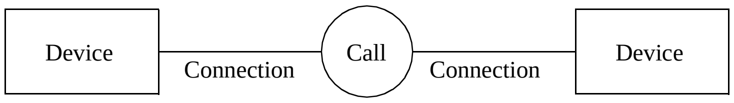
Once these characteristics are established for a digital data call, they can not be changed for the life of the call (i.e., they are static). These characteristics apply to all connections of the call.

The combination of these characteristics represents the *quality of service* associated with a given digital data call. The meaning of the quality of service which respect to characteristic combination is switching function specific.

6.1.5 Connection

A *connection* is a relationship in a switching sub-domain between a CSTA Device, and a call in which that CSTA Device is involved. This connection relationship can be both observed and manipulated. Figure 6-16 illustrates the relationship between calls, devices, and connections.

Figure 6-16 Relationship between Calls, Devices, and Connections



Observation and manipulation of these connections are the basis for call control services (such as Clear Connection, Answer Call, etc.). Connections are CSTA Objects that have the following attributes:

1. *Connection Identifier* - Each connection that can be observed and/or controlled shall be referenced across the service boundary. To accomplish this, each connection is assigned a unique identifier by the switching function. This identifier is comprised of a Device Identifier and a Call Identifier. For a call, there are as many Connection Identifiers as there are associated devices, and for a device there are as many Connection Identifiers as there are associated calls. The Connection Identifier is unique within a sub-domain and over a single service boundary. It is provided by the switching function when either a new call is created or a new device becomes involved in a a call. A Connection Identifier can change as a result of some operations (e.g., a transfer or conference) and in these cases the switching function presents the computing function with the appropriate information to transit from the old identifiers to the new. The Device Identifier used in the Connection Identifier may be either static or dynamically-assigned by the switching function.

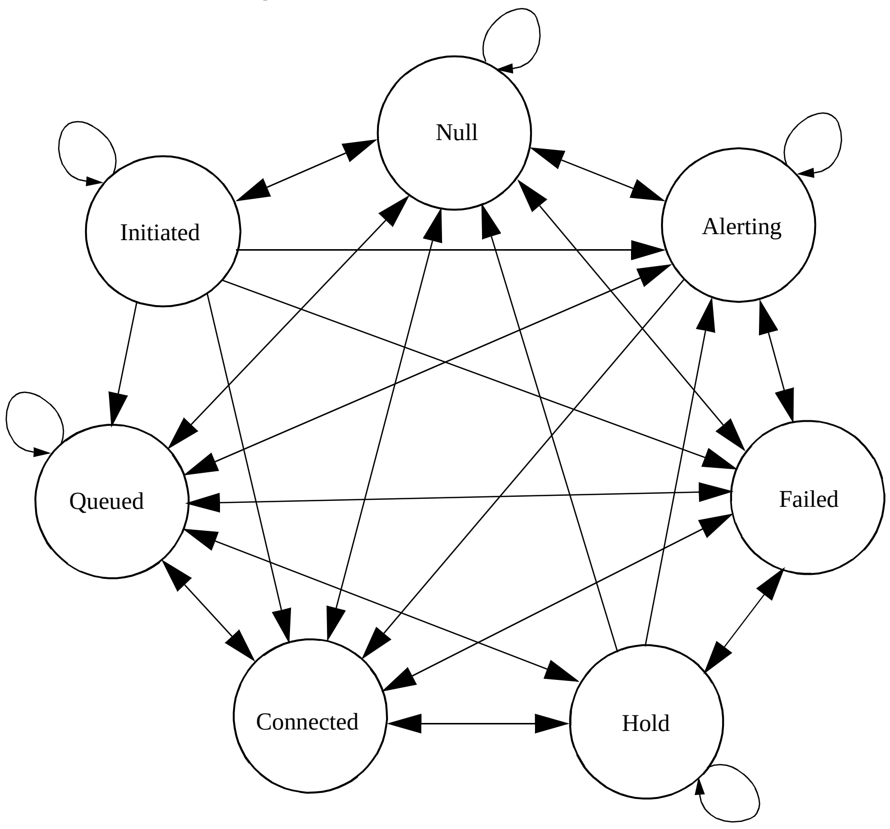
   As provided by the switching function to the computing function, a Connection Identifier will always include both a Device Identifier and a Call Identifier (unless otherwise noted in the specification of a particular CSTA event's parameters). Computing functions wanting to correlate event reports which associate devices connected together in a call can use the Call Identifier to do this correlation. The definitions of a Connection Identifier and those identifiers that it comprises (Call and Device Identifiers) restrict computing functions from fabricating Connection Identifiers.

As provided by the computing function to the switching function, a Connection Identifier shall be one which was originally provided by the switching function. An exception to this rule is where either a deviceID only or a callID only Connection Identifier is used in a specific service (as indicated by the capability exchange services). If a Connection Identifier, provided by the computing function, includes only a Device Identifier, then that Device Identifier shall be a static Device Identifier. These conditions ensure that it is possible to use only a Device Identifier (without a Call Identifier) or a Call Identifier (without a Device Identifier) to provide a Connection Identifier in certain specified circumstances.

For additional details regarding Connection Identifiers, including Connection Identifier formats and specific functional requirements, see 12.3.9, “ConnectionID”, on page 110 of this Standard.

2. *Media Stream Flow Direction* - This attribute is the direction or directions in which the media stream can be transmitted on the given connection. The following are the types of directions that can be associated with a connection:
   - *Transmit* - Media stream data can only be transmitted on the connection by the associated device.
   - *Receive* - Media stream data can only be received on the connection by the associated device.
   - *Transmit and Receive* -Media stream data can be transmitted and received on the connection by the associated device.
3. *Media Stream Channels* - A channel is a path of communications between devices within a network. Channels are created to transmit/receive the media stream when the associated connection is created for the device in the call. The correlation of a channel to an actual media stream communications path/channel within the switching function is switching function specific. The switching function may represent a channel as a group of actual media stream communication paths. A device’s connection represents a channel or set of channels on which the media stream associated with the call is to be transmitted and received. The number of channels per connection is switching function and device specific (the capability exchange services may be used to determine the value). A digital data connection can use one or more channels. In addition, there can be multiple media stream types associated with a given connection as well as the associated channels. The attachment of media services is to a connection and its associated channels as a whole. The switching function is then responsible for attaching the Media services to the appropriate channels.
4. *Connection State* - A connection state involves a single call/device relationship. When a call is present at a device, the connection representing that call at that device will transit through various stages. State transitions are observed by the computing function through event reports. The transition from one state to the next is caused by either a manual user stimulus or a CSTA service initiated across the service boundary. Connection states may also be reported by Snapshots on either calls or devices.

Figure 6-17 Connection State Model



The following are the connection state definitions:

- *Alerting* - A state in which an attempt is being made to connect a call to a device. There are three distinct modes where a connection may be in the alerting state:
  - *Offered* - In this mode, the call is in a pre-delivery state at the target device. The opportunity exists for a computing function to issue one of a set of supported services (e.g., Accept Call, Clear Connection ("reject"), Deflect Call) or an ISDN device to accept or reject the call. From the calling side perspective, the call is not delivered at the called device. As a consequence, delivery information such as Ringback indication and/or Network signalling is not provided. For example, the device makes no ringing sounds while in the Offered mode of the Alerting state. The Offered mode is indicated by an Offered event.
  - *Ringing* - In this mode, the call is being presented for the purpose of having the device connect to the call and the user is made aware that the call is being delivered at the device. The Ringing mode is indicated by a Delivered event with a cause code other than "Entering Distribution". The actual device activity to notify the user (e.g., ringing) is reported through the physical device feature events.
  - *Entering Distribution* - In this mode, a call is being delivered to a distribution device. The Entering Distribution mode is indicated by a Delivered event with a cause code of "Entering Distribution".
- *Connected* - A state in which a device is actively participating in a call. This state includes logical participation in a call as well as physical participation.
- *Failed* - A state in which call progression has been aborted. This state generally results when a device tries to become Connected to a call or a call tries to become Connected to a device and the attempt fails. The Failed state can result from failure to connect the calling device and call, failure to connect the called device and call, failure to create the call, failure when the call ends and other reasons. Refer to 6.8.2, "Connection Failure", on page 53 for more information.
- *Hold* - A state in which a device is inactively participating in a call. This state includes logical participation in a call while physical participation is suspended.

- *Initiated* - A state in which a device is requesting a service or in the process of dialling the necessary digit sequence to initiate a call to another device. The connection enters this state when the device goes off-hook (e.g., receiving dialtone) or the device is being prompted to go off-hook as a result of some service being initiated for the device.
- *Null* - A state in which there is no relationship between a call and device.
- *Queued* - A state in which call progression is suspended or made inactive while awaiting some form of action. Examples of situations in which a connection might transit to the Queued state include (among others) the following:
    - A call is parked at a device.
    - A call is queued at a distribution mechanism, waiting for an agent to become available.
    - A call is camped on to a device.
    - An appearance of a shared bridged device configuration is inactive with respect to the call.

Table D-1 on page 628 provides an example to illustrate each transition which is illustrated in Figure 6-17, "Connection State Model"." [1].

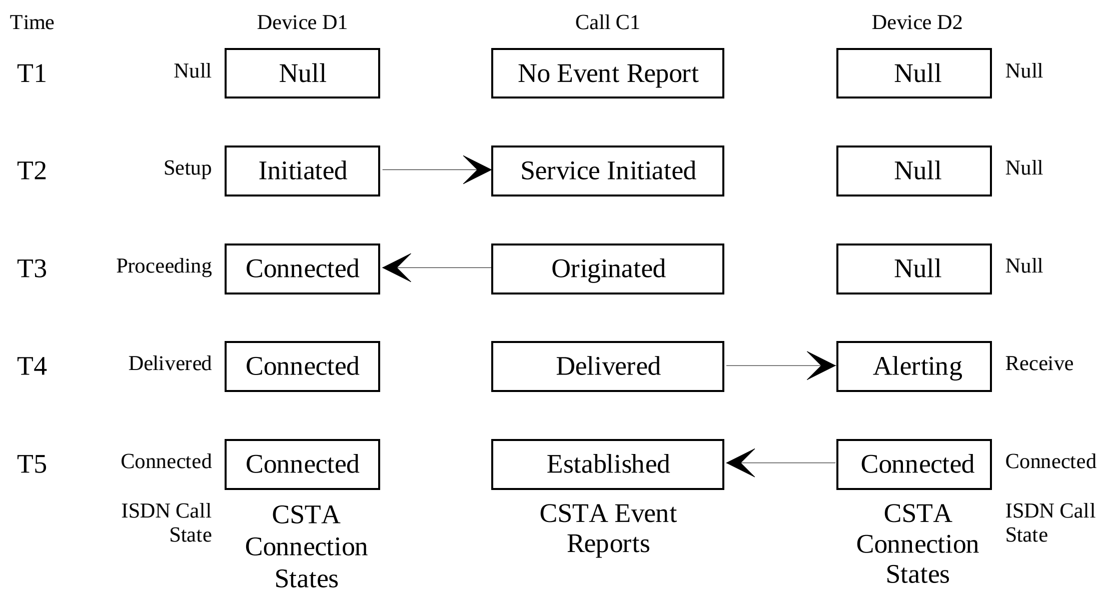
6.1.5.1 Call Event Reports

The Connection state model provides an abstract view of actual states and events that are communicated via underlying signalling systems. This abstract view is introduced to provide a language for describing CSTA Event Reports, states and Functional descriptions. Because of the topology of the Switching Function, the signals that report events and state changes have definite sources. Providing a telecommunications object (the Connection) that can be associated with the source of these signals helps when explaining the meaning of events and the operation of CSTA (and other) telecommunications services.

Note that on a typical ISDN access to a network there exists a distributed state machine. One part of this distributed state machine resides in the ISDN device. Another part resides on the other side of the ISDN access. There is another similar distributed-state access machine that resides across the ISDN network at a similar device. Using this concept, a call can be modeled as a collection of Connection state machines communicating with one another using signalling. When this communication occurs, a CSTA Event Report can be generated. In the following figure, this concept of communication between two state machines is illustrated for the case of establishing a simple call. Additionally, on each side of the figure the ISDN call states are indicated.

1. For an explanation of the Initial and Final State diagrams' nomenclature used the Table D-1, refer to Clause 11, "Template Descriptions", beginning on page 79.

Figure 6-18 Relationship of CSTA Call Event Reports



Notice in Figure 6-18 that the CSTA Event Reports are based on signalling interactions of the Switching Function. Many Connection events are of interest to CSTA applications. Typically, however, a CSTA application is interested in atomic telecommunications activities and these often involve many simultaneous Connection events. Generally, telecommunications operations embody changes to many Connections. These events can be summarized in a single Event Report. For instance, the Transfer, Conference and Clear Call Services all make changes to multiple Connections but are each represented by single Event Reports. The Connection state changes associated with each CSTA Event Report are defined in this Standard.

6.1.6 Call State Definitions

The state of a CSTA Call can be precisely expressed as the list of Connection states of all the devices involved in the call. This list is called the Compound Call State. The technique of listing the Connection states to describe the Call state can describe any call state that is possible in CSTA. However, most calls involve a small number of widely recognized states. CSTA defines those states in terms of their set of Connection states, but communicates them as atomic Call states - not as a list. These widely recognized states are called the Simple Call States.

For calls with one known Connection state, the single Connection state shall be provided as a Call state.

Note that since Null can be a known Connection state, for a nascent call it is possible to have a CSTA Call state with only one non-Null Connection (see Table 6-5).

For calls with more than two non-Null Connection states, the list of Connection states is provided as the call’s state.

CSTA simplifies Call states by relating them (at times) to particular devices. These relationships are described by differentiating the call’s Connection states. The Connection state associated with a particular device is called the local Connection state (for that device). Other Connection states are not differentiated from one another. Thus, CSTA Call state is defined for a device by the combination of Connection states as well as the order in which the Connection states are combined. For example, the Alerting-Connected Call state is not the same as Connected-Alerting. The first is defined as Received and the second is defined as Delivered. For calls with exactly two Connections, the CSTA Call state assigned to the combinations of Connection states are summarized in the following table. If there is no Simple Call state for a particular combination of Connection states, then a Compound state shall be provided as the Call state. For Compound Call states, the first Connection state in the list shall be the local Connection state.

Table 6-5 Definition of CSTA Simple Call States

Local Connection State	Other Connection State	CSTA Simple Call State
Alerting	Connected	Received
Alerting	Hold	Received-On Hold
Connected	Alerting	Delivered
Connected	Connected	Established
Connected	Failed	Failed
Connected	Hold	Established-On Hold
Connected	Null	Originated/Terminated
Connected	Queued	Queued
Failed	Null	Blocked
Hold	Alerting	Delivered-Held
Hold	Connected	Established-Held
Hold	Failed	Failed-Held
Hold	Queued	Queued-Held
Initiated	Null	Pending
Null	Null	Null

NOTE

The Originated / Terminated state may occur both during call set-up and when the call ends. When a far-end party drops from a two-party call and the near-end end-point is not returned immediately to idle, then the Originated / Terminated state is entered for call tear-down. It is also possible to enter a blocked state when a call ends.

6.1.7 Referencing Devices, Elements, Appearances and Device Configurations

In services and events, devices, elements, appearances and device configurations are referenced using Device Identifiers or Connection Identifiers when a call is present at the device, element, or appearance. Table 6-6 indicates how these Device Identifier references are interpreted. The Connection Identifier references are described in 6.1.5, “Connection”, and when a Connection Identifier is used it refers to the specific device, element or appearance associated with the given call. The following symbols are used in the table:

Logical	This indicates that the Device Identifier passed will be interpreted as reference to a specific logical element
Physical	This indicates that the Device Identifier passed will be interpreted as reference to a specific physical element
Device	This indicates that the Device Identifier passed will be interpreted as reference to the entire device configuration.
Appearance	This indicates that the Device Identifier passed will be interpreted as reference to a specific addressable appearance of a logical element. In order for computing function to determine what type of referencing is supported for a given logical element, it shall use the capability exchange services (13.1 beginning on page 118).

Refer to 10.1, “Device Identifier Formats”, for the format of Device Identifiers.

Table 6-6 Device Identifier Interpretation

Service/ Event Categories	Identifier Represents	Additional Information
Call Control, Call Associated, Media Service and Routeing Services	Device	The device (configuration) itself selects which appearance is to be used.
	Appearance	The Device Identifier selects the specific appearance which is to be targeted by the service.
Call Control, Call Associated and Media Service Events, as well as switching function to computing function	Device	The Device Identifiers that are associated with an appearance identify only the associated device configuration rather than a specific appearance. Note that this information relates to the content of the event parameters, not what is supplied on the Monitor Start service.
	Appearance	The Device Identifier selects the specific appearances which are being reported in the event. Note that this information relates to the content of the event parameters, not what is supplied on the Monitor Start service
Logical Device Services	Logical	The Device Identifier refers to the particular logical element.
Logical Device Events	Logical	The content of the event parameters indicates the given logical element being used. Not that this information relates to the content of the event parameters, not what is supplied on the Monitor Start service.
Physical Device Services	Physical	The Device Identifier shall refer to the particular physical element.
Physical Device Events	Physical	The content of the event parameters indicates the given physical element being used. Note that this information relates to the content of the event parameters, not what is supplied on the Monitor Start service.
Device Maintenance Events	Device	The event parameter contains the Device Identifier of the device configuration.
Monitor Start Service (Call Control/ Associated, and Media Service events)	Device or Logical	The Device Identifier of the device configuration (Device) results in the observation of the entire configuration. The Device Identifier of the particular logical element (Logical) results in the observation of the entire logical element. Note that the event filter determines the types (logical or device) of the events required. The Switching Function interprets the Device Identifier to meet the requirements of the filter. This may result in the same Device Identifier being interpreted as both Device and Logical for different event categories.
	Appearance	The use of the Device Identifier results in the observation of the specific appearance.
Monitor Start Service (Logical Device events)	Logical	The use of the Device Identifier results in the observation of the particular logical element.
Monitor Start Service (Physical Device events)	Physical	The use of the Device Identifier results in the observation of the particular physical element.

Logical = logical element’s Device Identifier
Physical = physical element’s Device Identifier
Appearance = addressable appearance (can be recognised from other forms of Device Identifiers by the suffix)
Device=a Device Configuration formed by multiple devices, which is referenced by the Device Identifier		

Table 6-6 Device Identifier Interpretation (continued)

Service/ Event Categories	Identifier Represents	Additional Information
Monitor Start Service (Device Maintenance events)	Device	The use of the Device Identifier results in the observation of the device configuration.
Snapshot Services	Logical	The Device Identifier refers to the particular logical element.

Logical = logical element's Device Identifier
Physical = physical element's Device Identifier
Appearance = addressable appearance (can be recognised from other forms of Device Identifiers by the suffix)
Device=a Device Configuration formed by multiple devices, which is referenced by the Device Identifier		

6.1.8 Management of Dynamically-Assigned Identifiers

Management of dynamically-assigned Device Identifiers and Call Identifiers is provided through management of Connection Identifiers. This ensures that an identifier whose meaning is dependent on another identifier is always provided in the proper context (i.e., with the other identifier needed to resolve its meaning). For example if a Call Identifier is given relative to a device, then giving the Connection Identifier ensures that the Call Identifier is provided with its reference - the Device Identifier. Management of Connection Identifiers shall be provided as follows.

Connection Identifiers shall be provided when either a new Call is created or a new device becomes involved in a call. When a call is made a Connection Identifier shall be provided. A Connection Identifier shall be provided in Event Reports that pertain to a call. When a device becomes involved in a call, the Connection Identifier shall be provided in the Event Reports that occur at that device.

If a call changes its Call Identifier when a Conference or Transfer occurs, Connection Identifiers shall be provided to link the old Call Identifier to the new Call Identifier. Similarly, if a Device Identifier is changed, new Connection Identifiers shall be provided for the devices in the call.

Management of identifiers shall be provided via parameters included in Service acknowledgements and Event Reports.

Identifiers shall cease to be valid when their context vanishes. If a call ends, its Call Identifier is no longer valid to refer to that call. Similarly, if a device is removed from service or from a call, its dynamically-assigned Device Identifier shall become invalid.

Identifiers can be reused. Once an identifier has lost its context it may be re-used to identify another object. It is recommended that implementations not re-use identifiers immediately.

Individual Call and Device Identifiers are not guaranteed to be globally unique. CSTA requires that the combination of Call and Device Identifier be unique within a CSTA switching sub-domain. To accomplish this, either the Call Identifier, or the Device Identifier (or both) shall be unique. In many cases the Connection Identifier requires both the Call and Device Identifiers to uniquely refer to Connections in a call.

6.2 Special Resource Functions

Special Resource Functions (SRFs) are functions that are typically "added-on" to a Switching or Computing Function. They can be modeled as part of either one of the two other Functions or as something totally independent.

A Special Resource Function may provide various functions, such as voice unit services, conference bridge, fax, video, or speech recognition, to the Switching and/or Computing Functions. Functionality concerning Voice Unit Services is defined in this Standard.

6.2.1 Voice Unit

A Voice Unit is a CSTA SRF that allows messages consisting of voice stream data to be created, manipulated, played to a Connection, or recorded from a Connection. A Voice Unit can be observed and controlled using the

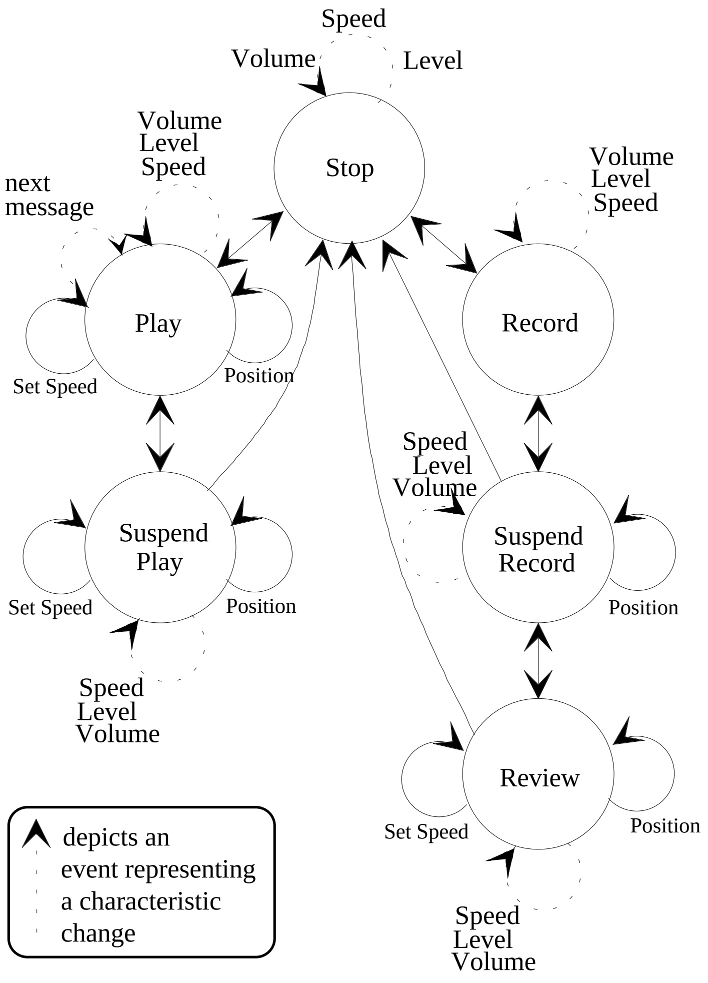
CSTA Voice Unit services (Clause 26, “Voice Unit Services & Events”, on page 517) and a state model as shown in Figure 6-19, “Voice Unit Operational Model”. A voice unit device could be used to implement a voice mail system.

A Connection Identifier is used to indicate the call that the Voice Unit relates to a message. Typically the Voice Unit will record some portion of the call or play a message as some portion of a call. There are some Voice Unit Services (e.g., Delete Message and Concatenate Message) that deal with the control of messages and do not require an interaction with a call.

A Message Identifier is used to allow manipulation of messages, many of which survive the life of multiple calls.

A Voice Unit state is a state that a Voice Unit may take in relating a call with a message. It relates a call to its message in terms of playing, recording, pausing, suspending, changing playback speed, etc. Voice Units may have several states concurrently with respect to different calls and messages. Voice Unit states shall be reported by Voice Unit Event Reports. A typical transition model for Voice Unit states is shown in the following figure.

Figure 6-19 Voice Unit Operational Model



In Figure 6-19, “Voice Unit Operational Model”, the states (circles) presented comprise the CSTA Voice Unit state set. Arrows represent transitions between states and show the typical states that may be entered from a given state. These transitions form the basis for providing Voice Unit Event Reports when they occur. The circular transitions show the effects of the Reposition and Set Speed Services. The following states are defined:

Stop	the state where a call and a message are not currently interacting.
Play	the state where a message delivers its voice stream data to a call.

Suspend Play	the state where a message that was in the Play state is temporarily suspended in its delivery. This state (rather than Stop) is entered when it the message is intended to be continued from its current position.
Record	the state where a message is created from the voice stream data in a call.
Suspend Record	the state where a message that was in the record state is temporarily suspended from recording. This state (rather than Stop) is entered when recording is intended to be continued from its current position.
Review	the state where a message that was in the Suspend Record state delivers recorded voice stream data back to the call. This allows the party who is creating the message to examine the voice stream data recorded so far.

6.3 I/O Services

I/O-Services support the exchange of data between a computer application (a computing function component) and a telephony device (to send Data from the computer application to the display of a telephony device, or to send Data from the keypad of a telephony device to the computer application, etc.).

NOTE

Both the I/O Services and the Physical Device Features provide the capability to write to the display of a device and to detect keypad activity at a device. The primary difference between these two approaches is that the I/O services operate within the context of a data path, which is described below.

The I/O-Services are defined as a distributed application between the switching function and the computing function. The special resource function is not involved.

Figure 6-20 I/O Services Functional Architecture

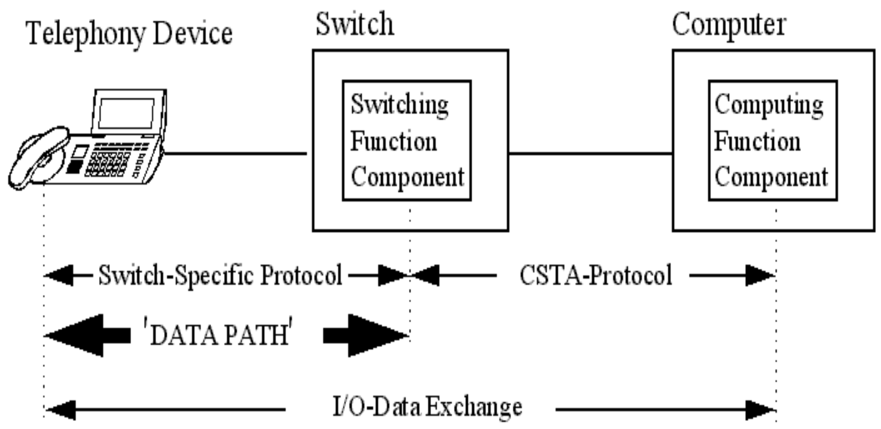


Figure 6-20 shows that the exchange of data between a telephony device and a computer application consists of two parts:

1. the exchange between the switching function component and the computing function component, provided by the CSTA-Protocol.
2. the exchange between the telephony device and the switching function component, provided by a switch-specific protocol.

6.3.1 Data Path Definition

To allow computer applications to cooperate with a switch-specific protocol (see Figure 6-20), a common view (the data path) is defined:

- The data path is an abstract model of a switch-specific protocol/mechanism.

- The data path is a logical object in the switching function that allows the exchange of data between a telephony device and a switching function component for a given application association.
- The computing function component is able to control the data path via the I/O-Services.
- The computing function component is also informed via the I/O-Services when an entity in the switching function controls the data path.
- The data path ends in the switching function component, it is not part of the CSTA-link between the switching function component and the computing function component.
- The switching function component is a gateway between the data path and the CSTA link:
    - It receives CSTA service requests from the computing function component to control the data path (via the Start Data Path, Suspend Data Path, Resume Data Path, and the Stop Data Path services) and activates the equivalent switch-specific protocol-mechanisms (and vice versa, including the Data Path Suspended/ Resumed services).
    - It receives data (via the Send Data, Send Multicast-Data, Send Broadcast-Data, and the Fast Data services) from the computing function component and sends this data through the data path (via a switch-specific protocol) to the target device (and vice versa, but only via the Send Data and Fast Data services).

6.3.2 I/O Registration Services

The I/O registration services registers the computing function as an I/O server for a specific device or for all devices within the switching sub-domain.

If the switching function supports the I/O Registration services, then the computing function shall use the I/O Register service to register for I/O services before it can receive any I/O service requests over switching function requested data paths.

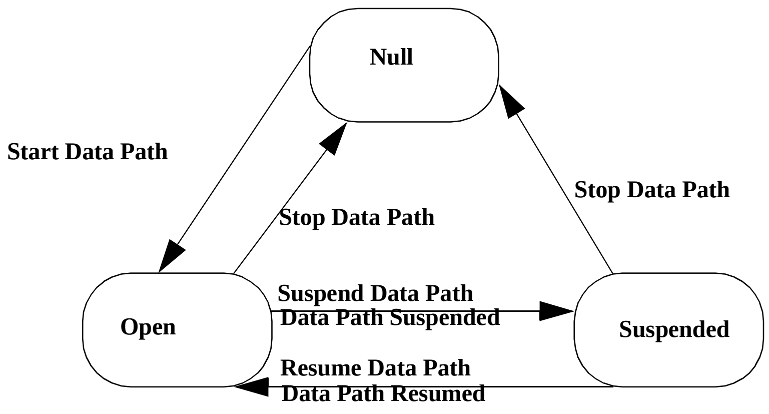
I/O Registration services are not used for data paths that are requested by the computing function.

6.3.3 Data Path States and Operational Model

The following data path states are defined in the I/O Services Operational Model:

Null	No relation between a data path and a telephony device. No I/O Cross Reference Identifier is defined.
Open	The data path is able to transfer data (direction is defined in the Start Data Path service). An I/O Cross Reference Identifier is defined.
Suspended	The data path is not able to transfer Data. The relation to a telephony device still exists, the I/O Cross Reference Identifier is still valid.

Figure 6-21 Data Path State Model



6.3.4 I/O Services Example

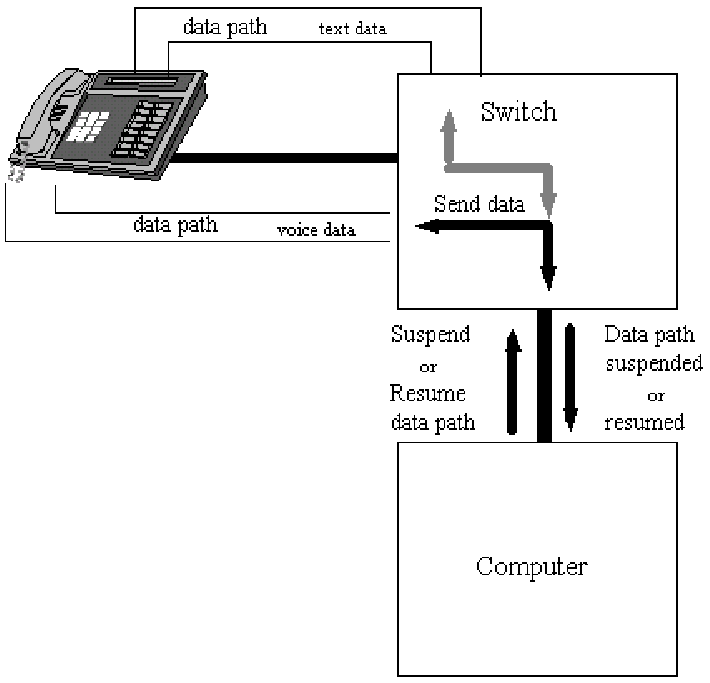
The following figure illustrates a possible CSTA configuration involving a data path from/to a CSTA object through the switching function.

The data path is established via the Start Data Path service issued, in this example, by the computing function.

If the switching function temporarily stops (suspends) the data flow (without destroying the data path), it informs the computing function via the Data Path Suspended service.

The switching function might have suspended the data path because it had received a Suspend Data Path service from the computing function or it may have suspended the data path without such a request from the computing function (because of an incoming call at the device, for example).

Figure 6-22 I/O Services Example



6.4 Call Detail Record (CDR) Services

Call Detail Record (CDR) Services allow access to information regarding call charges that has been collected, processed and/or stored by the switching function. This information may include detailed call charges, destination, bill-to-account, authorization codes, etc. This information may be provided in real-time (i.e., immediately after the conclusion of a call) or in batch mode.

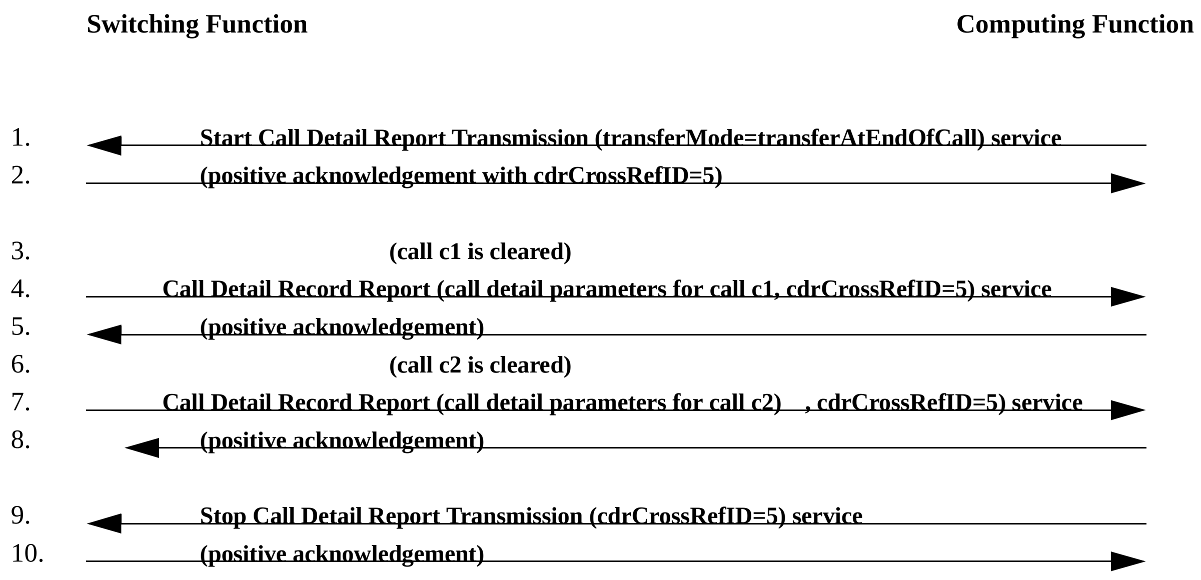
6.4.1 CDR Services Examples

Figure 6-23 illustrates the flow of CDR services when they are used to collect call detail information after every call. In this example, the computing function issues a Start Call Detail Report Transmission service and specifies that the switching function should send call detail information after every call by specifying the transferMode parameter as transferAtEndOfCall (line 1). The switching function responds with a positive acknowledgement (line 2) that includes a CDR cross reference identifier (with a value of 5) which will be present in subsequent CDR services.

Line 3 shows a call, called c1, that was cleared. Since the switching function is sending call detail information after every call, the switching function sends the call detail information for call c1 using the Call Detail Record Report service (line 4), which is acknowledged by the computing function (line 5). Lines 6 through 8 show the same sequence for another call, called c2.

When the computing function is no longer interested in CDR information, it stops the reporting by using the Stop Call Detail Report Transmission service, specifying the CDR cross reference identifier with a value of 5 (line9).

Figure 6-23 CDR Services Example: Call Details "After Every Call"


Switching Function
Computing Function
1. Start Call Detail Report Transmission (transferMode=transferAtEndOfCall) service
2. (positive acknowledgement with cdrCrossRefID=5)
3. (call c1 is cleared)
4. Call Detail Record Report (call detail parameters for call c1, cdrCrossRefID=5) service
5. (positive acknowledgement)
6. (call c2 is cleared)
7. Call Detail Record Report (call detail parameters for call c2) , cdrCrossRefID=5) service
8. (positive acknowledgement)
9. Stop Call Detail Report Transmission (cdrCrossRefID=5) service
10. (positive acknowledgement)

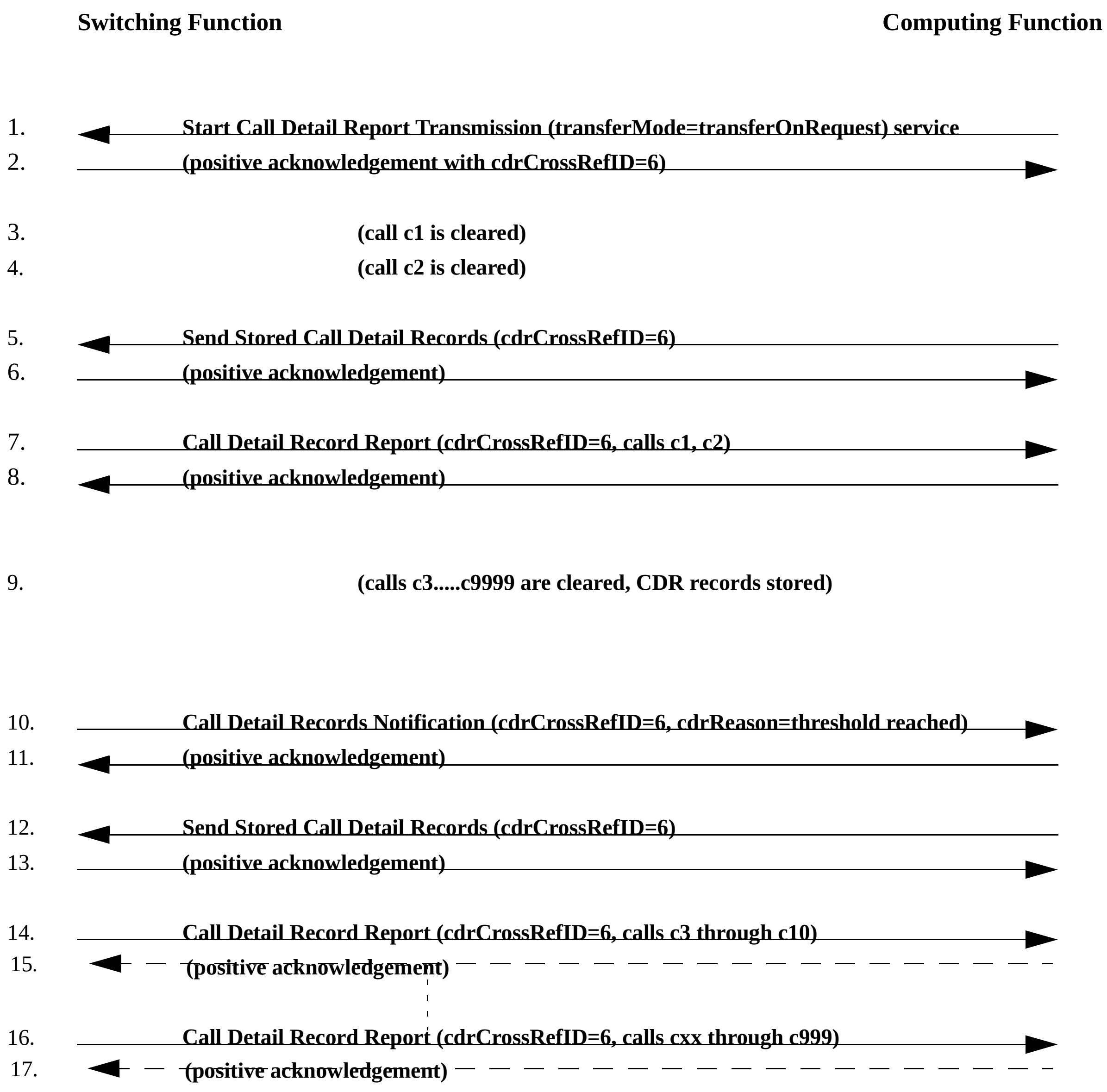
Another example as shown in Figure 6-24 illustrates the flow of CDR services when they are used to collect call detail information and retrieved at the request of the computing function.

In this example, the computing function issues a Start Call Detail Report Transmission service and specifies that the switching function should store call detail information (until explicitly requested by the computing function) by specifying the transferMode parameter as transferOnRequest (line 1).

Lines 3 and 4 show two calls in the switching function that were cleared. Since the switching function is not sending call detail information after every call, the switching function stores the CDR information instead of sending it at this time. When the computing function wants the CDR information, it sends the Send Stored Call Detail Records service (line 5) that requests the switching function to start sending its stored CDR records. The switching function sends its two stored reports for calls c1 and c2 via a Call Detail Record Report service (line 7).

Line 9 shows that the switching function has stored CDR information for a number of calls. Since the computing function has not requested CDR information to be sent during this period, the switching function has used the Call Detail Records Notification service to indicate that a threshold reached condition has occurred in the switching function (line 10). The computing function uses the Send Stored Call Detail Records service to start the transmission of CDR information from the switching function (line 12). The switching function sends a series of Call Data Record Report services that provides the stored CDR information (lines 14 & 16).

Figure 6-24 CDR Services Example: Call Details "On Request"


Switching Function
Computing Function
1. Start Call Detail Report Transmission (transferMode=transferOnRequest) service
2. (positive acknowledgement with cdrCrossRefID=6)
3. (call c1 is cleared)
4. (call c2 is cleared)
5. Send Stored Call Detail Records (cdrCrossRefID=6)
6. (positive acknowledgement)
7. Call Detail Record Report (cdrCrossRefID=6, calls c1, c2)
8. (positive acknowledgement)
9. (calls c3.....c9999 are cleared, CDR records stored)
10. Call Detail Records Notification (cdrCrossRefID=6, cdrReason=threshold reached)
11. (positive acknowledgement)
12. Send Stored Call Detail Records (cdrCrossRefID=6)
13. (positive acknowledgement)
14. Call Detail Record Report (cdrCrossRefID=6, calls c3 through c10)
15. (positive acknowledgement)
16. Call Detail Record Report (cdrCrossRefID=6, calls cxx through c999)
17. (positive acknowledgement)

6.5 Capabilities Exchange

The concept of capability exchange is one in which the switching function informs the computing function, through the service boundary, about the characteristics of its sub-domain in relationship to the operational model and feature definitions. This enables the computing function to use the services of the switching function based on its characteristics.

This exchange shall be performed before the computing function can control and/or observe any device in the switching sub-domain but not before the switching function has reported its system status (i.e., the System Status service).

There are two levels of capability exchange available to the computing function: *switching function* and *device specific capabilities*.

6.5.1 Switching Function Capabilities

The first level is the capabilities for the switching function. These capabilities represent the set of all capabilities within the switching function and are obtained using the Get Switching Function Capabilities service.

The list of devices that can be controlled and/or observed within the switching sub-domain can be obtained by using the Get Switching Function Devices service. This service causes the switching function to send one or more Switching Function Devices service(s) that contain the list of devices and optionally the device type, device name, and other information associated with each device.

6.5.2 Device Capabilities

Even though the Get Switching Function service gives the computing sub-domain most of the information to properly use the capabilities of the switching function, this information is sometimes not enough to totally understand the unique capabilities of a given device in relationship to the operational model or feature. Thus, another level of capability exchange exists which allows the computing function to obtain the specific capabilities associated with a given device or device configuration. This level of information is needed to better understand capabilities for an individual device. These specific device capabilities are obtained by using the following services (in any order):

- Get Physical Device Information service
- Get Logical Device Information service

The ability to use these services depends on whether or not the switching function actually supports these services. The computing function obtains this information from the Get Switching Function Capabilities service.

6.5.2.1 Physical Device Capabilities

This Get Physical Device Information service is used to obtain most of the capabilities and configuration information associated with the physical element of a device. To obtain the rest of the physical element characteristics, the computing function shall use the Get Button Information, Get Lamp Information, Get HookSwitch Status, Get Ringer Status, and Get Auditory Apparatus Information services to obtain all of the information associated with the device’s lamps and buttons. This information is used when controlling or observing the physical element of the device.

If the service indicates that the device has logical characteristics, the Get Logical Device Information service may be used for all associated logical elements.

If the service is used on a device that does not have any physical characteristics the service shall be rejected.

6.5.2.2 Logical Device Capabilities

The Get Logical Device Information service is used to obtain the capabilities and configuration information associated with the logical element of a device. This information is used when controlling or observing the logical element of the device.

If the service indicates that the device has physical characteristics, the Get Physical Device Information service may be used for all associated physical elements.

If the service is used on a device that does not have any logical characteristics the service shall be rejected.

6.5.3 Dynamic Feature Availability

A computing function can determine all possible CSTA services that can be applied to a connection given its state by using static information obtained through the Capability Exchange services. However, in certain implementations, there are situations where the set of services that can be applied to a connection varies depending upon how the connection got to a certain connection state and/or certain features active at a given device. In these cases, the static information provided in the Capability Exchange services may not reflect the actual set of services that are allowed.

If the Dynamic Feature Availability option is supported (as indicated through the Capability Exchange services), the actual set of CSTA services that can be applied to a connection at a given point is provided through the servicesPermitted parameter in every appropriate event.

Refer to 12.2.18, “ServicesPermitted”, on page 104 for a description on the use and restrictions of this parameter.

6.6 Switching Function Information Synchronization

Since the information obtained through the capability exchange services and call events may change after the information has been obtained, this Standard defines mechanisms that may be used to notify and provide the computing function with the updated information. This allows synchronization to be maintained with switching function information.

6.6.1 Switching Function Level Information

The following list describes how the computing function is notified when switching function level information has been changed.

- *switching function capabilities* - The Switching Function Capabilities Changed service is used to notify the computing function when information contained within the positive acknowledgement of the Get Switching Function Capabilities service has changed. The computing function shall issue the Get Switching Function Capabilities service to get the current information.
- *switching function devices* - The Switching Function Devices Changed service is used to notify the computing function when information contained within the Switching Function Devices service has changed. The computing function shall issue the Get Switching Function Devices service to get the current information.

6.6.2 Device Level Information

The following list describes how the computing function is notified when device level information has been changed.

- *device capabilities* - The Device Capabilities Changed event is used to notify the computing function when information contained within the positive acknowledgement of the Get Physical Device Information and/or Get Logical Device Information services has changed. The computing function shall then issue the appropriate Get Physical Device Information and/or Get Logical Device information services to get the current information.

6.6.3 Call Level Information

The following list describes how the computing function is notified when call level information has been changed.

- *account information and authorisation code* - The accountInfo and the authorisationCode parameters in the Call Control events represents the current account information and authorization code. In the situation where this information has changed independently of call activity (manual entry or via the Associate Data service, for example), the Call Information event is used to notify the computing function of the updated value.
- *calling device* - The callingDevice parameter in the Call Control events represents the calling device associated with the call. In the situation where this information was originally unknown and has now become available, the Call Information event is used to notify the computing function of the updated value.
- *dynamic feature availability* - The servicesPermitted parameter in the Call Control events represents the set of services that can be applied to a connection. In the situation where the servicesPermitted parameter changes due to another call's connection changing state, the Call Information event is used to notify the computing function of the updated capabilities for the connection that did not change state.
- *User Data and Correlator Data* - The userData and the correlatorData parameters in the Call Control events represents the current values of User Data and Correlator Data. In the situation where this information has changed independently of call activity (Send User Information or Associate Data services, for example), the Call Information event is used to notify the computing function of the updated values.

6.7 Status Reporting Services

Note that this section describes the Status Reporting services between the Switching Function and the Computing Function.

6.7.1 System Status

System Status services provide a way for the computing function and switching function to exchange information about the overall status of the system within each function. For each service boundary in a CSTA environment, the computing function and switching function on each side maintain a status attribute. System status services are bi-directional, enabling the computing function to report its status to the switching function, or to request the status of the switching function, and vice-versa.

6.7.1.1 System Status Registration

Before the computing function can receive any system status service requests, it may be required to register with the switching function for system status services using the System Status Register service. The positive acknowledgement to this service contains the system status register identifier (sysStatRegisterID) that the computing function uses to identify service requests that arrive for this registration.

If the switching function supports the System Status Registration services, then the computing function shall use the System Status Register service to register for system status services before it can receive any system status service requests. The first (mandatory) System Status service request from the switching function issued during an initialization sequence is an exception to this rule, however. If the switching function does not support the System Status Registration services, then the computing function may receive system status service requests at any time. The capabilities exchange services can be used to determine if the switching function supports the System Status Registration services.

The type of system status service requests that apply to the registration can be chosen by the computing function when it issues the System Status Register service request. A status filter can also be specified such that only the status's of interest to the computing function will be reported by the switching function (i.e., if the bit for a status is set in the status filter, then that status is not reported). This filter can be changed using the Change System Status Filter at any time while the registration is active.

A system status registration can be cancelled using the System Status Register Cancel service. Once the switching function sends a positive acknowledgement to this service, it will no longer send system status service requests to the computing function. Additionally, the switching function can cancel a system status registration at any time by sending the computing function a System Status Register Abort service request.

While the system status services themselves are bi-directional, the System Status Registration services are not. These services are only issued by the computing function. The switching function does not register with the computing function for system status services. The switching function is considered to be (implicitly) registered to receive system status service requests from the computing function at any time.

6.7.1.2 System Status Services

There are two System Status Services: System Status and Request System Status. The first service is used by the requesting function to report its status to the function receiving the service request. The second service is used by the requesting function to request (i.e., query) the status of the responding function.

The computing function can determine if the switching function uses the System Status service for periodic status reporting (i.e., heartbeats) using the capabilities exchange services. The Get Switching Function Capabilities service positive acknowledgement defines a parameter (systemStatusTimer) that is used to indicate whether periodic status reporting is used and if so, how often the computing function should expect the reports. The recovery action to be taken by the computing function in the event of a loss of heartbeats is implementation specific.

All System Status services use the following values to indicate system status:

- *Initializing* - The system is re-initializing or restarting. This status indicates that the system is temporarily unable to respond to any service requests. If provided, this status message shall be followed by an Enable status message that indicates that the initialization process is completed.
- *Enabled* - Request and responses are enabled, usually after a disruption or restart. This status indication shall be sent after an Initializing status indicator has been sent and may be sent under other conditions. This status indicates that there are no outstanding monitor requests.
- *Normal* - This value can be sent at any time to indicate that the status is normal. This status has no effect on other services.
- *Messages Lost* - This status indicates that a service request, response, or event report may have been lost.
- *Disabled* - This cause value indicates that active Monitor Start monitor requests have been disabled. Other requests and responses may also be disabled, but, unlike monitors, reject responses are provided for those.

- *Partially Disabled* - Some of the objects in the system can not be reached. Existing monitors on these objects will not provide events and computer requests targeting these objects will be rejected. This cause indicates to the receiving function that a degradation of service level may occur but not complete system disability. Automatic or manual actions may be taken to remedy the parts disabled.
- *Overload Imminent* - The system is about to reach an overload condition. The client should shed load to remedy the situation.
- *Overload Reached* - The system has reached an overload condition and may take action to shed load. The server may then take action to decrease message traffic. This may include stopping existing monitors or rejecting any new requests sent by the client.
- *Overload Relieved* - The system has determined that the overload condition has passed and normal application operation may resume.

Each system status service request may contain a system status registration identifier (sysStatRegisterID) to identify the associated system status registration (when system status registration is supported by the switching function). A system status service request from the computing function should never contain a system status registration identifier.

6.7.2 Monitoring

To track call control and other activity, and to receive notification of all changes in the switching Function, the computing function uses a feature called monitoring. By starting a monitor, the computing function indicates that it wants to be notified of specific changes that occur in the objects (call and device) and device attributes of a switching function. Examples include:

- Notification that a call has arrived at a device.
- Notification that a call has been answered.
- Notification that a feature such as “Forwarding” or “Do Not Disturb” has changed at a device.

Once a monitor is established, the switching function notifies the computing function of relevant activity by sending messages called *event reports*, or simply *events*.

For example, in the area of Call Control, events report the state transitions through which connections pass. In this way a computing function is able to determine what services are applicable to a given connection. For example, the Delivered event indicates when a connection state transits to the Alerting state.

The event categories are as follows:

- *Call Control* - These events report changes to information related to calls.
- *Call Associated* - These events report changes to information related to calls.
- *Media Stream* - These events report changes associated with attachment of a call to a media device.
- *Physical Device* - These events report changes to the components of a device's physical element.
- *Logical Device* - These events report changes to feature settings associated with a device's logical element(s).
- *Voice Unit* - These events report changes to Voice Unit messages.
- *Maintenance* - These events report changes regarding maintenance.
- *Private* - These events are switching function specific.

6.7.2.1 Starting and Stopping a Monitor

The Monitor Start service is used to establish a monitor. The computing function indicates the monitor object that it is interested in observing, the type of monitoring, the type of calls to monitor, and the list of events that it is interested in.

Once the Monitor Start service request has been validated by the switching function, the switching function provides a positive acknowledgement that includes a Monitor Cross Reference Identifier that uniquely identifies the monitor. The switching function also provides this identifier as a parameter in all events associated with this monitor. The computing function can use this identifier to correlate events to the particular Monitor Start service that established the monitor. (This identifier is also used in the Monitor Stop and Change Monitor Filter services.)

The Monitor Stop service is used to stop an established monitor. When a Monitor Stop service has been sent by the computing function, the switching function stops the monitor, releases the Monitor Cross Reference Identifier, and no longer provides events to the computing function.

The Monitor Stop service may also be sent from the switching function to the computing function when the switching function stops an existing monitor. This occurs when the monitor object is a call-object (Table 6-7), or when the switching function shall terminate a monitor due to load conditions, for example.

Refer to 15.1 beginning on page 166 for a complete description of the Monitor Start and Monitor Stop services.

6.7.2.2 Monitor Objects

The computing function indicates what it wants to monitor by specifying a monitor object parameter in the Monitor Start service request. There are two possible monitor objects: *call-object* and *device-object*.

The following table describes the monitor objects.

Table 6-7 Monitor Objects

Monitor Object	Description
call-object	Place a monitor on an existing call/connection. Only the specific call is monitored. A Monitor Stop service is sent by the switching function to indicate when the existing call is no longer monitored.
device-object	Place a monitor at the specified device.

6.7.2.3 Monitor Types

The computing function also indicates a monitor type when starting a monitor. There are two types of monitoring: *call-type* and *device-type*.

The following table describes the possible monitor types and their meanings.

Table 6-8 Monitor Types

Monitor Type	Description
call-type	The call continues to be monitored as long as it remains in the switching sub-domain. For example, if a call that is being call-type monitored is transferred to another device in the switching sub-domain, the call will continue to be monitored. The computing function receives events for all devices in the call until the call ceases to exist or until it leaves the switching sub-domain. The Diverted event is an exception. The switching function (as indicated through the capabilities exchange services) may or may not be providing Diverted events to all devices in a call. For call-type monitors: • When a device ceases to participate in a call, and the call is transferred or forwarded to another device, subsequent events at the new device are reported. The Monitor Cross Reference Identifier used in events at the new device will be the same one used before the call was forwarded or transferred. • If a call is being monitored using a call-type monitor and one of the devices consults to another device (i.e. a new call is created), then the computing function will not see events for the secondary call (new consultation call) until either the primary call is transferred to the consulted device, or until the two calls are conferenced together. • A call that is being monitored may have a new Call Identifier assigned to it after a conference or transfer. The switching function reports the new Call Identifier in a Conferenced or Transferred event.
device-type	The call does *not* continue to be monitored after the call leaves the device.

6.7.2.4 Relationship of Monitor Objects and Monitor Types

Monitor objects and monitor types are independent. A *monitor object* describes what the monitor is being placed on, while the *monitor type* describes if a call continues to be monitored after it leaves a device.

The following table describes the possible combinations of monitor objects and monitor types and what the resulting combinations represent.

Table 6-9 Monitor Object/Monitor Type Combination

Monitor Object	Monitor Type	Usage
call-object	call-type	This combination is used to track an existing call, for as long as that call remains in the switching sub-domain. Monitor Stop service is sent by the switching function when the call ceases to exist in the switching sub-domain to indicate that the monitor is stopped and the associated Monitor Cross Reference Identifier is no longer valid.
call-object	device-type	This combination is used to track an existing call, while that call remains at the specified device. Monitor Stop service is sent by the switching function when the call leaves the device to indicate that the monitor is stopped and the associated Monitor Cross Reference Identifier is no longer valid.
device-object	call-type	This combination is used to track all calls that arrive (or are present) at the device, for as long as the calls remain in the switching sub-domain. The specified device object can be thought of as a trigger device where all calls that become involved with this device become monitored as long as the call remains in the switching sub-domain. Monitor Stop service is *not* sent by the switching function when a call ceases to exist or moves away from the monitored device, since the monitor is still in place at the device.
device-object	device-type	This combination is used to track all calls that arrive (or are present) at the device, for as long as the calls remains at the device. Monitor Stop service is *not* sent by the switching function when a call ceases to exist or moves away from the monitored device, since the monitor is still in place at the device.

6.7.2.5 Monitoring in Relationship with Media Class

The computing function can also indicate the media class (voice, digital data, etc.) of calls to be monitored when starting a monitor.

Refer to the media class component of 12.2.15, “MediaCallCharacteristics”, on page 101 for the complete set of possible values. The media class is independent of the monitor object and monitor type.

6.7.2.6 Reporting Connection State Changes

Once a call is monitored (irrespective of monitor type or monitor object), all connection state changes that are known by the switching function for that call are reported to the computing function (subject to the Monitor Filter—refer to 6.7.2.7).

For example, if device A is being monitored (with a device-type monitor) and a call is placed to device B (no monitor on B), then any connection state changes for either device A or B (such as when B answers the call) will be reported through device A’s monitor.

Monitoring is only guaranteed for devices in the switching sub-domain. Activity related to devices outside the switching sub-domain may be only partially available or completely unreported.

6.7.2.7 Monitor Filtering

The computing function can request that a set of events be filtered out (not sent) by the switching function. This information is specified in the monitorFilter parameter in the Monitor Start service request.

The monitorFilter parameter contains a list of filters that are grouped together into the following categories:

- Call Control events

- Call Associated events
- Media Stream events
- Physical Device Feature events
- Logical Device Feature events
- Maintenance events
- Voice Unit events
- Private events

The switching function indicates the actual list of events that will be sent by returning the monitorFilter parameter in the positive acknowledgement to the Monitor Start and Change Monitor Filter services.

The computing function can request that the filtered list of events for an existing monitor be changed by issuing the Change Monitor Filter service.

Some categories of events are not provided for call-type monitors. The capability exchange services indicate the categories of events that are supported by the switching function for call-type monitors.

6.7.3 Snapshot Services

Snapshot services are used by the computing function to determine information about a call or a device. These services may be used at any time, independently of, or in combination with existing monitors. For example, a computing function may snapshot a device prior to starting a monitor on the device, in order to obtain information on existing calls at the device.

6.8 Additional Services, Features & Behaviour

This section specifies standardized switching function features affecting calls at a given device that do not have an explicit service request associated with the invocation of the feature. These features are usually configured within the switching function or have a service request which sets up certain conditions at a device that causes a particular behaviour with respect to calls at the device. As a result, these features are only reflected through an event sequence from the switching function. The following sections explain these features and the event sequences associated with them.

6.8.1 Forwarding

The forwarding feature is a trigger at a device that will redirect incoming calls to another device based on a specific condition. The following are the types of conditions that would trigger the redirection, or forwarding of the incoming call:

1. *Immediate* - This condition indicates that if a call arrives at a device, it is immediately redirected to another device.
2. *Busy* - This condition indicates that if a call arrives at a device, and the device is busy with another call, then the incoming call will be redirected to another device.
3. *No Answer* - This condition indicates that if a call arrives at a device, and the call is not answered within a certain number of rings or within a specific amount of time, then the incoming call will be redirected to another device.
4. *Do Not Disturb (DND)* - This condition indicates that if a call arrives at a device, and the device has the Do Not Disturb feature active at the device, then the incoming call will be redirected to another device. Note that the Do Not Disturb feature does not necessarily imply that incoming calls are forwarded.
5. *Type of Call Origination* - This condition indicates that if a call arrives at a device, and the originating device is a specific class (i.e., external, such as a device that is outside the switching sub-domain, or internal, such as a device that is within the switching sub-domain), then the incoming call will be redirected to another device. This condition can be used in combination with the others to create a compound condition. For example, if busy with another call and the calling device is outside the switching sub-domain, then redirect the call to another device.

Switching functions may support one or both of the following levels of forwarding settings:

- switching function default settings
- User specified settings

Switching function default settings are a single set of forwarding-type/forward-destination combinations that can be activated and deactivated as a set. The set includes all of the CSTA forwarding-types defined and the forward-destinations for each type. Activation, deactivation, or changes to the forward-destinations are not normally possible by users.

User specified settings are individual forwarding-type/forward-destination combinations that can be activated or deactivated one at a time. User specified settings supersede switching function default settings during activation, deactivation, and when forwarding occurs.

A switching function that supports switching function default settings may also support user specified settings. Switching function default settings are used for forwarding to a standard destination such as voice mail or an attendant. User specified settings may be used to override the default settings to forward calls temporarily to another office, for example.

A user specified forwarding type supersedes the same switching function default forwarding type when forwarding occurs. For example, a user specified type of “No Answer” and its corresponding forward destination supersede a switching function default type of “No Answer”. Note that this rule may not apply to types that are not alike. For example, a user specified type of “No Answer” (a delayed type of forwarding) does not supersede a switching function default type of “Immediate”, although a user specified type of “Immediate” does supersede a switching function default type of “No Answer” (since “No Answer” is a delayed type of forwarding).

The forwarding feature has service requests and events to control and observe the activation and deactivation of the forwarding triggers at the device (i.e., Get Forward, Set Forward, Forwarding). These service requests and events are documented in Clause 22, “Logical Device Features”, beginning on page 431, and do not actually forward the incoming call when it arrives at the device, but instead sets up the trigger to cause the switching function to perform the redirecting of the call. The computing function should use the capabilities exchange services to determine which of these services and events the switching function supports.

The computing function should use the capabilities exchange services to determine which of the following levels of forwarding settings are supported by the switching function:

- Switching function default settings (set of forwarding types and forward destinations).
- User specified settings.
    - Default forwarding type.
    - Default forward destination.

Switching function default settings may be activated or deactivated manually at the device, or by providing neither the forwarding type nor forward destination (forward DN) in Set Forward service requests.

User specified settings may be activated or deactivated manually at the device or by providing the forwarding type and/or the forward destination (forward DN) in the Set Forward service request. If the forwarding type is not specified and the forward destination is specified, the switching function uses a default forwarding type. Likewise, if the forwarding type is specified and the forward destination is not specified, the switching function uses a default forward destination.

The computing function is informed that default settings are being activated in the Get Forward positive acknowledgement and the Forwarding event.

When the call is redirected as a result of the (Immediate) forwarding feature, there are two basic event sequence models to indicate that the call has been forwarded. The following are the event sequence model definitions (Note that the computing function should use the capabilities exchange services to determine which of model or models that the switching function supports.):

1. *Forwarding Is Triggered before the Call Is Delivered to the Device* - There is basically no event sequence associated with this condition. The only characteristic associated with this event sequence is:

- The first event associated with the delivery of the call to the new device will have an appropriate forwarding event cause. If the RedirectionDeviceID parameter is available in this event, it will be provided based upon the definition of the Call Control event and 12.3.24, “RedirectionDeviceID”, on page 115. Refer to 6.8.6, “Tracking a Diverted Call”, on page 57 for additional information on event sequences for forwarded calls.

If the call is forwarded multiple times under the same condition (e.g., forwarded from device 1 to device 2 which is forwarded to device 3), then the information indicating that the call was forwarded will only be the information from the last device the call was forwarded from (e.g., device 2). As a result, the computing function will only see that the call has been forwarded one time.

2. *Forwarding Is Triggered after the Call Is Delivered to the Device* - The event sequence is a Diverted event followed by the first event associated with the delivery of the call to the new device. The characteristics associated with this event sequence are:

   - Depending on the capabilities of the switching function, a Delivered event may or may not flow as a result of presenting the to-be-forwarded call to the device from which it will be diverted.
   - The Diverted event will have an appropriate forwarding event cause. (Note that he reporting of this event is dependent on the capabilities of the switching function.)
   - The first event associated with the delivery of the call to the new device will have an appropriate forwarding event cause. If the RedirectionDeviceID parameter is available in this event, it will be provided based upon the definition of the Call Control event and 12.3.24, “RedirectionDeviceID”, on page 115. Refer to 6.8.6, “Tracking a Diverted Call”, on page 57 for additional information on event sequences for forwarded calls.

   If the call is forwarded multiple times under the same condition (e.g., forwarded from device 1 to device 2 which is forwarded to device 3), then the information indicating that the call was forwarded will be available each time the call is forward (e.g., device 1,
   device 2). This is possible because the call is actually delivered to the device before it is forward to another.

If the call is forward multiple times with a mixture of forwarding conditions (i.e., event sequence types), then the information indicating that the call was forwarded will be a mixture of the event sequences depending on the order of the forwarding conditions.

6.8.2 Connection Failure

The information indicating connection failure can be reported through several different event sequences. The computing function should use the capabilities exchange services to determine which of these services and events the switching function supports. The following are the possible event sequences associated with connection failure:

1. *Negative Acknowledgement* - When the switching function supports service requests that perform connection creation process and the switching function detects a failure, the negative acknowledgement can be used to indicate the failure to complete the connection. The following are the service requests associated with connection creation process:

   - Consultation Call
   - Deflect Call
   - Dial Digits
   - Join Call
   - Make Call
   - Make Predictive Call
   - Pickup Call
   - Single Step Conference Call
   - Single Step Transfer Call

If the switching function uses the negative acknowledgement to indicate the connection failure, then the appropriate error code will be used to indicate the particular failure.

2. *Support of the Failed Event with an Associated Failed Connection* - When the switching function detects a connection failure, it places that connection into the failed state. This indicates that the call control services which can be performed with respect to the connection are limited. The following is the list of call control services that are applicable:

   - Clear Call
   - Clear Connection
   - Call Back Call-Related
   - Call Back Message Call-Related
   - Camp On Call
   - Deflect Call
   - Intrude Call

   When a connection enters the Failed state, the event sequence provided is a Failed event. The characteristics associated with this event sequence are:

   - The Failed event will have an appropriate failure event cause.
   - The failedConnection parameter in the Failed event will contain a “complete” Connection Identifier (i.e., a Connection Identifier that has both a Device Identifier and Call Identifier)
   - The Failed event will be reported to all active device-type monitors associated with the call, as well as all call monitors associated with the call.

3. *Support of the Failed Event without an Associated Failed Connection* - This case is similar to the “Support of the Failed Event with an Associated Failed Connection” state (case 2). The difference is that when the switching function detects a connection failure, it does not create a connection for the failed device but instead indicates to the computing function that call control services, with respect to the connection, are limited. The following is the list of call control services that are applicable to the connection in the call under these conditions:

   - Clear Call
   - Clear Connection
   - Call Back Call-Related
   - Call Back Message Call-Related
   - Camp On Call
   - Deflect Call
   - Intrude Call

   When the failure is detected, the event sequence provided is a Failed event. The characteristics associated with this event sequence are:

   - The Failed event will have an appropriate failure event cause.
   - The failedConnection parameter in the Failed event will contain a “Call ID only” Connection Identifier. This indicates that there is not a valid connection for the failed device in the call but that the appropriate call control service can be performed (i.e., Call Back Call-Related, Intrude Call, etc.) on the call.
   - The Failed event will only be reported to the active device and call monitors associated with the devices that where in the call prior to the failure (i.e., if a device-type monitor was on the failed device, then the event sequence is not reported).

If the Camp On Call or Intrude Call service request is performed on the call, then the connection associated with the failed device will be created (i.e., a valid connection).

4. *Support of the Failed Event with an Associated Failed Connection, not reported via monitors on the failing device* - This case is similar to the "Support of the Failed Event with an Associated Failed Connection" state (case 2). The difference is for which monitors the Failed event is being sent: The Failed event will only be reported to the active device and call monitors associated with the devices that were in the call prior to the failure (i.e. if a device-type monitor was on the failed device, then the event sequence is not reported). Apart from this, all aspects from case 2 apply also to this case.

6.8.3 Recall

The Recall feature is a trigger that is associated with a call after a specific call control feature has been executed. When this feature is executed, it redirects or presents the call either back to the device on who's behalf the call control feature was executed or to a switching function administrated destination associated with the specific call control feature. There are several types of call control services which can have this feature associated with them. For example:

- Hold Call
- Transfer Call
- Single Step Transfer Call
- Deflect Call
- Park Call

The event sequence associated with this feature is the Diverted event (only if the device to whom the call is being redirected is not already in the call) and the first event associated with the delivery of the call to the new device or the device that performed the call control feature. The characteristics for this event sequence are:

- The Diverted event will have an appropriate recall event cause. This event is only reported when the device to whom the call is being redirected is not already in the call (i.e., a recall to a connection that is already in the call). (Note that the reporting of this event is dependent on the capabilities of the switching function.)
- The first event associated with the delivery of the call to the new device (i.e. Delivered), or the device that performed the call control feature will have an appropriate recall event cause (in either the Delivered, Held, Queued, etc.). If the lastRedirectionDevice parameter is available in this event, and the call was actually redirected to another device outside the current call, then it will be provided based upon the definition for this parameter. (Refer to the definition of the Call Control events and lastRedirectionDevice parameter for more details.) If the callingDevice parameter is available in this event, it may contain the same callingDevice information prior to the recall. This means that if the calling device is the Subject Device of the event, then the information in the callingDevice and corresponding SubjectDeviceID (e.g., Delivered event SubjectDeviceID = alertingDevice) parameters may be the same. If the switching function does not retain this information with the call, then the callingDevice parameter will contain a value of "Not Known".

6.8.4 Call Back

The Call Back feature is a trigger which is set up within the switching function. The trigger is used to initiate a call between a particular pair of devices. The pair of devices is comprised of a calling device (i.e., the device on who's behalf the trigger is setup) and the called device (i.e., the device whom the calling device wants to initiate a call to when certain conditions associated with the called device are met). The type of conditions associated with the trigger is switching function specific. A common type of condition is the called device is no longer actively involved in a call(s). The trigger is activated for the calling device by either the Call Back Call-Related or Call Back Non-Call-Related services. The trigger is deactivated by one of the following: successful execution of the trigger, the Cancel Call Back service, or a switching function specific timeout period. Once the trigger is activated, the switching function waits for the particular condition associated with the Call Back feature to be met. Once met, the switching function initiates a call on behalf of the calling device to the called device. This is done by first prompting the calling device (if supported by the device) and then initiating the call to the called device.

The event sequence associated with execution of the Call Back trigger is the Service Initiated event (only if the calling device is prompted), Originated event and the events associated with the called device's involvement in the call. The characteristic for this event sequence is that both the Service Initiated (if supported) and Originated events will have an event cause of Call Back.

6.8.5 External Calls

A call is considered to be external when there is at least one device in the call that is outside the switching sub-domain. For more details on how the switching function represents these devices within the switching sub-domain, refer to 6.1.3.4.2, "Network Interface Device Category", on page 20. The activities associated with external calls are broken down into two categories:

- *Incoming Calls* - A call is being initiated from a device outside the switching sub-domain.
- *Outgoing Calls* - A device inside the switching sub-domain is adding or initiating a call to a device outside the switching sub-domain.

These categories are represented by different event sequences. The characteristics associated with these event sequences are:

- Incoming Calls
    - A Service Initiated event is generated for the network interface device when the network interface device is allocated (e.g., seized) for the external incoming call. The initiatingDevice parameter will contain the information on which network interface device is being used for the call. Note that this event is only generated if the network interface device is monitored.
    - The Digits Dialled event is generated for the network interface device when a portion of the dialling sequence has been received over the network interface device. Note that this event is only generated if the network interface device is monitored.
    - The Originated event is generated for the network interface device when the external incoming call has originated from the network interface device. The NIDDevice parameter will contain the information on which network interface device is being used for the call. Note that this event is only generated if the network interface device is monitored.
    - In all subsequent events (independent of whether or not the network interface device is observable), this incoming call is distinguished from an internal incoming call by the presences of the associatedCallingDevice (i.e., containing either a Device Identifier or a value of "Not Known"). The information in the associatedCallingDevice parameter is first associated with the call when it enters the switching sub-domain (i.e., Service Initiated or Originated events) and will be present until the calling device leaves the call.
- Outgoing Calls
    - When initiating a connection to a device outside the switching sub-domain and the switching function is associating the Network Interface Device with the outside device, a Network Reached event is reported. In addition, the Network Reached event is the first event that indicates the call is an external outgoing call. Subsequent events (if available) will contain the associatedCalledDevice parameter (i.e., the value from the networkInterfaceUsed parameter of the Network Reached event) and will be present until the call is cleared. In addition, until the Network Reached event is generated, the call is consider to be an internal call.
    - The event sequence after the Network Reached event that is associated with the device outside the switching sub-domain may be limited but the events that are reported will contain one of the event causes documented in the event definition. If the Network Reached event contained the networkCapability parameter, future Network Capabilities Changed events may be provided indicating a change in the signalling capability of the network and ultimately the types of events that can be provided.

6.8.6 Tracking a Diverted Call

When observing a call or a device in a call, and the call diverts from a device in the call, the computing function shall use the Diverted event to track the progress of the call as a result of the redirection.

If the switching function does not provide the Diverted event for all devices in a call or for call-type monitors (as indicated through the capabilities exchange services), the computing function shall use parameters in the first event after the call has been diverted to properly track the progress of the call as a result of the redirection. The device identifiers are used to observe the movement of the call and the event cause is provided to indicate what caused the movement of the call. (Note that the call may have been diverted several times between the previous event (if one was generated) and the first event after the diversion. As a result, the computing function can only ascertain that either one or two redirections have occurred.)

6.8.7 Media Stream Access

The capability to control the information content within a call is called media stream access, or simply, media access. Media access is provided to a computing function through a media service. Common media service capabilities are play/record of voice and audio, automatic speech recognition, text to speech, fax, and data services.

6.8.7.1 Media Attachment Services

A computing function making use of both call control services and media services needs to establish sessions with both services, attach calls to the media service, and needs a way of associating the identifiers (e.g., Connection Identifiers, Media Stream Identifiers) used by the two services. This Standard defines a set of services, called media attachment services, that make these tasks significantly easier for the computing function.

6.8.7.2 Media Service Type

A particular media service, which is defined by its set of services and possibly its access methods (APIs, protocols, etc.) is identified for the purpose of the media attachment services by a unique media service type identifier. The mediaServiceType parameter is used to indicate which media service is to be (or has been) attached to or detached from a particular call or connection that the computing function is controlling and/or monitoring. In some instances, the media service version may also be provided, such that different versions of the media service can be identified by unique media service types.

Table 6-10 identifies and describes the set of media service types defined by this Standard:

Table 6-10 Media Service Types

Media Service Type	Media Stream ID Representation
CSTA Voice Unit	Refers to the connection identifier in the special resource sub-domain.
Data Modem	Refers to the address that is to be used to access the modem control and data stream.
Digital Data— Isochronous/IEEE 1394	Refers to the IEEE 1394 channel that is being used.
Digital Data—Isochronous/GeoPort	Refers to the GeoPort stream that is being used.
Digital Data— Isochronous/ATM	Refers to the ATM virtual channel/path identifier that is being used.
Digital Data— Isochronous/ISDN	Refers to the ISDN bearer channel that is being used.
Digital Data—API	Refers to the particular API’s digital data stream reference ID (e.g., Microsoft’s Winsock is socket identifier) to indicate which digital data stream is to be used.
ECTF S.100 Media Services Default	Refers to the ECTF S.100 Media Services CCR Resource ID (CCR ECTF ResourceID) to indicate which media stream channel to be used.
ECTF S.100 Media Services Application Service	Refers to the ECTF S.100 Media Services Application Service.
IVRScript1 through IVRScript10	Not defined; the attachment of the media stream channel to the vendor media server instance is vendor-specific.
Live Sound Capture—Analog	Refers to the analog jack that is being used.
Live Sound Transmit—Analog	Refers to the analog jack that is being used.
Live Sound Capture—IEEE 1394	Refers to the IEEE 1394 channel that is being used.

Table 6-10 Media Service Types (continued)

Media Service Type	Media Stream ID Representation
Live Sound Transmit—IEEE 1394	Refers to the IEEE 1394 channel that is being used.
Live Sound Capture and Transmit—GeoPort:	Refers to the GeoPort stream that is being used.
Live Sound Capture and Transmit—ATM	Refers to the ATM virtual channel that is being used.
Live Sound Capture and Transmit—ISDN	Refers to the ISDN bearer channel that is being used.
Sound Capture and Transmit—API	Refers to the particular API's sound stream reference ID (e.g., Microsoft's MCI's is MCI Device handle) to indicate which sound stream is to be used.
Sound Capture and Transmit—Rockwell ADPCM Packet	Refers to the address that is to be used to access the asynchronous stream.
Universal Serial Bus (USB)	Refers to the USB endpoint.
sfSpecific1 through sfSpecific10	Not defined; the attachment of the media stream channel to the vendor media server instance is vendor-specific.

6.8.7.3 Media Service Instance

A specific set of resources and/or functions that provide a particular media service (as identified by the media service type) are referred to as a media service instance. For example, a media service instance may be a specific media server, subsystem, or software that provides the given service. When the computing function attaches a call to a media service, it may request a particular instance of the service through the media service instance identifier. A media service instance may have associated with it zero or more media access devices that provide a means of physical connection between switching sub-domain resources and media service resources.

6.8.7.4 Media Access Device

A media access device is a device within the switching sub-domain used in establishing and modeling the physical connection of the media stream between switching sub-domain resources (i.e., devices in a call) and media service resources (i.e., media processing resources such as tone generators/detectors, speech recognition devices, text-to-speech converters, modems, etc.). During media service instance attachment, a media access device may be conferenced into a call, or the call may be transferred to the media access device from another device. During media detachment, the media access device is cleared from the call.

6.8.7.5 Media Stream ID

The media services that can be the target of the media attachment services are unlimited. That is, very little is assumed about the operational model of the media services themselves. The only requirement is that the media service defines some identifier, referred to here as the media stream identifier (i.e., mediaStreamID), that can be used by the computing function to reference the media stream associated with a connection.

The mediaStreamID is returned, if supported, as a parameter in the Media Attached event. The capability exchange services are used to determine if the switching function supports returning the mediaStreamID.

The format and meaning of the mediaStreamID is media service and implementation dependent, and is not defined here. The switching function only guarantees the mediaStreamID to be valid while the media service instance remains bound to the call. Once the media service instance is detached from the call, the validity of the mediaStreamID is media service dependent.

6.8.7.6 Service Operation

The media attachment services defined in this Standard consist of two services: Attach Media Service and Detach Media Service, and two events: Media Service Attached and Media Service Detached. A description of these services and events and their usage follows. For more information see Clause 19, "Media Attachment Services & Events", beginning on page 354.

A typical media enabled computing function will establish or accept a call, perform some media access functions associated with the call, and then clear the call. The first and third steps clearly require the call control services only, while the second involves coordination of both call control and media services. This second step can be further sub-divided into the following tasks:

1. A particular instance of the media service (e.g., specific media server or media subsystem providing the desired services) shall be chosen, either by the computing function or by the switching function on behalf of the computing function.
2. The computing function shall establish a session with the selected media service instance. The method of the session establishment is media service dependent (e.g., initialize with the media service API, send a message to the media service, establish a CSTA association with a SRF).
3. If the switching function supports returning a mediaStreamID to the computing function, an association between the switching function and the media service instance must be established. The way this association is realized is implementation dependent. The switching function and the media service instance require a means to attach the media stream channel of a connection to a media stream channel of the media service instance. This is referred to as a connection mode. Connection modes are enumerated in the specification of Media Attachment services. They fall into two categories:
    a. Explicit representation by adding a media access device into the call via a call control service (e.g. Conference Call, Transfer Call, Deflect Call, Divert Call, Directed Pickup Call). The media service instance is bound to the new connection. In general, the media access device behaves the same as any other call control device, although the services that can be applied to the device or a connection associated with the device may be restricted by some implementations. Connection modes in this category best suit, but are not limited to, configurations where call control resources and media resources consist of distinct, non-integrated hardware components. An example of such a configuration is one in which the switching resources reside in a PBX and the media service resources reside in a VRU.
    b. Implicit representation by an existing connection in the call (referred to as the direct connectionMode). A media access device is not added to the call. Instead, the media access device is already attached to an existing connection in the call. Connection modes in this category best suit, but are not limited to, configurations where call control resources and media resources consist of common or tightly integrated hardware components. An example of such a configuration is one in which the switching and media service resources are provided by an integrated telephony and media processing board in a PC. Another example of such a configuration is one in which an external voice response unit makes an outbound call to a media access device in a switching sub-domain, thus attaching its media services to the device's connection.
4. The associated mediaStreamID assigned by the media service instance may be returned by the switching function to the computing function.
5. The computing function can access the media service instance using the supplied mediaStreamID. At this point, the computing function may mix the use of call control services and services provided by the media service instance as needed.
6. When the computing function has finished its use of the media services, it shall unbind the media service instance from the call. The unbinding of the media service instance consists of releasing the attachment established in step 3, and, if applicable, removing the associated media access device from the call. As far as the switching function is concerned, the returned mediaStreamID is also invalidated once the media service instance is detached from the call.
7. The computing function may or may not close its session with the media service instance. The switching function may or may not release its association with the media service instance.

Several of these tasks are handled on behalf of the computing function through the use of the Media Attachment Services. Steps 1, 3, and 4 are provided by the Attach Media Service, and step 6 is provided by the Detach Media service.

The Attach Media Service attaches an existing call to a media service instance. The Attach Media Service service request provides a mediaServiceInstanceID parameter that selects a particular media service instance associated with the switching function when multiple choices exist. If the switching function has multiple choices and the parameter is not supplied, then the switching function shall select which media service instance is to be used.

The Attach Media Service, depending upon the connectionMode parameter supplied in the service request, initiates a connection to an available media access device associated with the service instance (for the explicit connection category). At the completion of a successful service invocation a Media Attached event is reported on monitors associated with the specified call or device in addition to any other events that flow as a result of making the media access device connection. The Media Attached event may contain the associated mediaStreamID for accessing the media service instance. The Attach Media Service service positive acknowledgement and Media Attached event may be correlated using the CSTA Connection and Device Identifiers associated with the chosen media access device or existing connection that was bound in the call.

A Media Attached event may flow any time a connection is bound to a media service instance, even if the binding was not the result of an Attach Media Service service request. This provides for automatic attachment and media service type determination by the switching function (e.g., situation where calls are automatically directed to a media access device by the switching function or automatically bound to an existing connection when a call arrives). When the computing function receives this event, it may open a session with the associated media service instance and immediately begin accessing the media service instance using the mediaStreamID provided in the event. The session establishment with the media service and mediaStreamID usage are media service dependent and outside the scope of this Standard.

The Detach Media Service service request undoes the actions of Attach Media Service service. It unbinds the media service instance from the specified call or connection and breaks the physical connection of the media stream between the switching sub-domain and media service instance. Any associated media access device involvement in the call is also cleared. The connection to the device is cleared and reported through normal call control events (i.e., Connection Cleared events). A Media Detached event report is used to indicate that media service instances have been detached from a call or connection. The switching function itself may initiate a media detachment. In this case, the detachment is reported using the same events as if the computing function had initiated the Detach Media Service service request (e.g., Media Detached and possibly Connection Cleared events).

6.8.7.7 Related Services

A number of other call control services return information related to the media access capability. The Snapshot Call service returns a list of media service types, media service versions, media service instance IDs, and media stream IDs associated with each connection in a call.

The Get Switching Function Capabilities service returns, for the entire switching function, a list of supported media service types, media service versions, media service instance IDs, and whether or not the media stream ID is supported for this combination of media attributes, as well as the supported connection modes for each combination.

Finally, the Get Logical Device Information service returns, for a selected device, a list of supported media service types, media service versions, media service instance IDs, and whether or not the media stream ID is supported for this combination of media attributes, as well as the supported connection modes for each combination.

6.8.8 Routeing Services

A switching function uses Routeing services when it needs the computing function to supply call destinations. This may be on a call-by-call basis or it may be non-call related. The computing function can use internal databases together with call information to determine a destination, or route, for each call. For example, the computing function might use the caller's number and information in a database to route incoming calls.

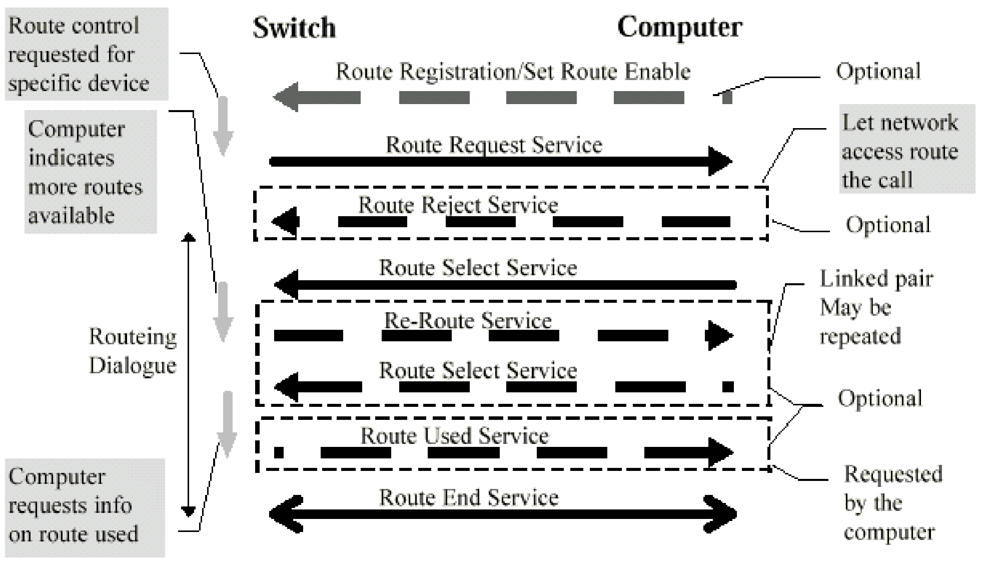
A switching function may support Routeing services for any type of call (e.g., external outgoing, external incoming, intra-switching sub-domain). Routeing services may require that the switching function be configured to direct calls to a device known as a *routeing device*. This device shall be addressable (i.e., visible within the switching sub-domain) with respect to Routeing services but may or may not be addressable with respect to other services (e.g., Call Control, Monitoring).

The routeing device may be a virtual device used only for routeing and thus may not be monitorable. The way a particular virtual routeing type device is used by a switching sub-domain is specific to each implementation. Examples include:

- a routeing device could be used to route all outgoing external calls from all devices within a given switching sub-domain
- a routeing device could be used to route all incoming external calls independent of the network interface device being used
- a routeing device could be used to route all calls that are considered to be priority calls independent of their origin.

A switching function implementation will implement as many routeing devices as it requires in order to reflect the different routeing processes it supports.

Figure 6-25 Overview of a Routeing Dialogue


Route control requested for specific device
Computer indicates more routes available
Routeing Dialogue
Computer requests info on route used
Switch
Computer
Route Registration/Set Route Enable
Optional
Route Request Service
Let network access route the call
Route Reject Service
Optional
Route Select Service
Linked pair May be repeated
Re-Route Service
Route Select Service
Optional
Route Used Service
Requested by the computer
Route End Service

Routeing services are used within a sequential “routeing dialogue” such as that represented in Figure 6-25. (Note that none of the routeing services return positive acknowledgements. Negative acknowledgements, though provided by routeing services when applicable, are not shown in the figure.)

A routeing dialogue is typically initiated by the switching function when a call is directed to a device and particular conditions are met for that call at that device. The conditions at a device under which the switching function may initiate a routeing dialogue are determined by its Route Mode and the Route Registration Service. Through these mechanisms the computing function may specify to the switching function that when calls encounter a particular device, the computing function should be consulted for a proposed route.

Routeing services are linked within a routeing dialogue by the routeing cross reference identifier (routeingCrossRefID). A routeing cross reference identifier is provided by the Switching function as part of the Route Request Service used to initiate a routeing dialogue. This routeing cross reference identifier is quoted by each subsequent invocation of a routeing service in the routeing dialogue.

Route Requests generated by the switching function may be call-related or non-call related.

6.8.8.1 Route Registration and Route Mode

Table 6-11 below specifies the conditions that must be satisfied before a given switching function initiates a routeing dialogue for a given routeing device by generating a Route Request. The switching function’s behaviour is governed by its support for the routeing registration services and support for the Route Mode attribute. Registration has no affect on routeMode, and enabling/disabling routeMode has no affect on registration. In order to use routeing services for a given routeing device, a computing function must satisfy the conditions specified in Table 6-11 by invoking the appropriate services.

Table 6-11 Routeing Behaviour

	Registration Not Supported	**Registration Supported**
Route Mode Supported	• Registration not required • RouteMode must be enabled • Switching Function must initiate routeing dialogue if a call of any media class arrives	• Registration required • RouteMode must be enabled • Switching Function must initiate routeing dialogue if a call of any media class arrives that matches the media class requested
Route Mode Not Supported	• RouteMode implicitly enabled • Registration not required • Switching Function may initiate routeing dialogue at its discretion	• RouteMode implicitly enabled • Registration required (specific or all) • Switching Function must initiate routeing dialogue if a call arrives that matches the media class requested

The positive acknowledgement to the Route Register service contains the route register request identifier (routeRegisterReqID) that the computing function uses to identify service requests that arrive for this registration.

If the switching function supports the Route Registration services, then the computing function shall use these services to register as a routeing server before it can route calls. If the switching function does not support the Route Registration services, then the computing function may receive route service requests for any routeing device at any time.

The computing function may either register as the routeing server for a specific routeing device or, if supported by the switching function, as a routeing server for all routeing devices within the switching sub-domain.

A route registration can be cancelled using the Route Register Cancel service. Once this service is positively acknowledged, the switching function will no longer send route service requests to the computing function. Additionally, the switching function can cancel a route registration at any time by sending the computing function a Route Register Abort service request.

Routeing services for a particular device may be suspended without cancelling route registration by disabling its Route Mode. This does not effect route registration and route requests for the given device will resume when its Route Mode is enabled.

The capabilities exchange services can be used to determine if the switching function supports the Route Registration services and if so, if the capability to register for all routeing devices is supported. The capabilities exchange services can also be used to determine if the switching function supports the Route Mode attribute.

6.8.8.2 Call Routeing

An example of a routeing process may involve the following sequence of steps:

1. The switching function receives a call at the routeing device. The routeing device may be any device within the switching sub-domain.
2. When the call arrives at the routeing device, the switching function creates a routeing dialogue for the call. The switching function allocates a routeing cross reference identifier (routeingCrossRefID) that references this routeing dialogue.
3. The switching function sends the Route Request service to the computing function (that registered as the routeing server) for the routeing device or as the routeing server for all routeing devices within the switching sub-domain. This service request contains the routeing cross reference identifier, the route registration request identifier (if supported), and call information such as the Connection Identifier for the call, and calling and called numbers.
4. The computing function decides whether to reject the Routeing service request for this call, provide a route for the call, or end the routeing dialogue. If the computing function decides to reject the call, it sends the

switching function a Route Reject service request. If the computing function decides to provide a route for the call, it sends the switching function a Route Select service request containing the destination for the call. The computing function may include an optional flag in the Route Select service request (i.e., routeUsed) instructing the switching function to inform it of the call’s final destination. The final destination may be different than the computing function-provided destination when switching function features such as call forwarding redirect the call. If the computing function decides to end the routeing dialogue, it sends the switching function a Route End service request. In this case, the computing function does not provide a destination for the call and the switching function uses an alternate mechanism (not defined) to route the call.

5. If the switching function receives a Route Reject service request, then it returns the call to the network for alternate routeing, and sends the computing function a Route End service request to indicate that the routeing dialogue is ended. If the switching function receives a Route Select service request, it attempts to route the call to the computing function-provided destination. If the destination is valid, the switching function routes the call to that destination and sends the computing function a Route End service request to terminate the routeing dialogue. If the computing function-provided destination is not valid (e.g., invalid directory number, destination busy), then the switching function may send a Re-Route service request to the computing function to request a route to an alternate destination. If the switching function receives a Route End service request, it terminates the routeing dialogue.
6. If the computing function receives a Re-Route service request it can select a different destination for the call and send the switching function another Route Select service request. Depending on the switching function implementation, the re-routeing service request exchange can repeat until the computing function provides an acceptable route. The computing function will find out about a successful route when the switching function sends a Route End service request or if the computing function included the routeUsed flag in its last Route Select service request.

Either the switching function or the computing function may send a Route End service request at any time to end the routeing process and terminate the routeing dialogue. This releases the routeing cross reference identifier for use in the future. This service request indicates, for example, that the computing function does not want to route the call, or the switching function (usually in the absence of a Route Select service request) routed the call using some default mechanism within the switching function.

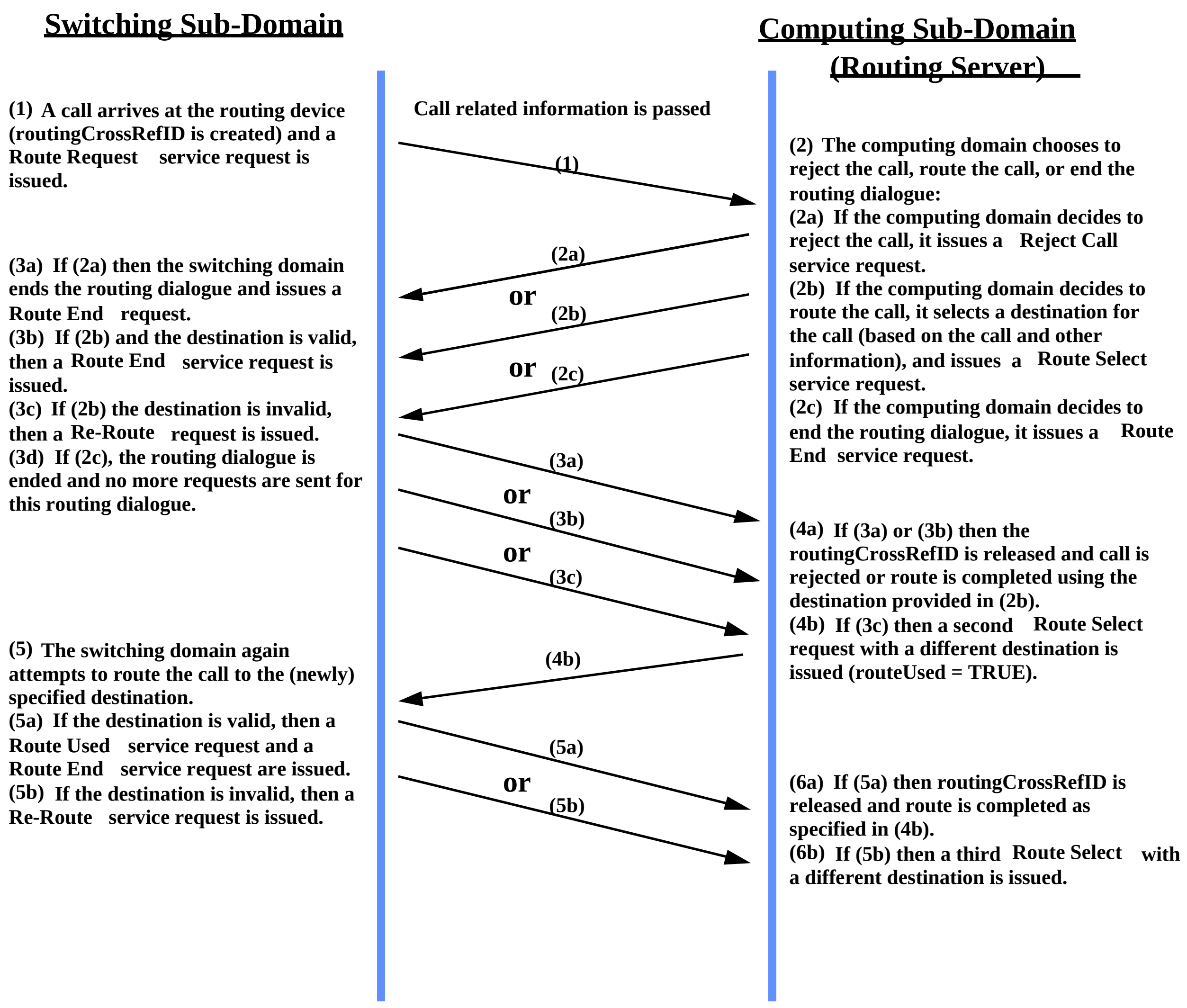
Note that a conflict may arise in this dialogue if the computing function invokes the Route End Service, for example to indicate that no more alternative routes are available, but still wants to receive a route used report via the Route Used Service invoked by the switching function. Avoidance or resolution of this conflict is the responsibility of the computing and/or switching function implementation(s).

A call that is not successfully routed does not necessarily mean that the call is cleared or not answered. Most switching function implementations will have a default mechanism for handling a call at a routeing device when the computing function has failed to provide an acceptable destination for the call. The switching function shall send a Route End service request to the computing function when it terminates the routeing dialogue, unless the routeing dialogue was terminated by a Route End service request from the computing function first.

The minimum set of services a switching function shall provide if it supports routeing are: Route Request, Route Select, and Route End (from the switching function). Other routeing services may be provided in any combination in addition to this minimum set.

Figure 6-26 illustrates the typical Routeing procedure.

Figure 6-26 Routeing Procedure


Switching Sub-Domain
Computing Sub-Domain (Routing Server)
(1) A call arrives at the routing device (routingCrossRefID is created) and a Route Request service request is issued.
Call related information is passed
(1)
(2) The computing domain chooses to reject the call, route the call, or end the routing dialogue:
(2a) If the computing domain decides to reject the call, it issues a Reject Call service request.
(2b) If the computing domain decides to route the call, it selects a destination for the call (based on the call and other information), and issues a Route Select service request.
(2c) If the computing domain decides to end the routing dialogue, it issues a Route End service request.
(2a)
or
(2b)
or
(2c)
(3a) If (2a) then the switching domain ends the routing dialogue and issues a Route End request.
(3b) If (2b) and the destination is valid, then a Route End service request is issued.
(3c) If (2b) the destination is invalid, then a Re-Route request is issued.
(3d) If (2c), the routing dialogue is ended and no more requests are sent for this routing dialogue.
(3a)
or
(3b)
or
(3c)
(4a) If (3a) or (3b) then the routingCrossRefID is released and call is rejected or route is completed using the destination provided in (2b).
(4b) If (3c) then a second Route Select request with a different destination is issued (routeUsed = TRUE).
(4b)
(5) The switching domain again attempts to route the call to the (newly) specified destination.
(5a) If the destination is valid, then a Route Used service request and a Route End service request are issued.
(5b) If the destination is invalid, then a Re-Route service request is issued.
(5a)
or
(5b)
(6a) If (5a) then routingCrossRefID is released and route is completed as specified in (4b).
(6b) If (5b) then a third Route Select with a different destination is issued.

6.8.8.3 Route Register Request ID and the Routeing Cross Reference ID

The routeing services use two identifiers to refer to different software objects in the switching sub-domain. The route register request identifier (routeRegisterReqID) identifies a routeing registration for which the computing function (acting as a routeing server) will receive Routeing service requests. This identifier may be associated with a particular routeing device within the switching sub-domain or it may indicate that the computing function is the routeing server for all routeing devices within the switching sub-domain. When the computing function uses the Route Register service to register for routeing services, it receives a routeRegisterReqID in the positive acknowledgement from the switching function. The routeRegisterReqID is only valid until the routeing session is ended by the computing function or switching function.

Within a routeing registration (routeRegisterReqID) the switching function may initiate many routeing dialogues (shown in Figure 6-26) to route multiple calls. A switching function uses a routeing cross reference identifier (routeingCrossRefID) to refer to each routeing dialogue. The computing function receives a routeingCrossRefID in each Route Request service request. The Route Request service initiates a routeing dialogue. The routeingCrossRefID is only valid for the duration of the routeing dialogue pertaining to a specific call.

The routeing cross reference identifier (routeingCrossRefID) is unique within the routeing registration (routeRegisterReqID). Some switching functions may provide the additional benefit of a unique routeing cross reference identifier across the entire switching sub-domain. This is also the case if routeing registration is not supported by the switching function. Routeing registration identifiers (routeRegisterReqIDs) are unique across a given CSTA service boundary.

6.8.8.4 Monitoring of Routeing Device

Some switching function implementations may support monitoring of routeing devices. For those computing functions that have an active monitor on the routeing device, any activity at the device (for instance call control activity) shall generate the relevant event sequence as specified throughout this specification.

6.8.8.5 Routeing Services with respect to Media Class

A routeing device can support the routeing of calls of any combination of media class (i.e., voice or digital data or both). Refer to the media class component of 12.2.15, “MediaCallCharacteristics”, on page 101 for the complete set of possible values.

Once the routeing session is visible to the computing function through the Route Request service, the media characteristics of the call will be identified and associated with the routeing cross reference identifier.

6.8.9 Device Maintenance

Device Maintenance events indicate changes in the maintenance state of a device. These events indicate if a device has been taken out of service (can no longer accept calls or be manipulated by the computing function), or if a device has been placed back in service.

6.8.10 Prompting

Some CSTA services (Make Call, Call Back, Pickup, Join Call, for example) may require to prompt the user of the targeted device in order to take that device off-hook. The implementation of a prompting mechanism is switching function specific (display flashing, ring pattern, lamp blinking, etc.).

For CSTA services that specify prompting (except the Make Call service), the switching function shall support (as indicated by the capability exchange services) one of the two possible prompting modes:

- prompting is a pre-condition to a service - in this mode prompting occurs before the execution of the CSTA service. The Service Initiated event that indicates prompting shall flow before any other service specific events and shall contain connection identifier that is not associated with the CSTA service. After the device goes offhook, a Connection Cleared event associated with the prompt is generated and the CSTA service that initiated the prompt is executed.
- prompting is part of a service - in this mode, prompting is part of the execution of the service. The Service Initiated event that indicates prompting is part of the completion criteria for the service and the connection identifier used in the Service Initiated event is associated with the CSTA service.

For information on event sequences with respect to prompting in the context of specific services, refer to the Monitoring Event Sequences associated with a CSTA service.

6.8.11 Telephony Tones Features

There are several features that support the generation and detection of telephony tones.

The Generate Telephony Tones service (18.1.4, “Generate Telephony Tones”, on page 338) generates a specified tone for a connection in a call. While a telephony tone is being generated, it may be canceled via the Cancel Telephony Tones service (18.1.2, “Cancel Telephony Tones”, on page 334).

The Telephony Tones Generated event (18.2.4, “Telephony Tones Generated”, on page 348) is used to monitor for telephony tones that are generated by a device (e.g., via the Generate Telephony Tones service).

The Data Collection services (Clause 25, “Data Collection Services”) are used to report telephony tones that are received over a connection at a device.

6.8.12 DTMF and Rotary Pulse Digits Features

Several services such as Make Call and Consultation Call provide a parameter for addressing a device while a call is being created. Also, for calls that are already created, the Dial Digits service provides address information to select a destination device or to complete a multi-stage dialling sequence. Depending upon the switching function implementation and the type of network, these parameters may be translated into DTMF or rotary pulse digit information used by the network to select a destination device. This addressing information shall not be used for end-to-end purposes.

Other services, as defined below, are used for generating and detecting end-to-end information that is to be sent to a device (i.e., not to address/select a device).

The Generate Digits service (18.1.3, “Generate Digits”, on page 336) is used to generate DTMF or rotary pulse digit information for a connection in a call.

The Digits Generated event (18.2.3, “Digits Generated”, on page 347) is used to monitor for DTMF or rotary pulse digits that are generated by a device, either manually or via the Generate Digits service.

The Data Collection services (Clause 25, “Data Collection Services”) are be used to report DTMF or rotary pulse digits that are received over a connection at a device.

6.8.13 Data Collection Services

The Data Collection services are used to collect information such as DTMF/rotary pulse digits and Telephony Tones that is received by a device over a connection.

The Start Data Collection service is used by the computing function to initiate the data collection. The service specifies if data should be collected for a specific connection or for the next connection at a device.

The Stop Data Collection service is used to stop the data collection. Data collection is also stopped if the connection over which data is being collected is cleared.

Information that is collected as part of the data collection is reported to the computing function via the Data Collected service.

The data collection may be suspended and resumed via the Suspend Data Collection and the Resume Data Collection services. The Data Collection Suspended and the Data Collection Resumed services notify the computing function if the data collection has been suspended or resumed.

The Data Collection services are specified in Clause 25, “Data Collection Services”.

7 Association Establishment

This Standard is based upon the assumption that the services defined here, and a protocol that supports these services, operate within an application association (otherwise known as a CSTA association or association) as provided by IS 8649 (ACSE). This association can be either:

- an implicit association achieved via off-line agreement or
- an explicit association realized through the use of ACSE.

The initialization sequence of CSTA messages for the implicit and explicit associations is described in the following sections.

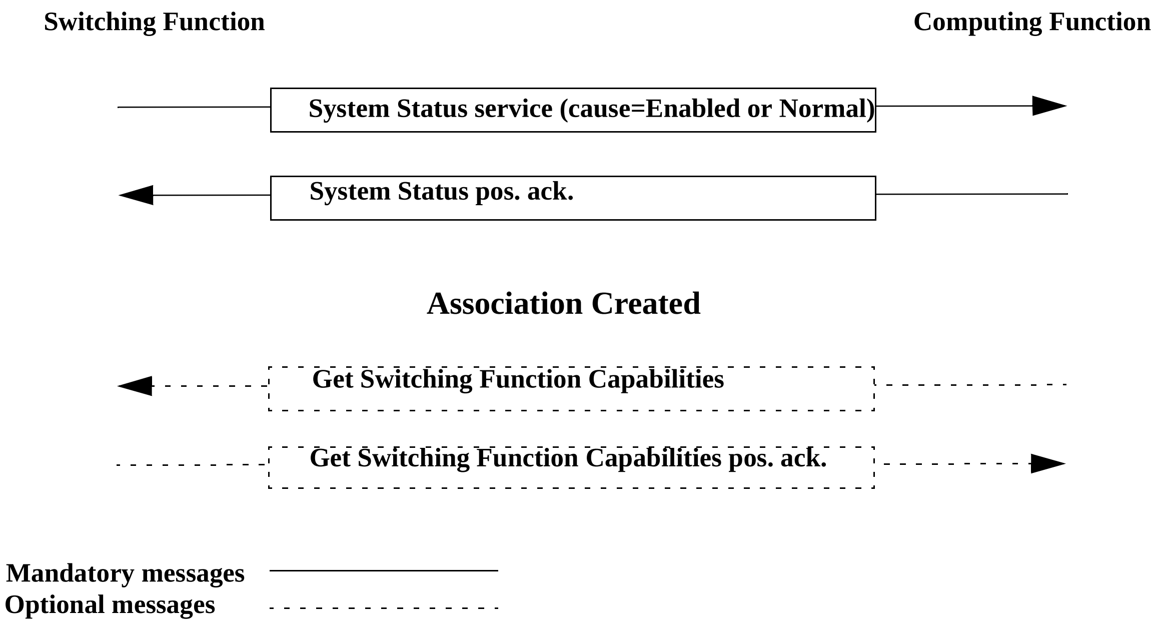
Once an association has been established, the switching function shall be prepared to receive CSTA services.

7.1 Implicit Association

In the initialization sequence for an implicit association, as shown in Figure 7-1, the switching function begins the sequence by sending a System Status service with a system status cause of either Enabled or Normal. The computing function shall respond with a positive acknowledgement. An implicit association is established once the positive acknowledgement is received by the switching function.

In this figure, the computing function uses the Get Switching Function Capabilities service to obtain the capabilities of the switching function after the association has been created.

Figure 7-1 Implicit Association - Initialization Sequence


Switching Function
Computing Function
System Status service (cause=Enabled or Normal)
System Status pos. ack.
Association Created
Get Switching Function Capabilities
Get Switching Function Capabilities pos. ack.
Mandatory messages
Optional messages

7.2 Explicit Association

In an explicit association, CSTA shall make use of a single application context name for all versions and variations of implementation of CSTA Services and Protocol. To facilitate the exchange of version and implementation information, CSTA specifies that the following information shall be exchanged in the ACSE *Association Information* field.

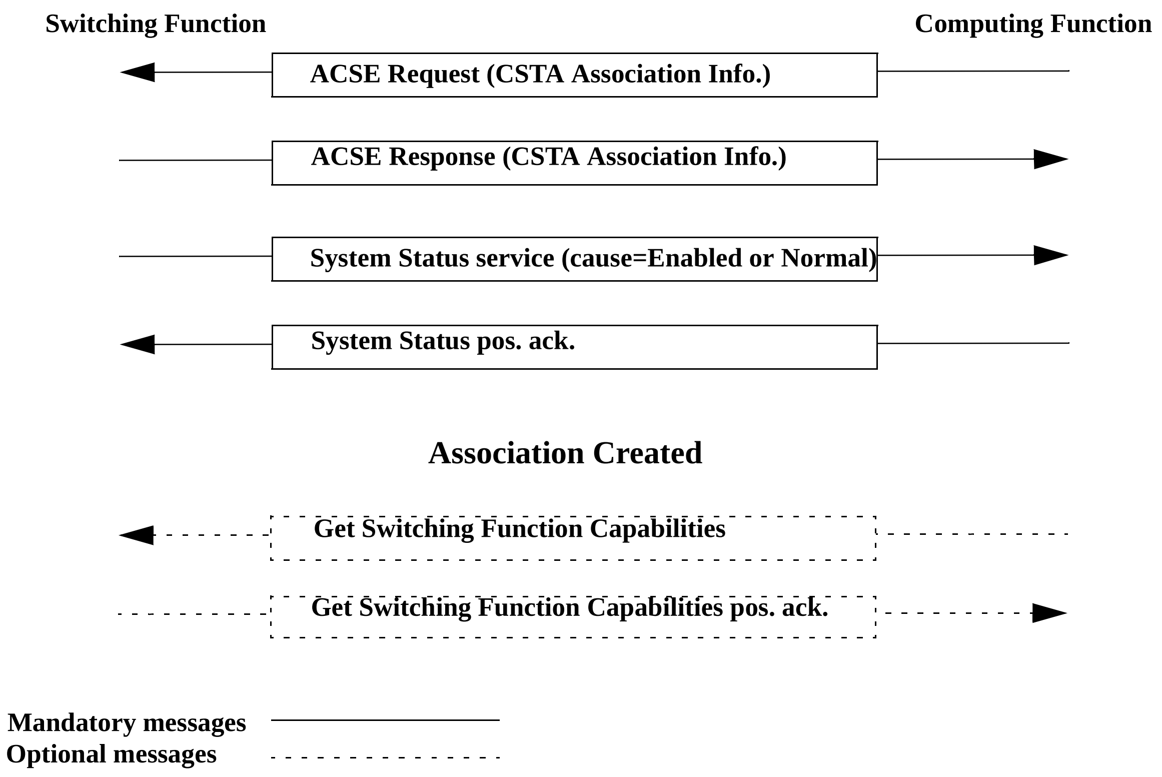
1. CSTA Association Information shall provide the following parameter:
   - CSTA Version - shall indicate the versions of the CSTA protocol that the implementation can support. If two interacting systems support more than one version, then the highest CSTA Version they both support shall be used for the association. A CSTA protocol version refers to the implementation of a specific phase of the ECMA Standard *Services for Computer Supported Telecommunications Applications* and is fully described in the corresponding ECMA Standard *Protocol for Computer Supported Telecommunications Applications*.
2. CSTA Association Information also may provide the following parameters:
   - Functionality Required - shall indicate the CSTA Services and Event Reports that are required by the function providing this information.
   - Functionality Offered - shall indicate the CSTA Services and Event Reports that are offered by the function providing this information for its highest-supported CSTA Version.
   - Private Data Version - shall indicate the Private Data versions that are offered by the function providing this information.

The initialization sequence for an explicit association is shown in Figure 7-2. The computing function begins the sequence by sending an ACSE request with the appropriate CSTA Association Information as described above. The switching function responds with an ACSE response that also includes the appropriate CSTA Association Information.

After the ACSE exchange, the switching function sends a System Status service with a system status cause of either Enabled or Normal. The computing function shall respond with a positive acknowledgement. The mandatory part of the initialization sequence is completed once the positive acknowledgement is received by the switching function.

In this figure, the computing function uses the Get Switching Function Capabilities service to obtain the capabilities of the switching function after the association has been established.

Figure 7-2 Explicit Association - Initialization Sequence


Switching Function
Computing Function
ACSE Request (CSTA Association Info.)
ACSE Response (CSTA Association Info.)
System Status service (cause=Enabled or Normal)
System Status pos. ack.
Association Created
Get Switching Function Capabilities
Get Switching Function Capabilities pos. ack.
Mandatory messages
Optional messages

8 Security Service

All CSTA messages provide:

- Timestamp information. This can be used to determine the “freshness” of a message.
- A Message Sequence Number. This provides a capability to number messages in a sequence so that the message receiver can detect that a message has been received out of sequence.
- Security Information. Support the implementation of a security process. This can be used to provide security such as access control and authentication. The format of this information is implementation specific.

For more information, refer to 12.2.11, “CSTASecurityData”, on page 88.

9 Generic Service Requirements

9.1 Service Request

This standard defines a set of CSTA operations that can be used to control and observe objects within a switching and/or special resource function. The CSTA operations are defined as “Services” in which one function requests, across the service boundary, that the other function perform a given CSTA operation. Services are defined for the CSTA service boundaries between the computing function, switching function, and special resource function. Services are defined in terms of what they accomplish (i.e. functionality), not how they should be implemented.

When one function sends a service request to the other function to perform a service with a given set of parameter values, it is called a *service request*. Each service defined in this Standard falls into one of following categories based upon the direction of the service request:

- *Switching Function Service* - Switching function services are services where the computing function is the client (i.e., service requestor) and the switching function is the server. An example of a switching function service is the Make Call service.

- *Computing Function Service* - Computing function services are services where the switching function is the client (i.e., service requestor) and the computing function is the server. An example of a computing function service is the Route Request service.
- *Special Resource Function Service* - Special Resource function services are services where the computing function is the client (i.e., service requestor) and the special resource function is the server. An example of a special resource function service is the Play Message service.
- *Bi-directional Service* - Bi-directional services are services where either the switching/special resource function or the computing function can be the client (i.e., service requestor). An example of a bi-directional service is the System Status service.

Some switching/special resource functions implementations support registration mechanisms that allow the computing function to indicate that it would like to receive service requests in a certain category (e.g., routeing, system status, escape) from the switching/special resource functions. (If the switching/special resource function indicates that it supports the computing function services in a particular category but does not support the registration mechanism, the computing function shall be prepared to handle the requests without previous registration.)

If the server detects that a service request is invalid, a negative acknowledgement shall be generated.

Every service request and service response defined in this Standard allows the inclusion of non-standardized, private data, that shall be informational in nature. Refer to 9.4, “Vendor Specific Extensions”, on page 71, for more information.

9.2 Service Response (Acknowledgements)

The other part of a service is the acknowledgement to the service request. This acknowledgement is used by the requesting function to verify that the other function has received the service request and that some level of processing has been performed with respect to the service. There are two types of acknowledgements: *positive acknowledgements* and *negative acknowledgements*, for a given service, as well as two types of positive acknowledgement models which a given service can adhere to. These definitions are documented in the following sections.

Note that there are some services defined in this Standard that do not provide a positive acknowledgement. For these services, if the service request is invalid, a negative acknowledgement shall be generated.

9.2.1 Positive Acknowledgement Models

All acknowledgements to each service request defined in this Standard shall follow the principles outlined by one of two models defined below. The computing function learns which model a switching function supports for each service through the capability exchange services described in Clause 13, “Capability Exchange Services”, beginning on page 118.

9.2.1.1 Atomic Model

Switching functions that indicate support of the atomic acknowledgement model designate that the particular service request can be accomplished in a single logical step. This acknowledgement model reflects whether or not the service request has meet the completion conditions as documented by each individual service.

An atomic positive acknowledgement indicates that not only were the parameters on the service request valid, but the switching function has successfully completed the service requested as defined in that service’s “Service Completion Conditions” section. The condition of the call(s) and/or connection states of the device(s) associated with the service request have transitioned to that service’s Operational Model After state.

9.2.1.2 Multi-Step Model

Switching functions that indicate support of the multi-step acknowledgement model designate that the particular service request is accomplished as its name implies, in multiple logical steps. This acknowledgement model reflects whether or not the parameters passed on the service were valid but does not guarantee anything as far as the completion conditions is concerned for the service.

A multi-step positive acknowledgement guarantees only that the parameters passed on the service request were accepted by the switching function to be valid. This positive acknowledgement does not determine if the service request's completion criteria are met. (However, depending on the switching function, the positive acknowledgement may indicate, in certain situations, the service request's completion conditions.) Therefore the computing function shall monitor for events associated with the particular service request, affected call(s) or device(s) to verify completion. A computing function shall also be prepared to handle the Service Completion Failure event and/or the Failed or Connection Cleared events after receiving the positive acknowledgement. The Service Completion Failure event will only be reported to the computing function which issues the service request *and* has a device-type monitor on the device which has or had connection(s) that were used in the particular request. Each of these events are provided by the switching function to indicate that the completion conditions for the service was not met.

If, through the event flow, a failure is detected, it is up to the computing function to apply the appropriate recovery to return the call(s) and/or device(s) back to the original conditions (if needed). Finally, a computing function should not issue subsequent service requests for a device until a previous multi-step service request's completion conditions has been satisfied. Doing so may result in unpredictable results generated by the switching function.

9.2.2 Negative Acknowledgement

A negative acknowledgement indicates that the service request has failed and the condition of the call(s) and/or connection states of the device(s) associated with the service request have not changed as a result of the failure (i.e., they remain as they were in the service's Operational Model Before state).

9.3 Diagnostic Error Definitions

CSTA provides diagnostic error information in the negative acknowledgement to service requests. The diagnostic error information consists of an error category and a category specific error value.

The definitions associated with the error categories and the error codes apply equally to services requested by a computing function and to those requested by a switching function. An error value indicates the server's best evaluation of the condition that caused the server to send a negative acknowledgement to the service request.

9.3.1 Error Categories

The error categories consist of the following:

- Operation Errors - Error values in this category shall indicate an error in the service request.
- Security Errors - Error values in this category shall indicate a security error.
- State Incompatibility Errors - Error values in this category shall indicate that the service request was not compatible with the condition of a related CSTA object.
- System Resource Availability Errors - Error values in this category shall indicate that the service request could not be fulfilled because of a lack of system resources within the serving sub-domain.
- Subscribed Resource Availability Errors - Error values in this category shall indicate that the service request could not be fulfilled because a required resource must be purchased or contracted by the client system.
- Performance Management Errors - Error values in this category shall indicate that an error has been returned as a performance management mechanism.
- Private Data Information Errors - Error values in this category shall indicate an error in the CSTA Private Data of the service request. The reason(s) why the private data is incorrect is not relevant to this CSTA Standard.
- Unspecified Errors - Error values in this category shall indicate that the error did not belong to any of the other error value categories.

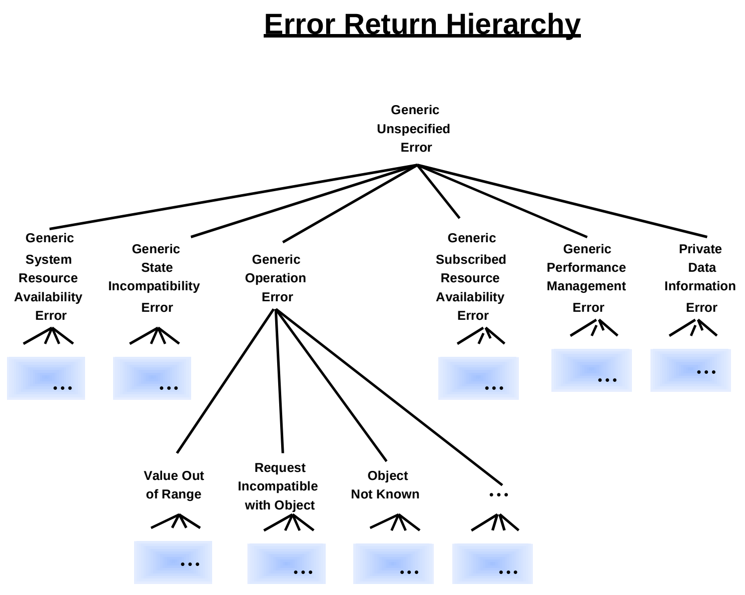
9.3.2 Error Values

The following definitions are used in all services to ensure a uniform meaning for error codes.

- Error codes reflect why the server could not carry out the service request on the specified call, device, or connection and do not reflect the status of any other call, device or connection.
- Error codes reflect why the server could not perform the request at the time that it attempted to execute the request. Thus a switching function will return the same error code in the same circumstance regardless of the past history of any object involved in the request.

- This Standard does not require that service parameters are validated in any order. Thus, when there are multiple errors in parameters (or when multiple errors apply to a single parameter), the computing function may receive any of the applicable errors.
- There is a hierarchy of error return values. The errors range from one high level error that spans all errors (Generic Unspecified) to specific detailed errors. The diagram below shows the hierarchy. The errors become more detailed toward the bottom of the diagram.

Figure 9-1 ErrorValue Hierarchy


Error Return Hierarchy
Generic Unspecified Error
Generic System Resource Availability Error
Generic State Incompatibility Error
Generic Operation Error
Generic Subscribed Resource Availability Error
Generic Performance Management Error
Private Data Information Error
Value Out of Range
Request Incompatible with Object
Object Not Known
...

The specific error values are defined in 12.2.12, “ErrorValue”, on page 88.

9.4 Vendor Specific Extensions

This Standard allows the provision of value added services and events that are beyond what is defined in this Standard. It is possible both to extend the existing services and events defined in this Standard as well as to create completely new services and events. A vendor may choose to support a vendor specific extension with the understanding that it may not interoperate with other CSTA (Phase III)-compliant products.

9.4.1 Private Data

Every service in this Standard allows for the inclusion of implementation-specific *private data*. This may be any supplemental information (not defined by this Standard) which provides access to vendor-specific extensions.

The computing function and the switching function, by mutual agreement (e.g., using the private data negotiation mechanism described below), assume full responsibility for the structure, representation (including byte order) and interpretation of this data. It is recommended that vendors adopt a platform independent encoding scheme (e.g., ASN.1/BER) for their private data.

If an implementation receives private data in a CSTA service or event that it does not recognize, it shall ignore the private data and process the rest of the CSTA service or event.

The size of private data is not limited by this Standard and is switching function and/or computing function specific. The capabilities exchange services can be used by the computing function to determine the maximum size used by the switching function implementation.

9.4.1.1 Private Data Version Negotiation

Private data version negotiation may be performed using the following process:

1. The switching function provides the computing function with its manufacturer name in the positive acknowledgement of the Get Switching Function Capabilities service. By associating the manufacturer name with information in the computing function, the computing function can determine if it supports the switching function's private data and its associated private data version negotiation mechanism. The switching function may also provide its supported private data versions in the Get Switching Function Capabilities acknowledgement.
2. The computing function will send the version to be used in the Private Data Version service request. The computing function may change the negotiated version by sending Private Data Version service requests at any time.

9.4.1.2 Private Data on CSTA Services and Events

For the services defined in this Standard, the use of private data allows vendor specific parameters to be added to each service. Private data should only be used to extend the existing definition of a service, and never to redefine the meaning of a service or any of its specified parameters. If a completely new vendor specific service is to be defined, the Escape service shall be used.

Private data can also be used to provide vendor specific parameters on events. As with services, private data should only be used to extend the existing definition of an event, and never to redefine the meaning of an event or any of its specified parameters. If a completely new event is to be defined, the Private event shall be used.

9.4.2 Escape Services and Private Event

The Escape service and the Private event are unique in that they include only private data and no other service specific parameters. Furthermore, they do not have any defined intent or meaning other than to allow for vendor specific extensions. The Escape service and the Private event may be used to define completely new services and events (i.e., ones not defined in this Standard), respectively.

9.4.2.1 Escape Registration

Before the computing function can receive any Escape service requests, it may be required to register with the switching function for escape services using the Escape Register service. The positive acknowledgement to this service contains the escape register identifier (escapeRegisterID) that the computing function uses to identify service requests that arrive for this registration.

If the switching function supports the Escape Registration services, then the computing function shall use the Escape Register services to register for escape services before it can receive any Escape service requests. If the switching function does not support the Escape Registration services, then the computing function may receive Escape service requests at any time. The capabilities exchange services can be used to determine if the switching function supports the Escape Registration services.

An escape registration can be cancelled using the Escape Register Cancel service. Once the switching function sends a positive acknowledgement to this, it will no longer send Escape service requests to the computing function. Additionally, the switching function can cancel an escape registration at any time by sending the computing function an Escape Register Abort service request.

While the Escape service itself is bi-directional, the Escape Registration services are not. These services are only issued by the computing function. The switching function does not register with the computing function for escape services. The switching function is considered to be (implicitly) registered to receive Escape service requests from the computing function at any time. The computing function never needs an escape registration to issue an Escape service request.

9.4.2.2 Private Data Version Service

The Private Data Version service is used by the computing function to negotiate a private data version (or to negotiate no private data). The Private Data Version service can be used as needed to re-negotiate (i.e., change) the private data version being used by the computing and switching functions.

9.4.2.3 Escape Service

The Escape service is used to request completely new, vendor-specific services not defined by this Standard. This service is bi-directional (i.e., can be issued by either the switching function or computing function). The vendor specific parameters to the Escape service request are transported by the private data (privateData) parameter to the service request. It is the responsibility of the vendor to define any extended services and the contents of the private data for these services.

The Escape service request may contain an escape registration identifier (escapeRegisterID) to identify the associated escape registration (when escape registration is supported by the switching function). An Escape service request from the computing function should never contain an escape registration identifier.

Escape service requests are always acknowledged by a positive or negative acknowledgement from the serving function (i.e., computing function or switching function processing the service request).

9.4.2.4 Private Event

The Private event is used to report completely new, vendor-specific events not defined by this Standard. As with other events, the computing function shall use the monitoring mechanism (i.e., Status Reporting services) in order to receive Private events. The type of monitors (i.e., call-type or device-type) on which a Private event is reported is switching function implementation specific. The monitor cross reference identifier (monitorCrossRefID) parameter associates the event with the monitor. There is no mechanism defined to allow the computing function to send Private events to the switching function.

The vendor specific parameters associated with the Private event are transported by a private data (privateData) parameter. It is the responsibility of the vendor to define any extended events and the contents of the private data for these events.

9.5 General Services and Event Functional Requirements

The following sections discuss functional requirements that are applicable to the services and events specified in this Standard.

9.5.1 Services

1. If a service is performed manually from a device, computing functions that have device-type or call-type (for device or call) monitors on this device receive the same event sequence as reported when performing the service through the service boundary (i.e., computing function-initiated). Refer to the appropriate service’s “Monitoring Event Sequence” sections for details.

   Depending on the particular switching function, additional events may also be reported as part of manual invocation services. For example: [2]

   - A Held event, if the device already has an active call.
   - A Service Initiated (event cause of NewCall) event for the device because a new call is needed to execute the service manually. This is followed by a Connection Cleared (event cause of Normal Clearing) event for the device when the service has been executed.
   - Logical and Physical device events that are associated with the execution of the service. These events may appear any time during the execution of the service.

2. If a service request is invoked after a device has manually gone off-hook (Service Initiated event), an implementation may either accept the service or it may reject the service. If it accepts the service, (unless otherwise specified for a particular service or event), the connection that has gone off-hook will be cleared and the computing function will receive a Connection Cleared event, followed by service specific events.

3. Other than for calls in the Initiated state, a service only affects connections that are specified by its service description. If, prior to its completion, the execution of the requested service would cause the switching function to affect any other connections, then the service shall be rejected with a negative acknowledgement.

2. These events are only reported if the computing function has the appropriate monitors started for the device (i.e., correct filters and type of monitor) and those monitors are supported by the switching function.

4. If the switching function permits the passing of Connection Identifiers without Call Identifiers, then the Device Identifiers they contain shall be within the switching sub-domain. In addition, if DeviceIDs only are passed in the Connection Identifiers, then:
   - If only one call exists at the specified device and the service request supports a single ConnectionID, its connection shall be in one of the initial states specified by the service or the service will be rejected.
   - If more than one call is in an initial state defined by the service, the service request will be rejected.
   - If two calls exist at the specified device and the service request supports two ConnectionIDs in the service request, the DeviceIDs within the ConnectionIDs shall be identical or the service will be rejected.
   - If two calls exist at the specified device and the service request supports two ConnectionIDs in the service request, both calls shall be a valid combination of initial states specified for that service or the service will be rejected.
   - If more than two calls exist at the specified device, the service will be rejected unless full and valid ConnectionIDs are specified.
5. If the device that is the subject of a service request is not capable of performing the service, a negative acknowledgement with an appropriate error code will be provided.
6. For optional parameters in service requests the following requirements apply:
   a. If an optional parameter is supported by the switching function but is not supplied in a service request, the switching function uses the specified default value associated with that parameter unless otherwise specified.
   b. If an optional parameter is not supported by the switching function, the switching function uses its administered value unless otherwise specified. (In addition, if the non-supported parameter is passed in the service request, see Services Requirement #7).
7. The switching function may either reject service requests that contain optional parameters that it does not support, or, it may accept the service request and ignore the unsupported optional parameters. However, the switching function shall handle unsupported optional parameters the same way for all service requests. The switching function indicates how it handles unsupported optional parameters via the capabilities exchange services.
8. When setting a value for a Physical or Logical Device Feature (specifically the “Set” features described in Clause 21, “Physical Device Features”, on page 383 and Clause 22, “Logical Device Features”, on page 431), the switching function shall return a positive acknowledgement when the feature is already set to the requested value specified in the service request. (Since the service request, in this case, did not result in a change of feature status, a feature event will not be generated.)
9. It is the switching function’s responsibility to verify that connections in a call are in their proper initial states prior to accepting a service request. Acceptable states are documented in each service request’s description.

9.5.2 Events

1. For the same telephony situation, the event generated for a call-type monitor will be the same as the event generated for a device-type monitor, except that the localConnectionInfo and the servicesPermitted parameters described in the Call Control event descriptions are not provided for events generated for call-type monitors.
2. If the computing function has call-type monitoring in effect, the event seen for that monitor will be the same event as the one seen for the subject device from a device-type monitor.
3. If the Device Identifier portion of a Connection Identifier is a static Device Identifier, then that portion of the Connection Identifier and the Device Identifier parameters in an event will not necessarily be the same. For example, the switching function may have a static internal representation of a device which will be used in the Connection Identifier, but the actual diallable representation for the same device may be different and may be used in one of the Device Identifier parameters in the same event. This requirement is in addition to and does

not supersede the definition for the Connection Identifier or Device Identifier parameters described in 12.3.9, “ConnectionID”, on page 110 and 12.3.11, “DeviceID”, on page 112.

4. The set of state transitions (refer to Figure 6-17, "Connection State Model" on page 33) supports the services and features documented in this Standard.

10 CSTA Device Identifier Formats

This clause describes the formats that may be used for Device Identifiers, their usage, and examples.

10.1 Device Identifier Formats

The possible types of Device Identifiers formats are:

- *Diallable Digits* - this format is a sequence of characters to be dialled to reach a device. The sequence of characters may contain diallable digits and/or special characters that specify to the switching function how digits should be dialled (“,” indicates that a pause should be inserted into the dialling sequence, for example). This format must be used when special dialling characters are required or when it is necessary to provide partial or incomplete dialling sequences.
- *Switching Function Representation* - this format is a sequence of characters that is used to reference devices within a switching sub-domain. In addition to specifying the directory number of the device, it also provides the ability to specify call appearance, agent identifier, subaddress, name, etc.
- *Device Number* - this format is an non-diallable, integer representation of a Device Identifier. This format of Device Identifier can be used to reference switching sub-domain devices that may not be typically associated with a diallable number such as trunks, line cards, etc.

In this section, the following example will be reflected. The called number is a subscriber in the US (country code 1) in San Jose (area code 408). The local number is 996 1010. The extension is 321. The name of the subscriber is “John Smith”.

10.1.1 Diallable Digits

Generic Format: DD

A first character of the Device Identifier string which is not “N”, “\” or “O” indicates that the Device Identifier uses the Diallable Digits format. This format may contain from 0 (a *null formatted* Device Identifier) to 64 characters. DD is a string of dialling commands/digits. The following is the list of the complete set of permitted dialling commands/digits and their definitions:

0-9	These characters represents the number digits on a telephone keypad.
*****	This represents the “*” character, typically found on a telephone keypad.
#	This represents the “#” character, typically found on a telephone keypad.
A-D	These characters represent DTMF digits.
!	The exclamation mark indicates that a hookflash is to be inserted into the dial string.
P	The character P followed by a string of digits indicates that the string of digits is to be pulse dialled.
T	The character T followed by a string of digits indicates that the string of digits is to be tone dialled.
,	The comma character indicates that dialling is to be paused. The length of the pause is provided by the switching function through the capabilities exchange services. Multiple commas can be used to create a long pause.
W	The character W followed by a string of digits indicates that the string of digits is to be dialled only after dial tone has been detected by the switching function.
@	The “at” symbol indicates that the switching function shall wait for “Quiet Answer” before dialling the rest of the string. This means that the switching function shall wait for remote ringing indication, followed by 5 seconds of silence.

$	This dollar sign indicates that the switching function shall wait for the billing signal (i.e., credit card prompt tone) before continuing.
;	The semi-colon character indicates that the digit string is incomplete and more digits will be dialled using the Dial Digits service. This character may only be used in a Diallable String Device Identifier.

Examples:

- If the number is called from France (country prefix 00[3]), the string is "00,14089961010W321".
- If the number is called from a switch in New York (dial 9 to get outside line), the string is "9,14089961010W321".
- If the number is called from San Jose, the string is "9961010W321".
- If the number is called from inside the subscriber's PBX, the string is "321".

Functional Requirements:

1. The switching function shall accept, as a minimum, digits 0-9 of this format when the computing function wants to make a call.
2. The diallable digits format shall be used to represent a device's dialling sequence. A device's dialling sequence is a string of outband digits used to initiate a call with another device. When placing a call from a device to another device, there are basically two ways a device's dialling sequence can be used:
   a. The entire sequence of digits is dialled to reach the destination. This is the most common way to place a call.
   b. The dialling sequence is broken up into a number of stages in order to execute and complete the call. This is called "multi-stage" dialling in this Standard. This type of dialling is needed in cases where the switching function prompts the device for more digits (by sending dialtone again or some other tone).

Note that switching functions support different combinations of dialling sequences.

10.1.2 Switching Function Representation

Generic Format: N<DN!SA&CA/EXT%AID>NM (*in this order*)

The syntax of the generic format is broken down as follows:

N	The "N" character at the beginning of the Device Identifier string (which is 2 to 64 characters in length) indicates that the Device Identifier uses the Switching Function Representation format. At least one of the following components needs to be present in this format:
< >	The angled brackets characters encompass the string when a name (NM) string representing the person associated with the device is provided after the ">" character. If the character "<" is not the first character in the string after the N then the string will not have a name string associated with it.
DN	The first string of characters represents the Directory Number (DN) associated with the given device. The Directory Number shall contain characters selected from the following set: "0" through "9", "*", "#", DTMF digits "A" through "D". The Directory Number may use any of the following notations:

3. The country prefix is the sequence of digits that needs to be dialled to make an international call (011 when calling from the US). It is always followed by the called country code.

- Implicit TON (Type Of Number)
  example: “0014089961010”[4]
  (refer to ECMA-155)
- PublicTON - unknown
  (refer to ECMA-155)
- PublicTON - international number
  example: “14089961010”
  (refer to ITU-T E.160)
- PublicTON - national
  example: “4089961010”
  (refer to ITU-T E.160)
- PublicTON - subscriber
  example: “9961010”
  (refer to ITU-T E.160)
- PublicTON - abbreviated
  example: “17”
  (refer to ITU-T E.131)
- PrivateTON - unknown
  (refer to ECMA-155)
- PrivateTON - level 3 regional
  example: “41396557321”
  (refer to ECMA-155)
- PrivateTON - level 2 regional
  example: “96557321”
  (refer to ECMA-155)
- PrivateTON - level 1 regional
  example: “557321”
  (refer to ECMA-155)
- PrivateTON - local
  example: “321”
  (refer to ECMA-155)
- PrivateTON - abbreviated
  example: “2”
  (refer to ECMA-155)
- Other (other numbering plans)
- Generic (the notation is unknown)

! This exclamation mark character represents the start of a Sub-Address (SA) string. If the “!” character is not present, then there will be no sub-address associated with this Device Identifier string. The termination character for the sub-address string will be the next key character found in the string or null.

& The ampersand symbol represents the start of a Call Appearance (CA) string. It is added to the logical element’s device identifier to uniquely identify an addressable standard appearance. The value of the string is switching function specific. The valid characters for the call appearance string

4. This example is a caller in France dialling the country prefix (00), the USA country code (1), the trunk code (408) and the subscriber number.

	are 0-9. The termination character for the call appearance string will be the next key character found in the string or null. Refer to 6.1.3.2.1, “Appearance”.
/	The slash symbol represents the start of a physical element extension (EXT) string. It is added to the logical element’s device identifier to uniquely identify a bridged appearance. Its value is the physical element’s device identifier that is associated with the appearance. The termination character for the physical element extension string will be the next key character found in the string or null. Refer to 6.1.3.2.1, “Appearance”.
%	The percent sign represents the start of an Agent ID (AID) string. This string represents an ACD agent identifier associated with a device. This string may be present when the computing function wants to focus a service at a specific agent identifier that is associated with a device or when the switching function generates an event that is associated with a particular device and agent. The valid characters for the agent identifier string are A-Z and 0-9. If the “%” character is not present then there will be no agent identifier associated with this Device Identifier string. The termination character for the agent identifier string will be the next key character found in the string or null.
NM	The name string (NM) represents the person associated with the device. This string can be used for selecting a Device Identifier associated with a user or for logging and informational purposes. The name string may contain any character.

Example:

- If the Device Identifier is PublicTON International, then the string can be “N14089961010”.
- If the Device Identifier is PublicTON Subscriber, then the string can be “N<9961010>John Smith”.

Functional Requirements:

1. This format shall always contain at least a directory number string or an agent ID string.
2. The interpretation of additional digits beyond those that are required to reach a destination are switching function specific.
3. When there is more than one bridged appearance associated with a single physical element (see 6.1.3.3.6, “Hybrid”, on page 19 for an example) there are two methods for representing these appearances: One is to have a unique call appearance (CA) and physical element extension (EXT) combination for each appearance where EXT is used to represent the given physical element and CA is used to represent multiple appearances associated with the same physical element. The other is to have a single EXT for each appearance, independently of their association with the physical element. In either case, the resulting Device Identifier is unique for the given appearance.

10.1.3 Device Number

Generic Format:

The Device Number format represents a Device Identifier using an integer. The integer shall be maximum size of four octets.

10.2 Functional Requirements

1. If the switching function detects a problem with a Device Identifier, the service will be rejected with a negative acknowledgement.
2. The switching function may use any format in service acknowledgements and events.
3. For Device Identifiers in service requests, the computing function should check the deviceIDFormat parameter in a capabilities exchange service to determine:
    - Which formats are supported.
    - For the Switching Function Representation format, which notations are supported.

- For the Diallable Digits format, which special characters are supported.

4. When providing a null Device Identifier, the Diallable Digits Format is used.

11 Template Descriptions

This Clause explains the template formats used to describe the CSTA services, events, and parameter types defined in this Standard.

11.1 Service Template

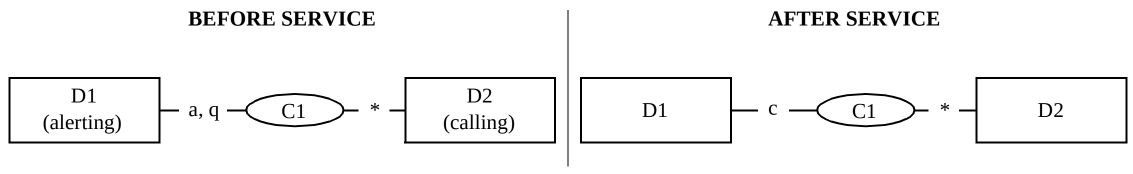
The following sections describe the Services template components.

11.1.1 Service Description

This is textual description of the service that may be followed by a figure. The figure is included when a service affects a connections state(s). The figure defines the role of devices and connections from a before/after service execution perspective. Note that this figure indicates the successful completion of the service but does not indicate the service completion criteria (see the Monitoring Event Sequences for the service completion criteria).

In order to describe the nomenclature used in the figures in the templates, the figure from the CSTA Answer Call service follows:

Figure 11-1 Example of a Figure in a CSTA Service (Answer Call)



DEVICES
D1 : alerting device
D2 : calling device

CALLS
C1 : call that has been delivered to D1

AFFECTED CONNECTIONS
D1C1 : *callToBeAnswered* connection

CONNECTION STATES
a : alerting state
c : connected state
q : queued state
* : (unspecified/unaffected)

In the figures, small boxes (labeled Dx) are used to represent devices, lines represent connections, ovals (labeled Cx) represent calls, and dotted lines represent connections with partial connection identifiers (see 6.8.2). The legend (large box) associates names with devices and calls. Names in italics refer to parameter names used in the service.

The connections are labeled with the set of possible connection states. In some cases the following symbols are used in place of a specific connection state:

“*” indicates that the connection state is not specified and it is not affected by the service

“!” indicates that the connection state is unspecified but may be affected by the service

“#” indicates that the connection state is not specified but the connection state is inherited from a connection that used to exist (for example, when a connection at a device changes its call identifier).

“@” indicates any non-Null connection state.

11.1.2 Service Request

This section contains a table with the possible parameters in the service request. Associated with each parameter are:

- Parameter Name (“Parameter Name”) - This is used to reference the parameter from other parts of the template and to distinguish the parameter from other parameters with the same parameter type. An example of a parameter name is connectionToBeCleared.
- Parameter Type (“Type”)- This is the parameter type as defined in Clause 12, “Parameter Types”, on page 82. In most cases a parameter type references a parameter type defined in either 12.2, “Defined Parameter Types”,

on page 83 (CorrelatorData, for example) or 12.3, “Identifier Parameter Types”, on page 107 (ConnectionID, for example). In other cases the parameter type may be a Boolean, Value, Enumerated, etc. Refer to 12.1, “Definitions”, on page 82 for more information.

- Parameter Optionality (“M/O/C”) - Indicates whether the parameter must be included (M for mandatory), if the parameter is optional (O), or if the parameter is conditional (C). If a parameter is conditional, then there are specific requirements when the parameter must be supported. These requirements are described in the parameter description column.
- Parameter Description (“Description”) - This is a brief description of the parameter in the context of the service. A description of the parameter in the context of its parameter type can be found with its parameter type description in Clause 12, “Parameter Types”, on page 82.

11.1.3 Service Response

This section includes:

- a description of the type of acknowledgement model that can be used with the service (see 9.2.1, “Positive Acknowledgement Models”, on page 69).
- a table that contains all of the parameters in the positive acknowledgement. The format of the table is the same as in the service request (see 11.1.2).
- a reference to the negative acknowledgement error codes.

11.1.4 Operational Model

The operational model consists of:

Connection State Transitions - This is a table with all possible connections affected by the service. Associated with each connection is the:

- Connection Name (“Connection”) - This is used to reference the connection from other parts of the template including the figure in the service description.
- Initial State (“Initial State (Required)”) - This is the set of allowed initial states (connection states before the service is executed). An implementation shall support one or more of the specified initial states associated with a service (as indicated in the capability exchange services).
- Final State (“Final State”) - This is the set of allowed final states (connection states after the service is executed). An implementation shall support one or more of these states. In many cases there are statements following the connection state transition table that further describe or clarify the information in the table.

Monitoring Event Sequences - For services that affect connections, this section includes tables that describes the event sequence generated for device-type and call-type monitors. Unless otherwise specified, the events (and associated causes) in this table are required as part of the service completion criteria. Each table contains:

- Monitored Device or Monitored Call (“Monitored Device” or “Monitored Call”) - For the device-type monitoring table, this indicates the monitored device. For the call-type monitoring table, this indicates the monitored call. The names can be used to reference back to the figure in the service description.
- Connection Name (“Connection”) - This column indicates the connection that is the subject of the event.
- Event (“Event”) - This is the name of the event generated as the result of the service.
- Event Cause (“Event Cause”) - This is the set of possible cause codes associated with the event. In many cases there are statements following the connection state transition table that further describes or clarifies the information in the table

Functional Requirements - The functional requirements contain additional requirements associated with the service.

11.2 Event Template

The following sections describe the Event Template components.

11.2.1 Event Description

This is textual description of the event followed by an optional figure. The figure is included when an event indicates a change in one or more connections. The figure defines the role of devices and connections from a before/after perspective. The nomenclature used in the figures is described in 11.1.2, “Service Request”.

11.2.2 Event Parameters

This section consists of a table that contains all of the parameters in the event. The format of the table is the same as in the service request table described in 11.1.2, “Service Request”.

11.2.3 Event Causes

This section consists of a table that contains all of the possible cause codes that can be included with the event. Associated with each cause code are:

- Event Cause (“Event Cause”) - This is the event cause name.
- Event Description (“Description”) - This is a description of the event cause in the context of the event.
- Associated Features (“Associated Features”) - This is the complete set of possible features associated with the cause code. A feature may either correspond to a CSTA service or it may be associated with switch features specified in 6.8, “Additional Services, Features & Behaviour”, on page 51.

11.2.4 Functional Requirements

Functional requirements contain additional requirements associated with the event.

11.3 Parameter Type Template

The following sections describe the Parameter Template components:

11.3.1 Parameter Type Description

This contains a description of the parameter type.

11.3.2 Format

This section specifies the format of the parameter type. For example, it could list the possible values associated with a parameter type (enumerated list).

For parameter types that are a type of device identifier, the format contains the allowed statuses associated with the parameter type (e.g. “Not Known”).

11.3.3 Functional Requirements

Functional requirements contain additional requirements associated with the parameter type.

12 Parameter Types

12.1 Definitions

This clause describes the parameter types for the parameters described in this Standard. There are five sets of parameter types:

1. *Basic parameter types* are simple types that are not necessarily specific to these specifications. The basic parameter types used in this Standard are:
   - *Boolean* - Either TRUE or FALSE.
   - *Value* - Integer value with a length of 4 bytes always.
   - *Characters* - Character string of varying lengths, as specified in specific services or events.
2. *Meta parameter types* refer to constructions that combine one or more parameter types. The meta parameter types used in this Standard are:
   - *Bitmap* - Multiple values may be set in a specified set.
   - *Enumerated* - One value only may be set in a specified set.
   - *Structure* - A combination of different types combined into one parameter type, as defined in a specific service or event. Multiple components may be present in the structure (each component in the structure is defined as mandatory, optional, or conditional).
   - *Choice Structure* - Like *Structure*, but one and only one component in the structure is present.
   - *List* - List of a single specified parameter type or structure
3. *Defined parameter types* are specific to this Standard. They are briefly defined in Table 12-1 on page 83, and further defined in the pages indicated.
4. *Identifier parameter types* are specific to this Standard. They are briefly defined in Table 12-2 on page 107, and further defined in the pages indicated.
5. *Capability bitmap parameter types* are bitmaps included in the Get Physical Device Information, Get Logical Device Information and Get Switching Function Capabilities services. They are defined in Annex C.

12.2 Defined Parameter Types

Defined parameter types specific to these specifications are summarized in the following table.

Table 12-1 Defined Parameter Types Summary

Defined Parameter Type	Description	Pg.
12.2.1 AccountInfo	Contains computing sub-domain/business specific code that is to be applied or has been applied to a call for accounting purposes.	84
12.2.2 AgentPassword	Specifies the agent password.	84
12.2.3 AuthCode	Contains an authorization code that the switching function understands and will use to check to see if the user of the computing sub-domain is authorized to perform the given service.	84
12.2.4 CallCharacteristics	Specifies the high level characteristics of the call.	84
12.2.5 CallQualifyingData	Specifies information such as wrap codes, walk away codes, hold reasons, transfer reasons, etc. that describes or helps qualify how a call is being (or has been) handled by a user.	85
12.2.6 ChargingInfo	Specifies information that represents a cumulative value of charging or currency units charged to a device for a call in which the device was involved.	85
12.2.7 ConnectionInformation	Specifies the connection information associated with the subject connection.	86
12.2.8 ConnectionList	Specifies the list of devices/connections that are known to the switching function, and which remain in the call after a conference or transfer.	86
12.2.9 CorrelatorData	Contains computing sub-domain-specific data that has been or will be attached to a call that the computing sub-domain is controlling or monitoring.	87
12.2.10 CSTAPrivateData	Provides a mechanism for providing non-standard parameters in events.	88
12.2.11 CSTASecurityData	Specifies the security attributes associated with the message.	88
12.2.12 ErrorValue	Contains hierarchical error codes.	88
12.2.13 EventCause	Provides additional information on why the event was generated.	98
12.2.14 LocalConnectionState	Describes the connection state of the device associated with the Monitor Cross Reference ID.	101
12.2.15 MediaCallCharacteristics	Specifies the media class (Voice, Digital Data, etc.) and media characteristics of the call.	101
12.2.16 MediaServiceType	Specifies which media service is to be (or has been) attached to or detached from a particular call or connection.	102
12.2.17 MonitorFilter	Specifies the events are filtered for a Monitor Start service.	103
12.2.18 ServicesPermitted	Specifies the set of services that the switching function permits to be applied to a connection.	104
12.2.19 SimpleCallState	Provides the simple call state.	104
12.2.20 SystemStatus	Indicates the reason for the System Status service request.	105
12.2.21 TimeInfo	Specifies the date and time.	106
12.2.22 UserData	Contains device-to-device or computing sub-domain-to-computing sub-domain data.	106

12.2.1 AccountInfo

The AccountInfo parameter type contains a computing sub-domain/business specific code that is to be applied or has been applied to a call for accounting purposes.

Format

This parameter type is a character string with a maximum length of 32.

Functional Requirements

1. The management of the account code data is done by the switching function. To understand how the switching function maintains this information with the call, you need to consult the switching function specific documentation.
2. The computing sub-domain will only be notified that the account code data has been added or changed through the Call Information event. This event will be generated when the user enters the account code data manually or after a service has added or changed the data.
3. The way to clear the data on the call is to pass a null string of data on one of the above mentioned services. (The actual parameter is passed, but the content is a null string.)
4. The switching function may choose to filter this information by not providing it in events for security reasons.
5. When this information is being attached to a call through a service request, its association shall be completed prior to any state transitions resulting from the request. Thus any state transition events which contain this parameter shall contain the information passed on the request.

12.2.2 AgentPassword

The AgentPassword parameter type specifies the password for an ACD agent.

Format:

This parameter is a character string with a maximum length of 32.

12.2.3 AuthCode

The AuthCode parameter type contains an authorization code that the switching function understands and will use to check if the computing function is authorized to perform a given service.

Format

This parameter type is a character string with a maximum length of 32.

Functional Requirements

1. If the switching function requires this parameter, and either the authorization code supplied is not valid or the parameter type is not supplied, then the service will be rejected with a negative acknowledgement.
2. The switching function may choose to filter this information by not providing it in events for security reasons.
3. When this information is being attached to a call through a service request, its association shall be completed prior to any state transitions resulting from the request. Thus any state transition events which contain this parameter shall contain the information passed on the request.

12.2.4 CallCharacteristics

The CallCharacteristics parameter type describes, when included on an event, the high level characteristics associated with a call.

When this parameter is included on a switching function service request, it indicates the requested set of high level characteristics that should be associated with the call.

Format

This parameter type is a bitmap. Multiple bits may be set. The complete set of possible values in the bitmap is:

- acdCall. This bit is set to indicate an ACD call. Once the call is no longer associated with the ACD, this bit is no longer set. See Functional Requirement #1.
- priorityCall - This bit is set to indicate a priority call.
- maintenanceCall - This bit is set to indicate a maintenance call.
- directAgent - This bit is set to indicate a call placed directly to a particular agent.
- assistCall - This bit is set to indicate a call whose purpose is to request assistance.
- voiceUnitCall - This bit is set to indicate a call involving a Voice Unit (e.g., voice mail system). Once the Voice Unit is no longer involved with the call, this bit is no longer set.

Functional Requirements

1. There are many conditions when a switching function may classify a call as an ACD call and when an ACD call becomes a non-ACD call. The specific conditions are switching function dependent.

12.2.5 CallQualifyingData

The CallQualifyingData parameter type specifies information such as wrap codes, walk away codes, hold reasons, transfer reasons, etc. that describes or helps qualify how a call is being (or has been) handled by a user.

Format

This parameter type is a character string with a maximum length of 32.

Functional Requirements

1. The computing function will be notified that the call qualifying data has been added or changed through the Call Information event. This event will be generated when the user enters the call qualifying data manually or after a service (Associate Data) has added or changed the data.

12.2.6 ChargingInfo

The ChargingInfo parameter type represents a cumulative value of charging or currency units charged to a device for a call in which the device was involved. This information can represent an intermediate (during the call) or final total (when the device leaves the call).

Format

This parameter consists of the following components:

- numberUnits (M) Choice Structure - This component consists of one of the following choices:
    - numberOfChargingUnits (List Structure) - indicates a cumulative number of charging units. This component consists of a sequence that may be repeated to report different types of charging units. The sequence consists of:
        - chargingUnits (M) Value - the number of charging units.
        - typeOfUnits (O) Octet String - the type of units. This may be included to differentiate among these types. Its definition is network-dependent.
    - numberOfCurrencyUnits (List Structure) - indicates a cumulative value of currency units. This sequence consists of:
        - currencyType (M) Octet String - indicates the type of currency. A null string (size of 0) indicates the default currency. Its definition is network-dependent.

    - currencyAmount (M) Value - indicates the cumulative value of currency units.
    - currencyMultiplier (M) Enumerated - indicates the currency unit multiplier. The complete set of possible values is: .001, .01, .1, 1, 10, 100, 1000.

- typeOfChargingInformation (M) Enumerated - This can have one of the following values:
  - Sub-total - indicates that the information is an intermediate value.
  - Total - indicates that the charging information is complete.

12.2.7 ConnectionInformation

The ConnectionInformation parameter type specifies the connection information associated with the subject connection (i.e., the connection that is the focus of the event or positive acknowledgement being reported).

Format

This parameter type is a comprised of the following parameters:

1. flowDirection (O) Enumerated - Specifics the direction of flow that is associated with the subject connection. If this parameter is not present, the connection's flow direction is unknown. The complete set of possible values is:
   - Transmit - Media stream data is only capable of being transmitted on the connection by the associated device.
   - Receive - Media stream data is only capable of being received on the connection by the associated device.
   - Transmit and Receive - Media stream data is capable of being transmitted and received on the connection by the associated device.
2. numberOfChannels (O) Value - Specifies the number of media stream channels that are associated with the subject connection. If this parameter is not present, the number of channels associated with the connection is one.

12.2.8 ConnectionList

The ConnectionList parameter type provides the linkage mechanism between a device's old connection ID and new connection ID resulting from the conference or transfer.

Format

This parameter includes the following components for every device or connection being reported:

- new ConnectionID (C) ConnectionID - The CallID portion of this ConnectionID refers to the resulting call. This component is optional for the transferringDevice in the Transferred event, otherwise it is mandatory.
- old ConnectionID (C) ConnectionID - The CallID portion of this ConnectionID refers to the original call. This component is mandatory if the switching function previously reported the CallID, otherwise it is optional.
- endPoint DeviceID (O) DeviceID - For internal calls, this is the representation of the device inside the switching sub-domain. For external calls (incoming or outgoing), this is the representation of the externally located device (if known by the switching function). This component is a character string. The maximum length supported by the switching function is provided via the capabilities exchange services. It may be:
  - of any device identifier format
  - of the following statuses: "Provided" or "Not Known"
- associatedNID (C) DeviceID - For external calls (incoming and/or outgoing), this component specifies the Network Interface Device (e.g., trunk, CO line) within the switching sub-domain that is associated with the externally located device. In that case the component endPoint DeviceID (if provided) shall represent the externally located device. The associatedNID component is mandatory in case of external calls and shall be omitted when the device is located inside the switching sub-domain. This component may be:

  - of any device identifier format
  - of the following statuses: “Provided” or “Not Known”
- resultingConnectionInformation (O) ConnectionInformation - This component contains the flow direction and channel characteristics associated with the resulting connection.

Functional Requirements

1. This list should be used by the computing function to associate devices which remain in a call, as a result of a Conference or Transfer, with the connection IDs that are used to manipulate them.

12.2.9 CorrelatorData

The CorrelatorData parameter type contains computing sub-domain specific data that has been or will be attached to a call that the computing function is controlling or monitoring. This allows the computing function to associate its own information with a call and, as a result, share it with other computing functions. For example, this information might be a key to a database entry, a computing function command sequence, file name, etc. This feature is useful when calls are moving from one computing function to another in a distributed computer network or from one switching sub-domain to another.

See 6.1.4.3, “Correlator Data”, on page 28 for specific rules on the use of Correlator Data.

This specification defines a mechanism for delivering both user data and correlator data through an external ISDN network at the same time. This mechanism is described in Annex B.

Format

This parameter type is an octet string. The maximum length supported by the switching function is provided via the capabilities exchange services, but is limited to 32.

Functional Requirements

1. The correlator data will stay with the call as long as the call exists. This means that the correlator data will be presented to the computing function on events that have this parameter and that the switching sub-domain supports.
2. The correlator data can be changed during the life of the call by any of the services that has the parameter or by using the Associate Data service.
3. The way to clear the data on the call is to pass a null string of data on one of the above mentioned services. (The actual parameter is passed but the content is a null string.) If correlator data is cleared, then the switching function notifies the computing function by sending a null string.
4. See 6.1.4.3, “Correlator Data”, on page 28 for a description of how Correlator Data is inherited by calls during a conference or transfer.
5. If the computing function issues the Consultation Call service without correlator data, initially the secondary call will not have correlator data associated with it, as it does not inherit any correlator data that may be associated with the primary call. If the computing function issues the Consultation Call service with correlator data, this data is for the secondary call only and does not affect any correlator data that may be currently associated with the primary call.
6. When correlator data is associated with a call, for all Call Control events listed in 17.2 on page 267 *except* for the Bridged, Call Cleared, Connection Cleared, Held, and the Retrieved events (i.e. call events that may indicate that a device becomes part of a call) shall include the correlator data (if supported). Correlator data can optionally be included with the four event exceptions listed above.
7. When this data is being attached to a call through a service request, its association shall be completed prior to any state transitions resulting from the request. Thus any state transition events which contain this parameter shall contain the information passed on the request.

8. If a computing function issues a Snapshot Call service after a service request has been issued with this information, but prior to the switching function making any state transitions, it is switching function specific as to what will be returned in the positive acknowledgement with regards to this parameter.

12.2.10 CSTAPrivateData

The PrivateData parameter type provides a mechanism for providing non-standard parameters in messages.

Format

This parameter type is a choice of one of the following:

- octet string of any length. The maximum length supported by the switching function is provided via the capabilities exchange services.
- ASN.1 NULL type. If this choice is used, an implementation shall replace the ASN.1 NULL type with another valid ASN.1 type.

12.2.11 CSTASecurityData

The CSTASecurityData parameter type provides information that can be used to determine if a message in a sequence has been lost, the time that a message was sent, and security information that can be used to provide security such as access control and authentication.

Format

The CSTASecurityData parameter type consists of the following components:

- messageSequenceNumber (Value) Optional. Shall be a sequential number that can be used to detect missing messages in a sequence and verify that their order has not been altered.
- timestamp (Timeinfo) Optional. Shall be a generalized time value that can provide an indication of the "freshness" of a message. It can indicate that the received message is not a replay of another message from a previous association or from the current association after the sequence numbers have recycled.
- securityInfo (Choice Structure) Optional - Shall indicate the security data that may be used to make appropriate access control decisions or to carry out the current security policy. This contents of this information is not defined by this Standard. This component is a choice of one of the following:
  - octet string of any length. The maximum length supported by the switching function is provided via the capabilities exchange services.
  - ASN.1 NULL type. If this choice is used, an implementation shall replace the ASN.1 NULL type with another valid ASN.1 type.

12.2.12 ErrorValue

The ErrorValue parameter type defines error codes.

Format

This parameter contains the following:

- *Error Category* - Operation, Security, State Incompatibility, System Resource Availability, Subscribed Resource Availability, Performance Management, Private Data, and Unspecified.
- *Error Value* - A value describing the error. The following text describes the various categories and values within each category.

Note that the error code hierarchy described in Figure 9-1, "ErrorValue Hierarchy," on page 71 is represented by indented bullet lists in the following sections, each indent representing an additional error code level.

12.2.12.1 Operation Errors

Error values in this category shall indicate an error in the Service Request. This category shall include one of the following specific error values:

- generic - The server has detected an operational error in the service request and the error is either not a specified Operation Class error or the switching function cannot be more specific.
    - atLeastOneConditionalParameterNotProvided - The service definition specifies a set of conditional parameters, at least one of which shall be provided. No parameter from this set was present.
    - featureAlreadySet - The feature cannot be set because it is already set.
    - invalidMessageIdentifier - There is no message with the specified Message Identifier.
    - invalidParameterValue - A value for a parameter is invalid. A value is in the specified range but invalid in the circumstance where it is used.
        - invalidAccountCode - The account code parameter is invalid.
        - invalidAgentGroup - An agent group is invalid.
        - invalidAgentIdentifer - An agent identifier is invalid.
        - invalidAgentPassward - An agent password is invalid.
        - invalidAgentState - An agent state setting is invalid.
        - invalidAlertTime - The alertTime parameter is invalid.
        - invalidAllocationState - The service request (MakePredictiveCall) specified an allocation state that is invalid in the present circumstance.
        - invalidAuthorizationCode - The authorization code is invalid.
        - invalidAutoAnswer - The autoanswer parameter is invalid.
        - invalidBitRate - The bitRate parameter is invalid.
        - invalidButtonIdentifier - A button identifier is invalid.
        - invalidButtonLabel - A button label is invalid.
        - invalidCallType - The callType parameter is invalid.
        - invalidConnectionRate - The connectionRate parameter is invalid.
        - invalidConsultPurpose - The consultPurpose parameter is not valid.
        - invalidCorrelatorData - The Correlator Data parameter is not valid.
        - invalidCrossReferenceIdentifier - The service request specified a Cross Reference Identifier that is not in use.
        - invalidDelayTolerance - The delayTolerance parameter is invalid.
        - invalidDestination - The service request contains a destination that is invalid. Note that for a forwarding destination, the switching function returns the invalidForwardingDestination error. This occurs when the calledDirectoryNumber, newDestination, or routeSelected parameter is invalid.
        - invalidDestinationDetect - the destinationDetect parameter is invalid.
        - invalidDoNotDisturb - The do not disturb setting is invalid.
        - invalidEscapeCrossReferenceIdentifier - The escape registration request identifier is invalid.
        - invalidFeature - The service request specified a feature that is invalid. Often, this is because the switching or computing function does not support the requested feature.
        - invalidFile - The specified file is not accessible.
        - invalidFlowDirection - The flowDirection parameter is invalid.
        - invalidForwardingDestination - The forwarding destination device is not valid.
        - invalidForwardingFlag - The forwarding flag is invalid.

    - invalidForwardingType - The forwarding type is invalid.
    - invalidHookswitchType - A hookswitch type is invalid.
    - invalidHookswitchComponent - A hookswitch component is invalid.
    - invalidLampIdentifier - A lamp identifier is invalid.
    - invalidLampMode - A lamp mode is invalid.
    - invalidMessageWaitingSetting - A message waiting setting is invalid. The switching function returns this error for the messageWaitingOn parameter.
    - invalidMicrophoneGain - A microphone gain setting is invalid.
    - invalidMicrophoneMute - A microphone mute setting is invalid.
    - invalidMonitorCrossReferenceIdentifier - The service request specified a monitor cross reference identifier that is not in use.
    - invalidMonitorFilter - The monitor filter is invalid.
    - invalidMonitorObject - The monitor object is invalid.
    - invalidMonitorType - The monitor type is invalid.
    - invalidNumberOfChannels - The numberOfChannels parameter is invalid.
    - invalidParticipationType - The participationType parameter is invalid.
    - invalidRemainRetry - The value of the remainRetry parameter is invalid.
    - invalidRingCount - The ring count setting is invalid.
    - invalidRingPattern - A ring pattern setting is invalid.
    - invalidRingVolume - A ring volume setting is invalid.
    - invalidRouteingAlgorithm - The computing function does not support the routeing algorithm.
    - invalidRouteingCrossReferenceIdentifier - The service request specified a routeing cross reference identifier that is not in use.
    - invalidRouteRegistrationCrossReferenceIdentifier - The route registration request identifier is invalid.
    - invalidSpeakerVolume - A speaker volume is invalid.
    - invalidSpeakerMute - A speaker mute setting is invalid.
    - invalidSwitchingSubdomainCharsType - The switchingSubDomainCCIEType parameter is invalid.
- invalidObjectType - A parameter in the service request contains an object type that is not the defined object type for that parameter.
    - invalidActiveCallObject - The value supplied for activeCall or one of its components is not of the proper type.
    - invalidCalledDeviceObjectType - The value supplied for calledDevice is not of the proper type.
    - invalidCallingDeviceObjectType - The value supplied for callingDevice is not of the proper type.
    - invalidCallToBePickedUpObjectType - The value supplied for callToBePickedUp or one of its components is not of the proper type.
    - invalidCallToDivertObjectType - The value supplied for callToBeDiverted is not of the proper type.
    - invalidCallToParkObjectType - The value supplied for callToPark or one of its components is not of the proper type.

    - invalidDestinationDeviceObject - The value supplied for newDestination in a or one of its components is not of the proper type.
    - invalidHeldCallObject - The value supplied for heldCall or one of its components is not of the proper type.
    - invalidMonitorObjectType - The monitorObject type is invalid.
    - invalidParkToObjectType - The value supplied for r parkTo is not of the proper type.
- messageIdentifierRequired - The request requires a Message Identifier.
- notDifferentDevices - Multiple parameters in the service request that shall specify different devices do not specify different devices.
- notSameDevice - Multiple parameters in the service request that shall specify the same device do not specify the same device.
- objectNotKnown - An object parameter (connection, device, or call) has a value that is not known.
    - invalidCallIdentifier - A call identifier parameter or a call identifier component of a connection identifier parameter is invalid or is not known to the switching function.
        - invalidActiveCallIdentifier - The call identifier in the activeCall connection does not specify a valid call.
        - invalidHeldCallIdentifier - The call identifier in the heldCall connection does not specify a valid call.
    - invalidConnectionIdentifier - A connection identifier or some component of the connection identifier is invalid.
        - invalidActiveConnectionIdentifier - The activeCall connection does not specify a valid call.
        - invalidHeldConnectionIdentifier - The heldCall connection does not specify a valid call.
    - invalidDeviceIdentifier - A device identifier parameter or a device identifier component of a connection identifier parameter is invalid or is not known to the switching function.
        - invalidActiveDeviceIdentifier - The device identifier in the activeCall connection does not specify a valid device.
        - invalidCalledDeviceIdentifier - The called device parameter is invalid.
        - invalidCallingDeviceIdentifier - The calling device parameter is invalid.
        - invalidCallToParkDeviceIdentifier - The device identifier in the callToPark connection does not specify a valid device.
        - invalidDestinationDeviceIdentifier - The device identifier in the newDestination does not specify a valid device.
        - invalidDivertingDeviceIdentifier - The diverting device identifier is invalid.
        - invalidHeldDeviceIdentifier - The device identifier in the heldCall connection does not specify a valid device.
        - invalidParkToDeviceIdentifier - parkTo does not specify a valid device.
        - invalidPickUpDeviceIdentifier - The device identifier in the callToBePickedUp does not specify a valid device.
- parameterNotSupported - The switching function does not support a parameter.
    - accountCodeNotSupported - The accountCode parameter is not supported.
    - agentGroupNotSupported - The agent group parameter is not supported.
    - agentPasswordNotSupported - The agent password parameter is not supported.

- agentStateNotSupported - The agent state is not supported.
- alertTimeNotSupported - The alertTime parameter is not supported.
- allocationNotSupported - The allocation parameter is not supported.
- authorisationCodeNotSupported - The authorization code is not supported.
- autoAnswerNotSupported - The autoAnswer parameter is not supported.
- bitRateNotSupported - The bitRate parameter is not supported.
- buttonNotSupported - The button parameter is not supported.
- callTypeNotSupported - The callType parameter is not supported.
- charactersToSendNotSupported - The charactersToSend parameter is not supported.
- connectionRateNotSupported - The connectionRate parameter is not supported.
- connectionReservationNotSupported. The connectionReservation parameter is not supported.
- consultPurposeNotSupported - The consultPurpose parameter is not supported.
- correlatorDataNotSupported - The correlator data parameter is not supported.
- delayToleranceNotSupported - The delayTolerance parameter is not supported.
- destinationDetectNotSupported - The destinationDetect parameter is not supported.
- digitModeNotSupported - The digitMode parameter is not supported.
- errorValueNotSupported - The errorValue parameter is not supported.
- flowDirectionNotSupported - The flowDirection parameter is not supported.
- forwardingDestinationNotSupported - The forwarding destination parameter is not supported.
- lampNotSupported - The lamp parameter is not supported.
- monitorTypeNotSupported - The monitor type is not supported.
- numberOfChannelsNotSupported - The numberOfChannels parameter is not supported.
- parameterTypeNotSupported - The participationType parameter is not supported.
- priorityNotSupported - The priority parameter is not supported.
- privateDataNotSupported - The privateData parameter is not supported.
- pulseDurationNotSupported - The pulseDuration parameter is not supported.
- pulseRateNotSupported - The pulseRate parameter is not supported.
- remainRetryNotSupported - The remainRetry parameter is not supported.
- ringCountNotSupported - The ringCount parameter is not supported.
- routeUsedNotSupported - The routeUsed parameter is not supported.
- securityNotSupported - The security parameter is not supported.
- switchingSubDomainCCIETypeNotSupported - The switchingSubDomainCCIEType parameter is not supported.
- toneDurationNotSupported - The toneDuration parameter is not supported.
- securityNotSupported - The security parameter is not supported.
- sysStatRegIDNotSupported - The sysStatRegID parameter is not supported.
- userDataNotSupported - The userData parameter is not supported.

- privilegeViolationSpecifiedDevice - Performing the service request would result in a privilege violation.

  - privilegeViolationActiveDevice - The request would violate a switching function restriction on the device activeCall that limits the device in some way.
  - privilegeViolationCalledDevice - Performing the service request would violate a switching function restriction that limits the called device in some way.
  - privilegeViolationCallingDevice - Performing the service request would violate a switching function restriction that limits the calling device in some way.
  - privilegeViolationCallToParkDevice - The service request would violate a switching function restriction on the device in callToPark connection that limits the device in some way.
  - privilegeViolationDestinationDevice - The request would violate a switching function restriction on the device newDestination that limits the device in some way.
  - privilegeViolationOnDivertingDevice - The service request would violate a switching function restriction on the diverting device in some way.
  - privilegeViolationHeldDevice - The request would violate a switching function restriction on the device in heldCall that limits the device in some way.
  - privilegeViolationOnParkToDevice - The service request would violate a switching function restriction on the device parkTo that limits the device in some way.
  - privilegeViolationPickupDevice - The request would violate a switching function restriction on the device in callToBePickedUp that limits the device in some way.
- routeingTimerExpired - The routeing timer or delayed ringback timer expired for a routeing request.
- requestIncompatibleWithObject - The service request is not compatible with the corresponding object specified in the service request. This error shall not reflect state incompatibility errors of an object.
  - requestIncompatibleWithConnection - The service request is not compatible with a connection specified in the service definition.
    - requestIncompatibleWithActiveConnection - The request is incompatible with the activeCall connection.
    - requestIncompatibleWithHeldConnection - The request is incompatible with the heldCall connection.
  - requestIncompatibleWithDevice - The service request is not compatible with a device specified in the service request.
    - requestIncompatibleWithCalledDevice - The service request is not compatible with the called device.
    - requestIncompatibleWithCallingDevice - The service request is not compatible with the calling device.
    - requestIncompatibleWithSubjectDevice - The service request is not compatible with the subject device (not a called or calling device).
      - requestIncompatibleWithActiveDevice - The service request is incompatible with the device in the activeCall connection.
      - requestIncompatibleWithCallToParkDevice - The service request is not compatible with the device in callToPark connection.
      - requestIncompatibleWithDestinationDevice - The service request is incompatible with the device in the newDestination connection.
      - requestIncompatibleWithDivertingDevice - The service request is incompatible with the device in the callToBeDiverted connection.

        - requestIncompatibleWithHeldDevice - The service request is incompatible with the device in the heldCall connection.
        - requestIncompatibleWithMedia - The media type associated with the message is incompatible with the associated device.
        - requestIncompatibleWithParkToDevice - The service request is not compatible with parkTo.
        - requestIncompatibleWithPickupDevice - The service request is incompatible with the device in the callToBePickedUp connection.
- serviceNotSupported - The service is not supported.
- securityViolation - The service request violates security.
- valueOutOfRange - A parameter (other than a CSTA object) has a value that is not in the enumeration or range specified for that parameter.
    - agentStateOutOfRange - An agent state is not one of the defined values.
    - alertTimeOutOfRange - The alertTime parameter has a value that is out of its permitted range.
    - allocationOutOfRange - The allocation parameter has a value that is out of its permitted range.
    - autoAnswerOutOfRange - The autoAnswer parameter has a value that is out of range.
    - bitRateOutOfRange - The bitRate parameter value is out of the defined range.
    - callTypeOutOfRange - The callType parameter value is out of the defined range.
    - connectionRateOutOfRange - The connectionRate parameter value is out of the defined range.
    - connectionReservationOutOfRange - The connectionReservation parameter has a value that is out of range.
    - consultPurposeOutOfRange - The consultPurpose parameter has a value that is out of its permitted range.
    - correlatorDataOutOfRange - The length of the correlator data exceeds the maximum length that the switching function supports.
    - delayToleranceOutOfRange - The delayTolerance parameter is out of the defined range.
    - destinationDetectOutOfRange - The destinationDetect parameter has a value that is out of its permitted range.
    - digitModeOutOfRange - The digitMode parameter value is out of the defined range.
    - doNotDisturbOutOfRange- The do not disturb setting in the doNotDisturb parameter is out of range.
    - flowDirectionOutOfRange - The flowDirection parameter is out of the defined range.
    - forwardingFlagOutOfRange - The forwarding flag is out of range.
    - forwardingTypeOutOfRange - The forwardingType parameter is not one of the defined values.
    - hookswitchComponentOutOfRange - A hookswitch component is not one of the defined components.
    - hookSwithTypeOutOfRange - A hookswitch type is out of range.
    - lampModeOutOfRange - A lamp mode setting is out of range.
    - messageWaitingSettingOutOfRange - A message waiting setting is out of range. The switching function returns this error for the messageWaitingOn parameter.
    - micGainOutOfRange - A microphone gain setting is out of range.
    - micMuteOutOfRange - A microphone mute setting is out of range.

        - monitorTypeOutOfRange - The monitor type is not a defined value.
        - numberOfChannelsOutOfRange - The numberOfChannels parameter is out of the defined range.
        - participationTypeOutOfRange - the participationType parameter has a value that is out of range.
        - pulseDurationOutOfRange- The pulseDuration parameter value is out of range.
        - pulseRateOutOfRange - The pulseRate parameter value is out of range.
        - ringCountOutOfRange - The ring count is out of range.
        - ringPatternOutOfRange - A ring patterns setting is out of range.
        - ringVolumnOutOfRange - A ring volume is out of range.
        - routingAlgorithmOutOfRange - The routeSelAlgorithm is not one of the defined values.
        - speakerMuteOutOfRange - A speaker mute setting is out of range.
        - speakerVolumeOutOfRange - A speaker volume is out of range.
        - switchingCcittType - The switchingSubDomainCCIEType parameter is out of the defined range.
        - systemStatusOutOfRange - The system status is not one of the defined values.
        - toneCharacterOutOfRange - One of more characters in the charctersToSend parameter is not in the permitted set.
        - toneDurationOutOfRange - The toneDuration parameter value is out of range.

12.2.12.2 Security Errors

Error values in this category shall indicate a security error. This category shall include one of the following specific error values:

- generic - This is a general purpose value that can be used when the server is unable to be any more specific about the cause of the error.
    - sequenceNumberViolated - Indicates that the server has detected an error in the operation’s message sequence number.
    - timeStampViolated - Indicates that the server has detected an error in the operation’s time stamp.
    - securityInfoViolated - Indicates that the server has detected an error in the operation’s security data.

12.2.12.3 State Incompatibility Errors

Error values in this category shall indicate that the service request was not compatible with the condition of a related CSTA object. This category shall include one of the following specific error values:

- generic - This is a general purpose value that can be used when the server is unable to be any more specific about the cause of the error.
    - invalidObjectState - An object (device, connection, call, message) is in an incorrect state for the service. This error value may be used when the server cannot be any more specific.
        - invalidDeviceState - A device object is in an incorrect state for the service request.
            - connectedCallExists - A physical element is already associated with another connection in the connected state.
            - invalidActiveDeviceState - The device in activeCall or callToBePickedUp connection is not in the correct state.
            - invalidCalledDeviceState - The device in the calledDevice connection is not in the correct state.
            - invalidCallingDeviceState - The device in the callingDevice connection is not in the correct state.

        - invalidCallToParkDeviceState - The device in the callToPark connection is not in the correct state.
        - invalidDestinationDeviceState - The newDestination device is not in the correct state.
        - invalidDivertingDeviceState - The diverting device is not in a correct state.
        - invalidHeldDeviceState - The device in heldCall connection is not in the correct state.
        - invalidParkToDeviceState - The parkTo device is not in the correct state.
    - invalidConnectionState - A connection object is in an incorrect state for the service request.
        - invalidActiveConnectionState - The activeCall connection is not in the correct state.
        - invalidConnectionIdentifierForActiveCall - A Connection Identifier specified as the activeCall in the service request is not in the correct state.
        - invalidHeldConnectionState - The heldCall connection is not in the correct state.
        - noActiveCall - The service request operates on an active call, but there was no active call.
        - noCallToAnswer - There is no call active for the connection identifier specified as the callToBeAnswered.
        - noCallToClear - There is no call associated with the connection identifier of the Clear Call request.
        - noCallToComplete - There is no call active for the connection Identifier specified as the callToBeCompleted.
        - noConnectionToClear - There is no connection for the connection identifier specified as the connectionToBeCleared.
        - noHeldCall - The service request operates on a held call, but the specified call was not in the Hold state.
    - incorrectMessageState - A message object is in an incorrect state for the service.
        - beginningOfMessage - The message pointer is at the beginning of the message.
        - endOfMessage - The message pointer is at the end of the message.
        - messageSuspended - The specified message is already suspended on the same Connection.
        - notAbleToPlay - The specified message exists, but cannot be played.
        - notAbleToResume - The specified message cannot be resumed.

12.2.12.4 System Resource Availability Errors

Error values in this category shall indicate that the service request could not be fulfilled because of a lack of system resources within the serving sub-domain. This category shall include one of the following specific error values:

- generic - This is a general purpose value that can be used when the server is unable to be any more specific about the cause of the error.
    - resourceBusy - The service is supported by the server, but is unavailable due to a resource that is busy.
        - internalResourceBusy - An internal resource is in use.
            - classifierBusy - All available classifiers are in use.
            - noMediaChannelsAvailable - There are no available media stream channels to complete the request.
                - channelsInUseForBridgedDevices - All applicable media stream channels are in use by other devices associated in a bridged device configuration.

            - channelsInUseForData - All applicable media stream channels are in use for digital data connections.
        - toneDetectorBusy - All available tone detectors are in use.
        - toneGeneratorBusy - All available tone generators are in use.
    - networkBusy - The server sub-domain is busy.
- resourceOutOfService - The service is supported by the server, but is unavailable due to a resource that is out of service.
    - deviceOutOfService - A device that is needed to carry out the service is out of service.
        - activeDeviceOutOfService - The device specified in activeCall connection is out of service.
        - calledDeviceOutOfService - The device specified in the calledDevice connection is out of service.
        - callingDeviceOutOfService - The device specified in the callingDevice connection is out of service.
        - callToParkDeviceOutOfService - The device specified in callToPark connection is out of service.
        - destinationDeviceOutOfService - The newDestination device is out of service.
        - divertingDeviceOutOfService - The device specified as the diverting device is out of service.
        - heldDeviceOutOfService- The device specified in heldCall connection is out of service.
        - parkToDeviceOutOfService - The device specified in parkTo is out of service.
        - pickupDeviceOutOfService - The device in callToBePickedUp is out of service.
        - divertingDeviceOutOfService - The device specified as the diverting device is out of service.
    - networkOutOfService - The server sub-domain is Out Of Service.
    - otherResourceOutOfService - Some resource needed to carry out the service other than the above is out of service.
- resourceLimitExceeded - The service is supported by the server, but is unavailable because it would exceed the internal usage limit of the resource.
    - overallMonitorLimitExceeded - The service request would exceed a switching function limit on the number of monitors (either an overall limit on the aggregate number of monitors or a limit on the number of monitors of different types (device-type, call-type) or some combination of the two).
    - conferenceMemberLimitExceeded - The requested service would exceed the server’s limit on the number of members of a conference.
    - registrationLimitExceeded - This service would exceed the switching function’s maximum number of registrations.

12.2.12.5 Subscribed Resource Availability Errors

Error values in this category shall indicate that the service request could not be fulfilled because a required resource must be purchased or contracted by the client system. This category shall include one of the following specific error values:

- generic - This is a general purpose value to be used when the server is unable to be any more specific about the cause of the error.
    - objectMonitorLimitExceeded - The service request would exceed the server’s limit of monitors for the specified object.
    - trunkLimitExceeded - The service request would exceed the server’s limit of trunks.

- outstandingRequestsLimitExceeded - The service request would exceed the servers's limit on the number of outstanding service requests.
- objectRegistrationLimitExceeded - This service request would exceed the switching function's limit on the number of registrations for this device.

12.2.12.6 Performance Management Errors

Error values in this category shall indicate that an error has been returned as a performance management mechanism. This category shall include one of the following specific error values:

- generic - This is a general purpose value to be used when the server is unable to be any more specific about the cause of the error.
  - performanceLimitExceeded - A performance limit has been exceeded.

12.2.12.7 Private Data Information Errors

Error values in this category shall indicate an error in the CSTA Private Data Information. The reason(s) why the Private Data Information is incorrect is not relevant to this CSTA Standard. This category shall include the following specific error value:

- cSTAPrivateDataInfoError - An error occurred in the privateData parameter. The reason for this error is implementation specific.

12.2.12.8 Unspecified Errors

Error values in this category shall indicate that the error did not belong to any of the other error value categories. This category shall include the following error value:

- unspecifiedError - Some error other than those covered by the error categories has occurred or server cannot determine the category of the error.

12.2.13 EventCause

The EventCause parameter type provides additional information on why an event was generated.

Format

Event causes are defined within the context of an event. For a description of an event cause refer to the event cause description associated with a specific event.

This parameter type contains one of the following event causes:

- ACD Busy
- ACD Forward
- ACD Saturated
- Active Participation
- Alert Time Expired
- Alternate
- Auto Work
- Blocked
- Busy
- Call Back
- Call Cancelled
- Call Forward
- Call Forward - Busy
- Call Forward - Immediate

- Call Forward - No Answer
- Call Not Answered
- Call Pickup
- Camp On
- Camp On Trunks
- Character Count Reached
- Conference
- Consultation
- Destination Detected
- Destination Not Obtainable
- Destination Out of Order
- Distributed
- Distribution Delay
- Do Not Disturb
- DTMF Digit Detected
- Duration Exceeded
- End of Message Detected
- Entering Distribution
- Forced Pause
- Forced Transition
- Incompatible Destination
- Intrude
- Invalid Account Code
- Invalid Number Format
- Join Call
- Key Operation
- Key Operation In Use
- Lockout
- Maintenance
- Make Call
- Make Predictive Call
- Message Duration Exceeded
- Message Size Exceeded
- Multiple Alerting
- Multiple Queuing
- Network Congestion
- Network Dialling
- Network Not Obtainable

- Network Out of Order
- Network Signal
- New Call
- Next Message
- No Available Agents
- Normal
- Normal Clearing
- No Speech Detected
- Not Available Bearer Service
- Not Supported Bearer Service
- Number Changed
- Number Unallocated
- Overflow
- Override
- Park
- Queue Cleared
- Recall
- Redirected
- Remains in Queue
- Reorder Tone
- Reserved
- Resources Not Available
- Selected Trunk Busy
- Silent Participation
- Single Step Conference
- Single Step Transfer
- Speech Detected
- Suspend
- Switching Function Terminated
- Termination Character Received
- Timeout
- Transfer
- Trunks Busy
- Unauthorized Bearer Service

Functional Requirements

1. Event causes are only present in events that result from the feature/situation associated with the meaning of the cause. Once that feature or situation ceases to be active, then the event cause is no longer present in the events.

12.2.14 LocalConnectionState

The LocalConnectionState parameter type describes the connection state of the device associated with the Monitor Cross Reference ID.

This parameter type is only applicable for events generated by device-type monitors.

Format

This parameter type shall contain one of the following connection states:

- Alerting
- Connected
- Fail
- Hold
- Initiated
- Null
- Queued

Refer to 6.1.5, “Connection”, for detailed descriptions of the connection states.

Example

The following is a scenario that illustrates the usage of this parameter.

Consider the case of a two device call where device one has called device two, and device two is ringing. If both devices are monitored, then the switching function generates two separate (Delivered) events to indicate that the call has been delivered. While the subject device is identical for both Delivered events, the localConnectionInfo parameters are different.

- Both events contain the same subject device, device two in this case, since that is the device in the call being alerted.
- The connection state for device one is connected (most likely listening to ringback). This is reported in the localConnectionInfo parameter of the Delivered event for device one.
- The connection state for device two is alerting. This is reported in the localConnectionInfo parameter of the Delivered event for device two.

12.2.15 MediaCallCharacteristics

The MediaCallCharacteristics parameter type specifies the media (voice, digital data, etc.) characteristics of the call.

Format

This parameter type is comprised of the following:

1. mediaClass (M) Bitmap - Specifies the media class (voice, digital data, etc.).

A CSTA call shall belong to at least one and may belong to more than one of the following classes:

- **Audio** - 3.1 KHz audio. Calls in this class involve devices that are used to make audio calls excluding speech calls. This includes calls involving devices such as G3 FAX and facsimile machines.
- **Data** - This class of calls involve digital data calls (both circuit switched and packet switched). Calls in this class include devices such as digital computer interfaces and G4 facsimile machines.
- **Image** - Digital data calls involving imaging, or high-speed, circuit-switched data in general. This includes calls involving devices such as digital video telephones and CODECs.
- **Voice** - Speech calls. This class of calls involves devices such as standard telephones.

- **Not Known** - The media class is not known.
- **Other** - A class of call not in the Data, Image, Audio or Voice classes.

2. connectionRate (O) Value - The digital data connection rate of the call. The contents of this parameter is switching function specific (the capability exchange services may be used to obtain the list of possible values that are supported by the switching function). A value of zero (0) indicates that the type of media stream associated with the connection is digital data but the connection rate is unknown.
3. bitRate (O) Enumerated - The digital data bit rate of the call. If this parameter is not present, the bit rate of the call is a constant bit. The following is the complete set of possible values:
    - Constant (Default) - A bit rate which ensures a dedicated bandwidth and a constant rate of media stream delivery.
    - Variable - A bit rate which may variable during the life of the call.
4. delayTolerance (O) Value - The digital data delay tolerance of the call. This parameter specifies the maximum amount of media stream delivery delay that will be toleranced for the call. If the bit rate is constant, then this value will indicate the actual amount of media stream delivery delay for the life of the call. Where as if the bit rate is variable, it will be the maximum delay allowed during the life of the call. The contents of this parameter is switching function specific, use the capability exchange services to obtain the list of possible values that are supported by the switching function. If this parameter is not present, the delay tolerance of the call is not known.
5. switchingSubDomainCCIEType (O) Enumerated - The type of switching sub-domain private call control information elements that are present in the switchingSubDomainInformationElements parameter. If this parameter is not present, there are no information elements associated with the call and the switchingSubDomainInformationElements parameter should be ignored. The following is the complete set of possible values:
    - ISDN
    - ATM (B-ISDN)
    - ISO-Ethernet (TDM part only)
    - RSVP
    - Other (switching sub-domain specific)
6. switchingSubDomainInformationElements (C) Characters - These parameters contain the private information elements that are available from the switching sub-domain (as specified by switchingSubDomainCCIEType) which represents a specific set of information elements. The format, meaning and behaviour of these information elements are specific to the given switching function. This parameter is only present and mandatory when the switchingSubDomainCCIEType parameter is present.

12.2.16 MediaServiceType

The mediaServiceType parameter type is used to indicate which media service is to be (or has been) attached to or detached from a particular call or connection that the computing function is controlling and/or monitoring.

Format

This parameter type shall contain one of the following values:

- cstaVoiceUnit
- dataModem
- digitalDataIsochronousIeee1394
- digitalDataIsochronousGeoport
- digitalDataIsochronousIeeeAtm

- digitalDataIsochronousIeeeIsdn
- digitalDataAPI
- ectfS100MediaServicesDefault
- ectfS100MediaServicesASI
- ivrScript1
- ivrScript2
- ivrScript3
- ivrScript4
- ivrScript5
- ivrScript6
- ivrScript7
- ivrScript8
- ivrScript9
- ivrScript10
- liveSoundCaptureAnalog
- liveSoundTransmitAnalog
- liveSoundCaptureIeee1394
- liveSoundTransmitIeee1394
- liveSoundCaptureTransmitGeoport
- liveSoundCaptureTransmitAtm
- liveSoundCaptureTransmitISDN
- soundCaptureTransmitADPCM
- soundCaptureTransmitApi
- usb
- sfSpecific1
- sfSpecific2
- sfSpecific3
- sfSpecific4
- sfSpecific5
- sfSpecific6
- sfSpecific7
- sfSpecific8
- sfSpecific9
- sfSpecific10

Refer to Table 6-10 on page 57 for a description of these service types.

12.2.17 MonitorFilter

The MonitorFilter parameter type specifies the list of events that are filtered (not sent) for a specific monitor.

Format

The parameter type consists of a list of bitmaps, each entry corresponding to a category of events, each bit in each category corresponding to a CSTA event. If the bit is TRUE, then the event corresponding to the bit is filtered (not sent) for the monitor. The list of bitmaps include:

- call control events - bitmap of the call control events as specified in Table 17-138 on page 267.
- call associated events - bitmap of the call associated events as specified in Table 18-22 on page 343.
- media attachment events - bit map of the media attachment events as specified in Table 19-14 on page 362.
- physical device events - bitmap of the physical device events as specified in Table 21-58 on page 418.
- logical device events - bitmap of the logical device events as specified in Table 22-54 on page 463.
- maintenance events - bit map of the maintenance events as specified in Table 23-1 on page 481.
- voice unit events - bit map of the voice unit events as specified in Table 26-38 on page 537.
- vendor specific (private) events - bit map of the vendor specific events as specified in Table 28-12 on page 564.

12.2.18 ServicesPermitted

The ServicesPermitted parameter type specifies the set of services that the switching function permits to be applied to a connection.

The servicesPermitted parameter (when provided in a call event) is similar to the localConnectionInfo parameter in that it applies to the services permitted for the connection at the monitored device.

This parameter type is only applicable for events generated by device-type monitors.

Format

This parameter type is a list of bitmaps where each bit represents a service that can be applied to a connection. When a bit is set, the corresponding service is permitted. The following is the list of bitmaps (multiple bits may be set in this parameter):

- call control services - the call control services as specified in Table 17-1 on page 185.
- call associated services - the call associated services as specified in Table 18-1 on page 331.
- media attachment services - the media attachment services as specified in Table 19-1 on page 354.
- routeing services - the routeing services as specified in Table 20-7 on page 371.
- voice unit services - the voice unit services as specified in Table 26-1 on page 517.

Functional Requirements

1. This parameter indicates which of a subset of CSTA services are permitted.
2. When the servicesPermitted parameter is provided in an event, it applies to the connection at the monitored device. This may or may not be the same as the subject device.
3. If there are multiple connections at a device, the information reported in the servicesPermitted parameter may not accurately reflect all possible service restrictions and interactions between multiple connections at a device.
4. There may be situations in a switching function that cause a service to fail after being presented as permitted in the servicesPermitted parameter. This may be due to dynamic system and/or resource conditions that may cause service availability restrictions. The switching function shall provide the appropriate error code in the negative acknowledgement to the failed service request.

12.2.19 SimpleCallState

The SimpleCallState parameter type indicates the main call states in simplified encoding. The semantics are identical to the sequence of connection states but they are represented by an item from the list below.

Format

This parameter type may contain one of the following:

- callNull
- callPending
- callOriginated
- callDelivered
- callDeliveredHeld
- callReceived
- callEstablished
- callEstablishedHeld
- callReceivedOnHold
- callEstablishedOnHold
- callQueued
- callQueuedHeld
- callFailed
- callFailedHeld
- callBlocked

12.2.20 SystemStatus

The SystemStatus parameter type indicates the reason for the System Status service request.

Format

The complete set of possible values are:

- *Disabled* - Existing Monitor Requests have been disabled. Other requests and acknowledgements also may be disabled, but negative acknowledgements should always be provided.
- *Partially Disabled* - Some of the objects in the system can not be reached. Existing monitors on these objects will not provide events and computer requests targeting these objects will be rejected. This cause indicates to the receiving function that a degradation of service level may occur but not complete system disability. Automatic or manual actions may be taken to remedy the parts disabled.
- *Enabled* - Requests and acknowledgements have been enabled. This usually occurs after a disruption or restart. This status cause is always sent after an Initializing cause has been sent and may be sent under other conditions. This status indicates that there are no outstanding monitors (existing monitors and their associated monitor cross reference identifiers are no longer valid).
- *Initializing* - The system is initializing or restarting. This status indicates that a system is temporarily unable to respond to any requests. If provided, this status message is followed by an Enable status message to indicate that the initialization process has completed.
- *Messages Lost* - Requests and/or acknowledgements, including event reports, may have been lost.
- *Normal* - May be sent at any time and indicates that the status is normal. This status has no effect on other Services.
- *Overload Imminent* - The receiver is requested to take initiative to shed load.
- *Overload Reached* - The requester may take initiative to shed load. This cause may be followed by Stop Monitor requests sent to the client and by rejections to additional service requests.

- *Overload Relieved* - The overload condition has passed.

12.2.21 TimeInfo

The TimeInfo parameter type provides the calendar date and the time of day. There are three possible value representations: as local time, coordinated universal time, or as local time with a time differential factor. All representations use a four character representation of the year.

Format

This parameter type is based upon the GeneralizedTime as defined in ISO/IEC 8824:1990 (Information technology - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)).

12.2.22 UserData

The UserData parameter type contains computing sub-domain to computing sub-domain data. Note that the capabilities exchange services return the maximum length of the user data for a switching function.

User Data is described further in 6.1.4.4, “User Data”, on page 29. Also, refer to 12.2.9, “CorrelatorData”, on page 87.

This specification defines a mechanism for delivering both user data and correlator data through an external ISDN network at the same time. This mechanism is described in Annex B.

Format

This parameter type is an octet string. The maximum length supported by the switching function is provided via the capabilities exchange services, but is limited to 256 octets.

Functional Requirements

1. The ability to send User Data, the timing of when user data can be sent, and the size of user data, is dependent upon the switching function’s capabilities and the underlying network (such as ISDN).
2. Unlike correlator data, User Data is not attached to a call for the life of the call. User Data that has been associated with a primary or secondary call does not get retained with a resulting conference or transferred call.
3. The switching function reflects the delivery of User Data in the call control events that result from the switching function or network carrying out the call control activity with which the User Data was associated. When the switching function receives user data independent of call activity (i.e., Send User Information service), the User Data is provided in the Call Information event.
4. User data addresses a specific user in a call (e.g. the initially called device). The delivery and propagation of the user data to other devices inside the switching sub-domain in regards to features that apply to the call (e.g. forwarding, do not disturb) is switching function dependent.
5. When this data is being attached to a call through a service request, its association shall be completed prior to any state transitions resulting from the request. Thus any state transition events which contain this parameter shall contain the information passed on the request.
6. If a computing function issues a Snapshot Call service after a service request has been issued with this information, but prior to the switching function making any state transitions, it is switching function specific as to what will be returned in the positive acknowledgement with regards to this parameter.

12.3 Identifier Parameter Types

Identifier parameter types specific to these specifications are summarized in the following table.

Table 12-2 Identifier Parameter Types Summary

Defined Parameter Type	Description	Pg.
12.3.1 AgentID	Identifies an ACD agent.	108
12.3.2 AssociatedCalledDeviceID	Describes the switching function's internal representation of the originally called device in a call.	108
12.3.3 AssociatedCallingDeviceID	Describes the switching function's internal representation of the calling device in the call when the calling device is outside the switching sub-domain (i.e., trunk number).	108
12.3.4 AuditoryApparatusID	Indicates the auditory apparatus containing the speaker whose volume has changed.	109
12.3.5 ButtonID	Specifies the button identifier on a device.	109
12.3.6 CalledDeviceID	Specifies the device to be called via a service. This parameter describes the originally called device associated with a call.	109
12.3.7 CallingDeviceID	Describes the calling device associated with the call.	110
12.3.8 CDRCrossRefID	Specifies the CDR services cross reference identifier.	110
12.3.9 ConnectionID	Describes a device's connection in a given call.	110
12.3.10 DCollCrossRefID	Used to identify a specific data collection.	112
12.3.11 DeviceID	Identifies or represents a device in the switching function.	112
12.3.12 DisplayID	Specifies the display identifier on a device.	112
12.3.13 EscapeRegisterID	Used to identify an escape services registration.	112
12.3.14 HookswitchID	Used to specify the hookswitch to query at a specified device.	112
12.3.15 IOCrossRefID	Specifies the I/O services cross reference identifier.	113
12.3.16 IORegisterReqID	Used to identify an I/O services registration.	113
12.3.17 LampID	Specifies the lamp identifier.	113
12.3.18 MediaServiceInstanceID	Identifies a particular media access service instance (e.g., specific media access server or subsystem)	113
12.3.19 MediaStreamID	Specifies a media stream identifier that can be used to access an attached media service.	113
12.3.20 MessageID	Specifies a particular Voice Unit message.	114
12.3.21 MonitorCrossRefID	Specifies an identifier that is used to correlate an event to an established monitor.	114
12.3.22 NetworkCalledDeviceID	Specifies the called device information provided by the network for external incoming calls.	114
12.3.23 NetworkCallingDeviceID	Specifies the calling device information provided by the network for external incoming calls.	114
12.3.24 RedirectionDeviceID	Describes the last device known by the switching function from which the current call was routed.	115
12.3.25 RingerID	Specifies the ringer identifier associated with a physical device	116
12.3.26 RouteingCrossRefID	References the routeing dialogues initiated by the switching function within a routeing registration.	116
12.3.27 RouteRegisterReqID	Identifies a routeing registration for which the computing function (acting as a routeing server) will receive routeing requests.	116
12.3.28 ServiceCrossRefID	Specifies an identifier that is used to correlate one service request to another service request.	116
12.3.29 SubjectDeviceID	Describes the device where a telephony event occurred or was invoked.	117
12.3.30 SysStatRegisterID	Used to identify system status registration.	117

12.3.1 AgentID

The AgentID parameter type identifies an ACD agent.

Format

This parameter type is a character string with a maximum length of 32.

12.3.2 AssociatedCalledDeviceID

For outgoing external calls, the AssociatedCalledDeviceID parameter type specifies the Network Interface Device (e.g., trunk, CO Line) within the switching sub-domain that is associated with the originally called device. This parameter in mandatory on all events dealing with external outgoing calls.

For incoming external calls, this parameter specifies a device within the switching sub-domain that is associated with the originally called device (such as a switching function internal representation of DNIS, for example). This parameter is optional on all events dealing with incoming external calls.

Format and Status

This device identifier type may be one of the following (see Clause 10, “CSTA Device Identifier Formats”, on page 75 for more information):

- a character string when using the Diallable Digits and the Switching Function Representation formats.
- an integer value when using the Device Number format.
- a value of “Not Known”

Functional Requirements

1. A device identifier of this type will only be present when the switching sub-domain is using a network interface device for an external call; that is, the call is an External Outgoing or External Incoming call.
2. A device identifier of this type is not used to provide DNIS (Dialed Number Identification Service) or DID (Direct Inward Dialing) digit information or a string of digits that represents the called device. (This information is provided in the corresponding CalledDeviceID parameter.)
3. A device identifier of this type is set to “Not Known” when the switching function does not know the Network Interface Device associated with the original called device.
4. A device identifier of this type will never contain the value “Not Required” or “Not Specified”.

12.3.3 AssociatedCallingDeviceID

The AssociatedCallingDeviceID parameter type specifies the Network Interface Device (e.g., trunk, CO line) within the switching sub-domain that is associated with the calling device in the call if the call is an external incoming call. This parameter shall be included on all external incoming calls.

Format and Status

This device identifier type may be one of the following (see Clause 10, “CSTA Device Identifier Formats”, on page 75 for more information):

- a character string when using the Diallable Digits and the Switching Function Representation formats.
- an integer value when using the Device Number format.
- a value of “Not Known”

Functional Requirements

1. A device identifier of this type will only be present when the switching function is using a network interface device for an inbound call; that is, the call is an external incoming call.

2. A device identifier of this type is not used to provide ANI (Automatic Number Identification), CLID (CallerID) or SID (Station Identification) digit information or a string of digits that represents the calling device. (This information is provided in the corresponding CallingDeviceID parameter.)
3. A device identifier of this type will never contain the value “Not Required” or “Not Specified”.
4. If a call is created that contains multiple AssociatedCallingDeviceIDs (i.e., a conference call calling back to a device), the AssociatedCallingDeviceID status shall be “Not Known”.

12.3.4 AuditoryApparatusID

The AuditoryApparatusID parameter type specifies a particular auditory apparatus associated with the device.

Format

This parameter type is an octet string with a maximum length of four.

12.3.5 ButtonID

The ButtonID parameter type specifies the button identifier on a device.

Format

This parameter type is an octet string with the maximum length of four.

Table 12-3 Reserved Button ID Assignments

Button ID	Button Label
0-9	Keypad Digits: “0” through “9”
10	Keypad Symbol: “*”
11	Keypad Symbol: “#”

12.3.6 CalledDeviceID

A device identifier of the CalledDeviceID type describes the originally called device associated with a call.

Format and Status

This device identifier type may be one of the following (see Clause 10, “CSTA Device Identifier Formats”, on page 75 for more information):

- a character string when using the Diallable Digits and the Switching Function Representation formats.
- an integer value when using the Device Number format.
- a value of “Not Known”.

Functional Requirements

1. A device identifier of this type contains the originally called device in the call. For External Incoming calls, a device identifier of this type will contain DNIS (Dialed Number Identification Service) or DID (Direct Inward Dialing).
2. This parameter will never contain the value “Not Required” or “Not Specified”.
3. When two calls are being joined through a conference or transfer, the CalledDeviceID information for the resulting call shall be taken from the secondary call.
4. This parameter type is different from the NetworkCalledDeviceID parameter type in that the CalledDeviceID information may change if the call is transferred and/or conferenced whereby the NetworkCalledDeviceID information does not change as long as the NID associated with the original calling device remains in the call. Also, the NetworkCalledDeviceID is limited to information passed over a Network Interface Device.

12.3.7 CallingDeviceID

The CallingDeviceID parameter type specifies the calling device associated with the call.

Format and Status

This device identifier type may be one of the following (see Clause 10, “CSTA Device Identifier Formats”, on page 75 for more information):

- a character string when using the Diallable Digits and the Switching Function Representation formats.
- an integer value when using the Device Number format.
- a value of “Not Known”.

Functional Requirements

1. A device identifier of this type contains the calling device in the call. For External Incoming calls, a device identifier of this type will contain ANI (Automatic Number Identification), CLID (CallerID) or SID (Station Identification) digit information or a string of digits that represents the calling device.
2. This parameter will never contain the value “Not Required” or “Not Specified”.
3. If more than one device is the calling device in a call (i.e., a conference call calling back to a device), the CallingDeviceID status will be “Not Known”.
4. This parameter type is different from the NetworkCallingDeviceID parameter type in that the CallingDeviceID information may change if the call is transferred and/or conferenced whereby the NetworkCallingDeviceID information does not change as long as the NID associated with the original calling device remains in the call. Also, the NetworkCallingDeviceID is limited to information passed over a Network Interface Device.

12.3.8 CDRCrossRefID

The CDRCrossRefID is used to correlate subsequent CDR services to the Start Call Detail Records Transmission service.

Format

This parameter type is an octet string with a maximum length of four.

12.3.9 ConnectionID

The ConnectionID parameter type describes a device’s connection in a given call. (Connection Identifiers are also discussed in 6.1.5, “Connection”, on page 31.)

Format

The ConnectionID is always comprised of the following parameters (except in special cases which are described below):

1. callID (M) Octet String - An identifier used by the switching function to represent a valid call. The maximum length of this ID is eight octets. These IDs are created by the switching function and are globally unique among all calls within the switching sub-domain.
2. deviceID (M) DeviceID - An identifier which is used to represent a device in the switching sub-domain. This identifier can be either one of the two following values:
   - *Static* - This type of identifier is defined in 6.1.3, “Device”, on page 6.
   - *Dynamic* - This type of identifier is one that is created by the switching function for a device when it enters into a call and shall remain constant for the life of the device’s participation in the call (i.e., the creation of a connection identifier for the device). As soon as the device leaves the call, the identifier becomes invalid. The use of a dynamic identifier by a switching function is determined when the

switching function does not have a static identifier for the device or the identifier can not uniquely identify the device in a call. This type of identifier is an octet string, with a maximum length of 32. It is never a diallable number and can never be used outside the context of the connection identifier. This type of identifier is not directly related to a device element but is strictly used to make the connection identifier unique. Refer to 6.1.8, “Management of Dynamically-Assigned Identifiers”, on page 38, for more information.

Functional Requirements

1. The computing function shall not fabricate its own Connection IDs. This will lead to unpredictable results.
2. The Connection IDs in events and service acknowledgements are always allocated by the switching function.
3. Computing functions can extract Device IDs from Connection IDs and use them on services that have Device ID parameters only if the Device ID extracted is a static Device ID that the switching function accepts. Otherwise, the Device ID cannot be used.
4. Computing functions shall extract Call IDs from Connection IDs, provided by the switching function, to correlate event reports associated with devices that are connected together in a call.
5. The computing function will always receive an event to indicate the termination of a Connection ID if the appropriate monitor is started. Refer to the individual services and events to better understand the meaning of individual events with respect to connection states.
6. If the computing function issues a service with a Connection ID that cannot be controlled by the switching function, the service will be rejected with a negative acknowledgement.
7. Connection IDs used as parameters can only have three formats:
    a. A *complete* Connection ID (i.e., call ID and device ID). This extracted from either events received by the computing function or positive acknowledgements received as a result of services issued.

       When supplied as a parameter, the Connection ID will be validated by the switching function with respect to the service being issued. If this Connection ID is not valid, the service request will be rejected with a negative acknowledgement.

    b. A *DeviceID only* Connection ID. If a service has more than one Connection ID parameter, the switching function supports this type of Connection ID, and the computing function wants to use this type of Connection ID, then all Connection ID parameters in the service shall be of this type.

       If this type of Connection ID is used as the Connection ID parameter for a service, then rules documented in the services sections will determine whether it is accepted or not by the switching function. If this type of Connection ID is not accepted, then the service will be rejected with a negative acknowledgement.

    c. A *Call ID only* Connection ID. In events, this format can only be used for the Call Cleared and Failed event. If this format is used for any service other than Clear Call, Monitor Start, or Snapshot Call, it will be rejected with a negative acknowledgement.
8. If a call changes its Call ID when a Conference or Transfer occurs, Connection IDs shall be provided to link the old Call IDs to the new Call IDs. When this occurs, the event will contain a list of originally known Connection IDs of devices that are still in the call along with the new replacement Connection IDs. When the new Connection IDs are created in such cases, new dynamic Device IDs may also be used to create the Connection IDs.
9. Connection IDs that come from the switching function (events and positive acknowledgement to services) will always contain both the Call ID and Device ID portions (see item 7a above) except for the Call Cleared and Failed events that may also contain only a valid call ID in the connection ID (see item 7c above).
10. The computing function should never assume the reuse of callIDs, although some switching functions may reuse one or the other.

12.3.10 DCollCrossRefID

The DCollCrossRefID parameter type identifies a specific data collection that was initiated via the Start Data Collection service. The DCollCrossRefID is valid only for the duration of the data collection.

Format

This parameter type is an octet string with a maximum length of four.

12.3.11 DeviceID

The DeviceID parameter type identifies or represents a device.

Format

This DeviceID parameter type consists of two components:

- Device Identifier - A mandatory component that specifies the device identifier as described in Clause 10, “CSTA Device Identifier Formats”, on page 75. This may include one of the following:
    - a character string when using the Diallable Digits and the Switching Function Representation formats.
    - an integer value when using the Device Number format.
- Media Class - An optional component that specifies the media class(s) of the device. This may be used for selecting a device based upon a particular media capability, for example. Refer to the mediaClass component in 12.2.15, “MediaCallCharacteristics”, on page 113 for the complete set of possible values. Note that multiple bits may be set.

The maximum length of the Device Identifier component that is supported by the switching function is provided via the capabilities exchange services.

Functional Requirements

1. For more details on DeviceID parameter types, refer to 6.1.3, “Device”.
2. For information on DeviceID in Connection Identifiers, refer to 12.3.9, “ConnectionID”, on page 110 and 6.1.5, “Connection”, on page 31.

12.3.12 DisplayID

The DisplayID parameter type specifies a particular display associated with the device.

Format

This parameter type is a string with the maximum length of four characters.

12.3.13 EscapeRegisterID

The EscapeRegisterID parameter type is used to identify an escape service registration.

Format

This parameter type is an octet string with the maximum length of four.

12.3.14 HookswitchID

The HookswitchID parameter type is used to specify the hookswitch to query at a specified device. If not provided, the default is to get the status of each hookswitch at the specified device.

Format

This parameter type is an octet string with the maximum length of four.

12.3.15 IOCrossRefID

The IOCrossRefID parameter type identifies each I/O data path. The computing function receives a IOCrossRefID in each I/O service request. The Start Data Path service initiates an I/O data path. The IOCrossRefID is only valid for the duration of the data path.

The IOCrossRefID is unique within the I/O registration (IORegisterReqID). Some switching functions may provide the additional benefit of a unique IOCrossRefID across the entire switching sub-domain. This is also the case if I/O registration is not supported by the switching function.

The parameter type also specifies if the switching function or the computing function started the data path.

Format

This parameter type shall be one of the following:

- switchProvided (octet string with a maximum length of four) - indicates that the switching function has started the data path
- computerProvided (octet string with a maximum length of four) - indicates that the computing function has started the data path

12.3.16 IORegisterReqID

The IORegisterReqID parameter type identifies a I/O registration for which the computing function (acting as an I/O server) will receive I/O service requests. This identifier may be associated with a particular device within the switching sub-domain or it may indicate that the computing sub-domain is the I/O server for all devices within the switching sub-domain. When the computing function uses the I/O Register service to register for I/O services, it receives a IORegisterReqID in the positive acknowledgement sent by the switching function. The IORegisterReqID is only valid until the I/O registration is ended by the computing function or switching function.

IORegisterReqID parameters are unique across a given CSTA service boundary.

Format

This parameter type is an octet string with a maximum length of four.

12.3.17 LampID

The LampID parameter type specifies the lamp identifier.

Format

This parameter type is an octet string with the maximum length of four.

12.3.18 MediaServiceInstanceID

The MediaServiceInstanceID parameter type identifies a particular media access service instance (e.g., specific media access server or subsystem).

Format

This parameter type is an octet string with a maximum length of 64.

12.3.19 MediaStreamID

The MediaStreamID parameter type specifies a media stream identifier that can be used to access an attached media service. The usage of the mediaStreamID is defined by a particular media service.

Format

This parameter type is an octet string. The maximum length supported by the switching function is provided via the capabilities exchange services.

12.3.20 MessageID

The MessageID parameter type specifies a particular Voice Unit message.

Format

This parameter type is an octet string. The maximum length supported by the switching function is provided via the capabilities exchange services.

12.3.21 MonitorCrossRefID

The MonitorCrossRefID parameter type specifies an identifier that is used to correlate an event to an established monitor. When a monitor is established using the Monitor Start service, a monitorCrossReferenceID parameter is returned as part of the positive acknowledgement message. This monitorCrossReferenceID parameter is included in every event for that specific monitor.

Format

This parameter type is an octet string with a maximum length of four.

Functional Requirements

1. This parameter is allocated by the switching function.
2. The switching function is responsible for providing unique monitorCrossReferenceID parameters over a specific service boundary.

12.3.22 NetworkCalledDeviceID

For external incoming calls, this parameter specifies the called device information that was provided by the Network over a Network Interface Device. For example, this may contain DNIS (Dialed Number Identification Service) or DID (Direct Inward Dialing) digit information or a string of digits that represents the called device.

This information is established when the call is first created and stays with the call as long as the Network Interface Device (NID) associated with the original calling device remains in the call, even if the call is transferred from the original called device, for example.

Format and Status

This device identifier type may be one of the following (see Clause 10, “CSTA Device Identifier Formats”, on page 75 for more information):

- a character string when using the Diallable Digits and the Switching Function Representation formats.
- an integer value when using the Device Number format.
- a value of “Not Known”.

Functional Requirements

1. This parameter will never contain the value “Not Required” or “Not Specified”.
2. This parameter type is different from the CalledDeviceID parameter type in that the CalledDeviceID information may change if the call is transferred and/or conferenced whereby the NetworkCalledDeviceID information does not change as long as the NID associated with the original calling device remains in the call. Also, the NetworkCalledDeviceID is limited to information passed over a Network Interface Device.

12.3.23 NetworkCallingDeviceID

For external incoming calls, this parameter specifies the calling device information that was provided by the Network over a Network Interface Device. For example, this may contain ANI (Automatic Number Identification), CLID (CallerID) or SID (Station Identification) digit information or a string of digits that represents the calling device.

This information is established when the call is first created and stays with the call as long as the Network Interface Device (NID) associated with the original calling device remains in the call, even if the call is transferred from the original called device, for example.

Format and Status

This device identifier type may be one of the following (see Clause 10, “CSTA Device Identifier Formats”, on page 75 for more information):

- a character string when using the Diallable Digits and the Switching Function Representation formats.
- an integer value when using the Device Number format.
- a value of “Not Known”.

Functional Requirements

1. This parameter will never contain the value “Not Required” or “Not Specified”.
2. This parameter type is different from the CallingDeviceID parameter type in that the CallingDeviceID information may change if the call is transferred and/or conferenced whereby the NetworkCallingDeviceID information does not change as long as the NID associated with the original calling device remains in the call. Also, the NetworkCallingDeviceID is limited to information passed over a Network Interface Device.

12.3.24 RedirectionDeviceID

The RedirectionDeviceID parameter type describes the last device known by the switching function from which the current call was routed. “Routed” includes forwarded from, diverted from, or redirected from.

Format and Status

This device identifier type may be one of the following (see Clause 10, “CSTA Device Identifier Formats”, on page 75 for more information):

- a character string when using the Diallable Digits and the Switching Function Representation formats.
- an integer value when using the Device Number format.
- “Provided” - indicates that the DeviceID of the last redirection device is provided
- “Not Known” - indicates that the call has been redirected but the switching function cannot provide the DeviceID
- “Not Required” - indicates that the current call has never been redirected during the existence of the call
- “Not Specified” - indicates that the switching function cannot determine whether or not the call has ever been redirected

Functional Requirements

1. The information in a device identifier of this type will stay with the call until the call is established. If the call is routed multiple time before it is established, then the information in this parameter will be updated to the last known device from which the call was routed. If the call was redirected from a device, but the device identifier is unknown, “Not Known” shall be used. Depending on the capabilities of the switching function, the last known device for the call might be reflected by only one device identifier and in actuality the call might have been routed several times before arriving at the final destination.

   Note that in the case of Immediate Forwarding, where forwarding is triggered *before* the call is delivered to a device, the lastRedirectionDevice in the event associated with the delivery of the call to a new device (after it was immediately forwarded) shall contain “Not Known”. Refer to 6.8.1, “Forwarding”, on page 51 for more information on this behaviour.

12.3.25 RingerID

Specifies the ringer identifier associated with a physical element.

A device can be associated with one or more ringers.

Format

This parameter type is an octet string with a maximum length of four.

12.3.26 RouteingCrossRefID

The routeingCrossRefID parameter type identifies each routeing dialogue. The computing function receives a routeingCrossRefID in each Route Request service request. The Route Request service initiates a routeing dialogue. The routeingCrossRefID is only valid for the duration of the routeing dialogue pertaining to a specific call.

The routeingCrossRefID is unique within the routeing registration (routeRegisterReqID). Some switching functions may provide the additional benefit of a unique routeing cross reference identifier across the entire switching sub-domain. This is also the case if routeing registration is not supported by the switching function.

Format

This parameter type is an octet string with a maximum length of four.

12.3.27 RouteRegisterReqID

The RouteRegisterReqID parameter type identifies a routeing registration for which the computing function (acting as a routeing server) will receive routeing requests. This identifier may be associated with a particular routeing device within the switching sub-domain or it may indicate that the computing sub-domain is the routeing server for all routeing devices within the switching sub-domain. When the computing function uses the Route Register service to register for routeing services, it receives a routeRegisterReqID in the positive acknowledgement sent by the switching function. The routeRegisterReqID is only valid until the routeing registration is ended by the computing function or switching function.

routeRegisterReqID parameters are unique across a given CSTA service boundary.

Format

This parameter type is a string with a maximum length of four.

12.3.28 ServiceCrossRefID

The ServiceCrossRefID parameter type specifies an identifier that is used to correlate one service request to another service request.

For example, a service may be specified to request information from a switching function using an asynchronous mechanism. In this case there would be a service request from the computing function requesting information. The switching function would return a ServiceCrossRefID in the positive acknowledgement to this request. The switching function would subsequently send messages in the form of Service Requests to the computing function that would contain the same ServiceCrossRefID that could be used to correlate the service request with the original service request.

Format

This parameter type is an octet string with the maximum length of four.

Functional Requirements

1. This parameter is allocated by the switching function.
2. The switching function is responsible for providing unique ServiceCrossRefIDs over a specific CSTA service boundary.

12.3.29 SubjectDeviceID

The SubjectDeviceID parameter type represents a device which is the focus of the action associated with the event being reported.

Format and Status

This device identifier type may be one of the following (see Clause 10, “CSTA Device Identifier Formats”, on page 75 for more information):

- a character string when using the Diallable Digits and the Switching Function Representation formats.
- an integer value when using the Device Number format.
- a value of “Not Known”.

12.3.30 SysStatRegisterID

The SysStatRegisterID parameter type is used to identify system status registration.

Format and Status

This parameter type is an octet string with the maximum length of four.

13 Capability Exchange Services

This clause describes the Capability Exchange Services.

13.1 Services

Table 13-1 Capability Exchange Services Summary

Capability Exchange Service	Description	Pg.
13.1.1 Get Logical Device Information	Obtains the current set of logical device information for a given device identifier.	119
13.1.2 Get Physical Device Information	Obtains the current set of physical device information for a given device identifier.	128
13.1.3 Get Switching Function Capabilities	Obtains the current set of capabilities for the entire switching function.	132
13.1.4 Get Switching Function Devices	Obtains the devices in the application working domain (i.e. devices that can be controlled and/or observed).	146
13.1.5 Switching Function Devices	Provides the actual list of devices in the application working domain (i.e. devices that can be controlled and/or observed).	148

13.1.1 Get Logical Device Information

C ⟶ S

The Get Logical Device Information service is used to obtain the current set of characteristics/capabilities associated with the logical element of a given device.

13.1.1.1 Service Request

Table 13-2 Get Logical Device Information—Service Request

Parameter Name	Type	M/ O/C	Description
device	DeviceID	M	Specifies the device being queried.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

13.1.1.2 Service Response

This service follows the atomic acknowledgement model for this request.

13.1.1.2.1 Positive Acknowledgement

Table 13-3 Get Logical Device Information—Positive Acknowledgement

Parameter Name	Type	M/ O/C	Description
deviceCategory	Enumerated	M	Specifies the device category (station, ACD device, etc.) of the device in the service request. The complete set of possible values is: • ACD • Group • Network Interface (e.g., trunk, CO line) • Park • Routeing Device • Station (default) • Voice Unit • Other
groupDeviceAttibutes	Bitmap	C	Specifies the group device attributes of the device being queried. If a bit is TRUE then the specified attribute is present. The following is the list of bits (multiple bits may be set): • ACD • Hunt • Pick • Other This parameter shall be provided if the deviceCategory is Group, otherwise it shall not be provided.

Table 13-3 Get Logical Device Information—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
namedDeviceTypes	Enumerated	O	If assigned by the switching function, this parameter indicates the named device type associated with the device in the service request. The complete set of possible values are: • ACD • ACD Group • Button • Button Group • Conference Bridge • Line • Line Group • Operator • Operator Group • Parking Device • Station • Station Group • Trunk • Trunk Group • Other • Other Group
shortFormDeviceID	DeviceID	O	Specifies an alternative (a shorter length, for example) device identifier that the switching function may use to reference the device in the service request.
hasPhysicalElement	Boolean	M	Specifies if the device has a physical element associated with this device identifier. The complete set of possible values is: • FALSE - The device does not have a physical element. • TRUE - The device does have a physical element. The device identifier in the service request should be used with the Get Physical Device Information service to obtain the physical element’s characteristics for this device.
acdModels	Bitmap	M	Specifies the type of ACD Model(s) that are present at this device. If a bit is TRUE, then the specified model is supported. The following is the list of bits (multiple bits may be set): • Visible ACD-related Devices • Non-Visible ACD-related Devices Note that these bits are valid when the device is an ACD device.

Table 13-3 Get Logical Device Information—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
agentLogOnModels	Bitmap	C	Specifies the types of agent log on models that are supported by the device. If a bit is TRUE, then the specified agent log on model is supported. The following is the list of bits (multiple bits may be set): • Log On to an ACD device • Log On to an ACD Group (explicit/one step) • Log On to an ACD Group (explicit/two steps) • Log On to an ACD Group (implicit/one step) Note that Log On to an ACD Group (implicit/one step) model cannot be simultaneously supported with the Log On to an ACD device model. The switching function shall provide this parameter if the agent log on model is configured by the switching function at the logical device element level (agent, ACD device, or ACD group), otherwise the parameter may or may not be provided.
appearanceAddressable	Boolean	M	Specifies whether the appearances of the logical element are addressable (via the Call Appearance “CA” string or the physical element extension “EXT” string in the Switching Function Representation Device Identifier format). The complete set of possible values is: • FALSE - The appearances are not addressable. • TRUE - The appearances are addressable
appearanceType	Enumerated	M	Specifies the type of appearances associated with the logical element. The complete set of possible values is: • Selected-Standard • Basic-Standard • Basic-Bridged • Exclusive-Bridged • Independent-Shared-Bridged • Interdependent-Shared-Bridged
appearanceList	List of Characters	C	Specifies the list of device identifier suffices for each of the appearances that are available at the logical element. This parameter is mandatory if the appearances are addressable and if it is a Selected-Standard or a Basic-Standard type. This list will only contain appearance suffices that can be observed and/or controlled within the switching sub-domain (via the Call Appearance string in the Switching Function Representation Device Identifier format).

Table 13-3 Get Logical Device Information—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
otherPhysicalDeviceList	List of DeviceIDs	C	Specifies the list of device identifiers for other devices with physical elements that are associated with the logical element appearance. This parameter is mandatory if the appearances are addressable and any type of bridged appearance. The Get Physical Device Information service should be used to obtain the physical element characteristics associated with these other devices. This list will only contain devices that can be either observed and/or controlled within the switching sub-domain. Note that, for a Hybrid configuration, the order of device identifiers in this list is the same as the order of devices in the appearanceList parameter. See Functional Requirement #3 in 10.1.2 for additional information.
miscMonitorCaps	Bitmap	O	Specifies the special types of monitoring considerations for this device. If a bit is TRUE then the monitoring consideration is associated with the device. The following is the list of bits (Multiple bits may be set): • Group Inclusive Model - the scope of the monitor on the group device includes the distribution mechanism and all member devices. This bit is only valid for group devices that include a distribution mechanism (e.g. Hunt and ACD groups). This bit shall not be set if the Group Exclusive Model bit is set. • Group Exclusive Model - the scope of the monitor on the group device includes only the distribution mechanism. This bit is only valid for group devices that include a distribution mechanism (e.g. Hunt and ACD groups). This bit shall not be set if the Group Inclusive Model bit is set • Monitor the physical element to report call control events for all appearances associated with a device. (Only a valid bit if the appearanceType is any form of bridge appearance.) (i.e. use the device identifiers from the otherPhysicalDeviceList). • ACD Device Inclusive - the scope of the monitor on an ACD device includes both the ACD device and the distributed-to devices (including ACD groups). (This capability is valid only for ACD devices). This bit shall not be set if the ACD Device Exclusive bit is set • ACD Device Exclusive - the scope of the monitor on an ACD device only the ACD device. (This capability is valid only for ACD devices). This bit shall not be set if the ACD Device Inclusive bit is set Note that if this parameter is not present, then the monitoring considerations are not known.
associatedGroupList	List of DeviceIDs	C	Specifies the list of device identifiers for all the other devices which are members of this group device. Use the appropriate capabilities exchange services to obtain the characteristics of these devices. This list shall only contain devices that can be either observed and/or controlled within the switching sub-domain. This parameter shall be provided when the device is a Group device. It may or may not be provided if the device is a member of a group and shall not be provided if the device is neither a Group device nor is a member of a group.

Table 13-3 Get Logical Device Information—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
maxCallbacks	Value	O	Specifies the maximum number of concurrent call back requests that can be outstanding for this device. If this parameter is not present, then the maximum number of concurrent callback requests is not known for the device.
maxAutoAnswerRings	Value	O	Specifies the maximum number of rings before a call is auto-answered at this device. If this parameter is not present, then the maximum number of Auto Answer rings is not known for the device.
maxActiveCalls	Value	O	Specifies the maximum number of concurrent calls that can be active at any one time for this device. If this parameter is not present, then the maximum number of active calls is not known for the device.
maxHeldCalls	Value	O	Specifies the maximum number of concurrent calls that can be held at any one time for this device. If this parameter is not present, then the maximum number of held calls is not known for the device.
maxFwdSettings	Value	O	Specifies the maximum number of user-specified settings (forwarding-type/forward-destination combinations) that can be activated at any one time for this device. If this parameter is not present, then the maximum number of activated user-specified settings is not known for the device.
maxDevicesInConf	Value	O	Specifies the maximum number of devices both within and outside the switching function that this device can conference into a call. If this parameter is not present, then the maximum number of devices in a conference is not known for the device. The minimum value that can be supplied for this value is 3.
transAndConfSetup	Bitmap	O	Specifies the different ways that this device can set up for a conference and/or transfer. (Note that if this parameter is not present, then the device can only set up transfers and conferences through the Consultation Call service.) If the bit is TRUE, then the specified way to setup a conference or transfer is supported by the switching function. The following is the list is of bits (multiple bits may be set): • Consultation Call • Hold Call - Make Call • Alternate Call • two calls in the initial state of Hold • two calls in the initial state of Connected
deviceOnDeviceMonitorFilter	MonitorFilter	C	Specifies the complete monitorFilter parameter that this device supports with respect to device-type monitoring. This parameter shall be provided if this form of device-type monitoring is supported, otherwise the parameter shall not be provided. The information in the monitor filter parameters used in the Get Logical Device Information and the Get Physical Device Information services should be the same when the same device identifier is used (assuming that the device identifier has a logical and a physical element).
deviceOnConnectionMonitorFilter	MonitorFilter	C	Specifies the complete monitorFilter parameter that connections at this device supports with respect to device-type monitoring. This parameter shall be provided if this form of device-type monitoring is supported, otherwise the parameter shall not be provided.

Table 13-3 Get Logical Device Information—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
callOnDeviceMonitorFilter	MonitorFilter	C	Specifies the complete monitorFilter parameter that this device supports with respect to call-type monitoring on a device. This parameter shall be provided if this form of call-type monitoring is supported, otherwise the parameter shall not be provided.
callOnConnectionMonitorFilter	MonitorFilter	C	Specifies the complete monitorFilter parameter that this device supports with respect to call-type monitoring for a connection at this device. This parameter shall be provided if this form of call-type monitoring is supported, otherwise the parameter shall not be provided.
mediaClassSupport	Bitmap	O	Specifies the media class of calls that the device can support. If a bit is TRUE then the specified type of call can be present at the device. The following is the list of bits (multiple bits may be set): • Audio • Data • Image • Voice • Other

Table 13-3 Get Logical Device Information—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
mediaServiceCapsList	List of Structures	C	Specifies a list of structures of the media service types, version, media service instances, connection modes supported. This parameter is a list, each element of which contains the following: • mediaServiceType (M) MediaServiceType - A media service type used to identify the media service. • mediaSeviceVersion (O) Value - The version of the media service. • mediaServiceInstance (O) MediaServiceInstanceID - A media service instance associated with the media service. • connectionModeBMap (O) Bitmap - The media service connection modes supported for the media service type, version and instance. The following is the list of bits (multiple bits may be set): • consultationConference • consultationConferenceHold • deflect • directedPickup • join • singleStepConference • singleStepConferenceHold • singleStepTransfer • transfer • direct • mediaStreamIDSupported (M) Boolean - Specifies if the mediaStreamID is supported for the combination of media service type, version, and instance. The complete set of values is: • TRUE - indicates that the switching function shall provide the conditional mediaStreamID parameter where specified. • FALSE - indicates that the conditional mediaStreamID is not provided. This parameter shall be provided if the device is capable of media access and shall not be provided otherwise.
connectionRateList	List of Values	O	Specifies the list of connection rates that are supported for this device.
delayToleranceList	List of Values	O	Specifies the list of delay Tolerances that are supported for this device.
numberOfChannels	Value	O	Specifies the number of available channels at this device. If the parameter is not present, the number of channels at the device is not known but it is one or greater.
maxChannelBind	Value	O	Specifies the maximum number of channels that can be associated with a given connection at a device. If the parameter is not present, the maximum number of channels per connection is one.

Table 13-3 Get Logical Device Information—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
routeingServList	RouteingServList	C	Specifies a list of bitmaps. Each bitmap entry represents a Routeing service that is supported by the device (both service requests to and from the switching function, when the service is bi-directional). This includes the following categories of services: • Routing Services This parameter shall be provided if the switching function supports any of these categories of services for this device. If a Routeing service's bitmap entry is not included in the list, then the service is not supported by the switching function. Note that the Routeing Mode feature is grouped with Logical Device features.
logDevServList	Structure	C	Specifies a list of capability bitmap parameter types corresponding to categories of services. Each bitmap entry in the lists represents a service that applies to a logical device that is supported by the device. This includes the following categories of services: • callControlServList (O) CallControlServList - specifies the list of call control services supported. • callAssociatedServList (O) CallAssociatedServList - specifies the list of call associated services supported. • logicalServList (O) LogicalServList - specifies the list of logical device feature services supported. • mediaServList (O) MediaServList - specifies the list of media services supported. • ioServicesServList (O) IOServicesServList - specifies the list of I/O services supported. • dataCollectionServList (O) DataCollectionServList - specifies the list of data collection services supported. • voiceUnitServList (O) VoiceUnitServList - specifies the list of voice unit services supported. This parameter shall be provided if the switching function supports at least one of these categories of services for this device. If a logical device service's bitmap entry is not included in the list, then the service is not supported by the device.

Table 13-3 Get Logical Device Information—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
logDevEvtsList	Structure	C	Specifies a list of capability bitmap parameter types corresponding to categories of events. Each bitmap entry in the lists represents an event that applies to a logical device that is supported by the device. This includes the following categories of events: • callControlEvtsList (O) CallControlEvtsList - specifies the list of call control events supported. • callAssociatedEvtsList (O) CallAssociatedEvtsList - specifies the list of call associated events supported. • logicalEvtsList (O) LogicalEvtsList - specifies the list of logical device feature events supported. • mediaEvtsList (O) MediaEvtsList - specifies the list of media events supported. • voiceUnitEvtsList (O) VoiceUnitEvtsList - specifies the list of voice unit events supported. This parameter shall be provided if the switching function supports any of these categories of events for this device. If a logical device event's bitmap entry is not included in the list, then the event is not supported by the device.
deviceMaintEvtsList	DeviceMaintEvtsList	C	Specifies a list of bitmaps. Each bitmap entry represents a device maintenance event that is supported by the device. This includes the following categories of services: • Device Maintenance events This parameter shall be provided if the switching function supports any of these categories of events for this device. If a device maintenance's bitmap entry is not included in the list, then the event is not supported by the switching function.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

13.1.1.2.2 Negative Acknowledgement

The negative acknowledgement error codes are described in 12.2.12, "ErrorValue", on page 88.

13.1.1.3 Operational Model

13.1.1.3.1 Connection State Transitions

There are no connection state changes as the result of this service.

13.1.1.3.2 Device-Type Monitoring Event Sequences

There are no events generated as a result of this service.

13.1.1.3.3 Call-Type Monitoring Event Sequences

There are no events generated as a result of this service.

13.1.1.3.4 Functional Requirements

1. This service shall be rejected with a negative acknowledgement (i.e., Error Value of Object Not Known), if the device does not contain a logical element.

13.1.2 Get Physical Device Information

$C \longrightarrow S$

The Get Logical Device Information service is used to obtain the current set of characteristics/capabilities associated with the physical element of a given device.

13.1.2.1 Service Request

Table 13-4 Get Physical Device Information—Service Request

Parameter Name	Type	M/ O/C	Description
device	DeviceID	M	Specifies the device being queried.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

13.1.2.2 Service Response

This service follows the atomic acknowledgement model for this service request.

13.1.2.2.1 Positive Acknowledgement

Table 13-5 Get Physical Device Information—Positive Acknowledgement

Parameter Name	Type	M/ O/C	Description
deviceCategory	Enumerated	M	Specifies the device category (station, ACD device, etc.) of the device being queried. The complete set of possible values is: • ACD • Group • Network Interface (i.e., trunk) • Park • Routeing • Station (default) • Voice Unit • Other
groupDeviceAttributes	Bitmap	C	Specifies the group device attributes of the device being queried. If a bit is TRUE then the specified attribute is present. The following is the list of bits (multiple bits may be set): • ACD • Hunt • Pick • Other This parameter shall be provided if the deviceCategory is Group, otherwise it shall not be provided.

Table 13-5 Get Physical Device Information—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
namedDeviceTypes	Enumerated	O	If assigned by the switching function, this parameter indicates the named device type associated with the device being queried. The complete set of possible values are: • ACD • ACD Group • Button • Button Group • Conference Bridge • Line • Line Group • Operator • Operator Group • Parking Device • Station • Station Group • Trunk • Trunk Group • Other • Other Group
hasLogicalElement	Boolean	M	Specifies if the device has a logical element associated with this device identifier. The complete set of possible values is: • FALSE - The device does not have a logical element. • TRUE - The device does have a logical element. The device identifier in the service request should be used with the Get Logical Device Information service to obtain the logical element’s characteristics for this device.
otherLogicalDeviceList	List of Device IDs	O	Specifies the list of device identifiers for other devices with logical elements that are associated with this device. The Get Logical Device Information service should be used to obtain the logical element characteristics associated with these other devices. This list will only contain devices that can be either observed and/or controlled within the switching function.
deviceModelName	Characters (64)	O	Specifies the switching function specific model name of the device. If this parameter is not present, then the model name is not known.
deviceOnDeviceMonitorFilter	MonitorFilter	C	Specifies the complete monitorFilter parameter that the device supports with respect to device-type monitoring. This parameter shall be provided if this form of device-type monitoring is supported, otherwise the parameter shall not be provided. The information in the monitor filter parameters used in the Get Logical Device Information and the Get Physical Device Information services should be the same when the same device identifier is used (assuming that the device identifier has a logical and a physical element).

Table 13-5 Get Physical Device Information—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
deviceOnConnectionMonitorFilter	MonitorFilter	C	Specifies the complete monitorFilter parameter that a connection at the device supports with respect to device-type monitoring. This parameter shall be provided if this form of device-type monitoring is supported, otherwise the parameter shall not be provided.
callOnDeviceMonitorFilter	MonitorFilter	C	Specifies the complete monitorFilter parameter that the device supports with respect to call-type monitoring on a device. This parameter shall be provided if this form of call-type monitoring is supported, otherwise the parameter shall not be provided.
callOnConnectionMonitorFilter	MonitorFilter	C	Specifies the complete monitorFilter parameter that the device supports with respect to call-type monitoring on a connection at the device. This parameter shall be provided if this form of call-type monitoring is supported, otherwise the parameter shall not be provided.
maxDisplays	Value	O	Specifies the maximum number of displays associated with the device being queried. If this parameter is not present, then the device either does not have any displays or the maximum number of displays at this device is not known.
maxButtons	Value	O	Specifies the maximum number of buttons associated with the device being queried. If this parameter is not present, then the device either does not have any buttons or the maximum number of buttons at this device is not known.
maxLamps	Value	O	Specifies the maximum number of lamps associated with the device being queried. If this parameter is not present, then the device either does not have any lamps or the maximum number of lamps at this device is not known.
maxRingPatterns	Value	O	Specifies the maximum number of ring patterns that the ringer has for the device being queried. If this parameter is not present, then the device either does not have a ringer or the maximum number of ring patterns at this device is not known.
physDevServList	PhysDevServList	C	Specifies a list of bitmaps. Each bitmap entry represents a Physical Device service that is supported by the specified device. This includes the following categories of services: • Physical Device Feature services This parameter shall be provided if the switching function supports any of these categories of services for this device. If a physical device service’s bitmap entry is not included in the list, then the service is not supported by the specified device.
physDevEvtsList	PhysDevEvtsList	C	Specifies a list of bitmaps. Each bitmap entry represents a Physical Device Event that is supported by the specified device. This includes the following categories of events: • Physical Device Feature events This parameter shall be provided if the switching function supports any of these categories of events for this device. If a physical device event’s bitmap entry is not included in the list, then the event is not supported by the specified device.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

13.1.2.2.2 Negative Acknowledgement

The negative acknowledgement error codes are described in 12.2.12, “ErrorValue”, on page 88.

13.1.2.3 Operational Model

13.1.2.3.1 Connection State Transitions

There are no connection state changes as the result of this service.

13.1.2.3.2 Device-Type Monitoring Event Sequences

There are no events generated as a result of this service.

13.1.2.3.3 Call-Type Monitoring Event Sequences

There are no events generated as a result of this service.

13.1.2.3.4 Functional Requirements

1. This service shall be rejected with a negative acknowledgement (i.e., Error Value of Object Not Known), if the device does not have a physical element.
2. In order to obtain the entire set of physical device capabilities and characteristics, the computing function shall also use the Get Button Information, Get Lamp Information, and Get Display services to collect the detailed information on the device’s buttons, lamps, and displays.

13.1.3 Get Switching Function Capabilities

C $\longrightarrow$ S

The Get Switching Function Capabilities service is used by the computing function to obtain the current set of capabilities for the entire switching function.

13.1.3.1 Service Request

Table 13-6 Get Switching Function Capabilities—Service Request

Parameter Name	Type	M/ O/C	Description
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

13.1.3.2 Service Response

This service follows the atomic acknowledgement model for this service request.

13.1.3.2.1 Positive Acknowledgement

Table 13-7 Get Switching Function Capabilities—Positive Acknowledgement

Parameter Name	Type	M/ O/C	Description
switchingSubDomainName	Character (64)	M	Specifies the name of switching sub-domain which distinguishes it from other switching sub-domains.
manufacturerName	Characters (64)	M	Specifies the name of the manufacturer of the switching sub-domain.
profiles	Bitmap	M	Specifies the CSTA Profiles supported by the switching function. The following is the list of the possible profiles (multiple bits may be set): • Basic Telephony Profile • Routing Profile Note that at least one profile shall be supported by the switching function.

Table 13-7 Get Switching Function Capabilities—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
deviceIDFormat	Bitmap	M	Specifies the types of device ID formats supported by the switching function. If a bit is TRUE, then the specified format is used by the switching function. The following is the list of the possible formats (multiple bits may be set): • Diallable Digits format - “*” • Diallable Digits format - “#” • Diallable Digits format - “A-D” • Diallable Digits format - “!” • Diallable Digits format - “P” • Diallable Digits format - “T” • Diallable Digits format - “,” • Diallable Digits format - “W” • Diallable Digits format - “@” • Diallable Digits format - “$” • Diallable Digits format - “;” • SF Representation format - “!” • SF Representation format - “&” • SF Representation format - “/” • SF Representation format - “%” • SF Representation format - “NM” • SF Representation format - Generic • SF Representation format - ImplicitTON • SF Representation format - PublicTON - unknown • SF Representation format - PublicTON - international number • SF Representation format - PublicTON - national • SF Representation format - PublicTON - subscriber • SF Representation format - Public TON - abbreviated • SF Representation format - PrivateTON - unknown • SF Representation format - PrivateTON - level 3 regional • SF Representation format - PrivateTON - level 2 regional • SF Representation format - PrivateTON - level 1 regional • SF Representation format - PrivateTON - local • SF Representation format - PrivateTON - abbreviated • SF Representation format - Other • Device Number format Note that the Diallable Digits format with the 0-9 characters shall be supported by the switching function.

Table 13-7 Get Switching Function Capabilities—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
swDomainFeatures	Bitmap	M	Specifies which features are supported by the switching function. If a bit is TRUE, then the specified feature is supported. The following is the list of possible features (multiple bits can be set): Forwarding Call Associated models: • Is (Immediate) Forwarding triggered before the call is logically delivered to the device? • Is (Immediate) Forwarding triggered after the call is logically delivered to the device? Level of Forwarding Default Settings: • Switching function default setting - allows activation/deactivation of a single switching function forwarding type/forwarding destination combination. • User specified settings - allows the setting of individual forwarding types and forwarding destinations. • User specified setting (default forwarding type) - If this is TRUE, when the forwarding type is omitted in the Set Forward service, the switching function applies a default value, otherwise there is no default value applied. • User specified setting (default forward destination) - If this is TRUE, when the forward destination is omitted in the Set Forward service, the switching function applies a default value, otherwise there is no default value applied. Connection Failure: • Negative Acknowledgement • Support of Failed event with an associated failed connection • Support of Failed event without an associated failed connection • Support of Failed event with an associated failed connection, not reported via monitors on the failing device Other: • Recall • Call Back • External Calls—Incoming Calls • External Calls—Outgoing Calls • Prompting
swAppearanceAddressability	Bitmap	M	Specifies what types of appearance addressability is available within the switching sub-domain. The following is the list of bits (multiple bits can be set): • addressable • non-addressable

Table 13-7 Get Switching Function Capabilities—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
swAppearanceTypes	Bitmap	M	Specifies what types of appearances are available within the switching sub-domain The following is the list of bits (multiple bits can be set): • Selected-Standard • Basic-Standard • Basic-Bridged • Exclusive-Bridged • Independent-Shared-Bridged • Interdependent-Shared-Bridged
ignoreUnsupportedParameters	Enumerated	M	Specifies how the switching function handles unsupported optional parameters in service requests. The complete set of possible values is: • Ignore parameters - This indicates that the switching function treats unsupported optional parameters as if they were not present. • Reject Request - This indicates that the switching function returns a negative acknowledgement in response to any requests that contain unsupported optional parameters.
callCharacteristicsSupported	Bitmap	C	Specifies the characteristics that the switching function reports via the callCharacteristics parameter. If a bit is TRUE then the specified characteristic is reported. The following is the list of bits (multiple bits may be set): • acdCall • priorityCall • maintenanceCall • directAgent • assistCall • voiceUnitCall This parameter shall be provided if the switching function characterizes calls via the callCharacteristics parameter.
mediaClassSupport	Bitmap	O	Specifies the media class of calls the switching sub-domain can support. If a bit is TRUE then the specified type of call can be present within the switching function. The following is the list of bits (multiple bits may be set): • Audio • Data • Image • Voice • Other If this parameter is not present, then the switching function only supports voice calls.

Table 13-7 Get Switching Function Capabilities—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
numberOfChannels	Value	O	Specifies the highest number of available channels at a given device within the switching sub-domain. If the parameter is not present, the number of channels at a device is not known but is one or greater.
maxChannelBind	Value	O	Specifies the highest maximum number of channels that can be associated with a given connection at a device within the switching sub-domain. If the parameter is not present, the maximum number of channels per connection is one.
miscMediaCallCharacteristics	Bitmap	O	Specifies the media call characteristics supported. If a bit is TRUE then the specified feature is present within the switching function. The following is the list of bits (multiple bits can be set) • Does the switching function support the adjustment of the media characteristics when a call is being made?
connectionRateList	List of Values	O	Specifies the list of connection rates that are supported for the given switching function.
delayToleranceRateList	List of Values	O	Specifies the list of delay tolerances that are supported for the given switching function.
pauseTime	Value (1..2000)	O	Specifies the amount time that a pause (as specified by the comma “,” character in the Diallable Digits Device Identifier format) within a dialling sequence will last for in the switching function. This time is specified in milliseconds. If this parameter is not present, then the pause time is not known for the switching function.
currentTime	TimeInfo	O	Specifies the current date and time of the switching function.
messageSeqNumbers	Bitmap	C	Specifies if the switching function supports message sequence numbers (via the security parameter) on services and events. If a bit is TRUE, then it is supported. The following is the list of bits (multiple bits may be set): • allEvents - message sequence number is provided on all events from the switching function. • allAcks - message sequence number is provided on all (positive and negative) acknowledgements from the switching function. • allServReqs - message sequence number is provided on all service requests from the switching function. This parameter shall be provided if the switching function provides message sequence number information.

Table 13-7 Get Switching Function Capabilities—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
timeStampMode	Bitmap	C	Specifies when the switching function provides timestamp information (via the security parameter). If a bit is TRUE, then the mode is supported. The following is the list of bits (multiple bits may be set): • allEvents - timestamp parameter is provided on all events from the switching function. • allAcks - timestamp parameter is provided on all (positive and negative) acknowledgements from the switching function. • allServReqs - timestamp parameter is provided on all service requests from the switching function. This parameter shall be provided if the switching function provides timestamp information.
securityMode	Enumerated	C	Specifies when the switching function provides securityInfo (via the security parameter). The following is the list of bits (multiple bits may be set): • allEvents - securityInfo is provided on all events from the switching function. • allAcks - securityInfo is provided on all (positive and negative) acknowledgements from the switching function. • allServReqs - securityInfo is provided on all service requests from the switching function. This parameter shall be provided if the switching function provides securityInfo via the security parameter.
securityFormat	Bitmap	C	Specifies the format(s) of the securityInfo information (in the security parameter) supported by the switching function. The following is the list of bits (multiple bits may be set): • octetStringFromSF - the switching function provides securityData in the octetString format • otherTypeFromSF - the switching function provides securityData in another format. • octetStringToSF - the switching function supports receiving securityData in the octetString format • otherTypeToSF - the switching function supports receiving securityData in another format. This parameter shall be provided if the switching function supports sending or receiving securityInfo in the security parameter.

Table 13-7 Get Switching Function Capabilities—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
privateDataFormat	Bitmap	C	Specifies the format(s) of the privateData information supported by the switching function. The following is the list of bits (multiple bits may be set): • octetStringFromSF - the switching function provides privateData in the octetString format • otherTypeFromSF - the switching function provides privateData in another format. • octetStringToSF - the switching function supports receiving privateData in the octetString format • otherTypeToSF - the switching function supports receiving privateData in another format. This parameter shall be provided if the switching function supports sending or receiving privateData.
transAndConfSetup	Bitmap	O	Specifies the different ways that the switching function can set up for a conference and/or transfer. (Note that if this parameter is not present, then the switching function can only set up transfers and conferences through the Consultation Call service.) If the bit is TRUE, then the specified way to setup a conference or transfer is supported by at least one device in the switching sub-domain. The following is the list is of bits (multiple bits may be set): • Consultation Call • Hold Call - Make Call • Alternate Call • two calls in the initial state of Hold • two calls in the initial state of Connected
deviceOnDeviceMonitorFilter	MonitorFilter	C	Specifies the complete monitorFilter parameter that is supported by the switching function when the monitorObject is a device and the monitorType is a device. Each bitmap entry represents an event that is supported by at least one of the devices in the switching function. This parameter shall be provided if this form of device-type monitoring is supported by at least one of the devices in the switching function, otherwise it shall not be provided.
deviceOnConnectionMonitorFilter	MonitorFilter	C	Specifies the complete monitorFilter parameter that is supported by the switching function when the monitorObject is a connection and the monitorType is a device. Each bitmap entry represents an event that is supported by at least one of the devices in the switching function. This parameter shall be provided if this form of device-type monitoring is supported by at least one of the devices in the switching function, otherwise it shall not be provided.

Table 13-7 Get Switching Function Capabilities—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
callOnDeviceMonitorFilter	MonitorFilter	C	Specifies the complete monitorFilter parameter that is supported by the switching function when the monitorObject is a device and the monitorType is a call. Each bitmap entry represents an event that is supported by at least one of the devices in the switching function. This parameter shall be provided if this form of call-type monitoring is supported by at least one of the devices in the switching function, otherwise it shall not be provided.
callOnConnectionMonitorFilter	MonitorFilter	C	Specifies the complete monitorFilter parameter that is supported by the switching function when the monitorObject is a connection and the monitorType is a call. Each bitmap entry represents an event that is supported by at least one of the devices in the switching function. This parameter shall be provided if this form of call-type monitoring is supported in the switching function, otherwise it shall not be provided.
miscMonitorCaps	Bitmap	O	Specifies the special types of monitoring capabilities that are present within the switching sub-domain. If a bit is TRUE then the monitoring capability is present within the switching sub-domain. The following is the list of bits (multiple bits may be set): • Group Inclusive Model - the scope of the monitor on a group device includes the distribution mechanism and all member devices. This bit applies to group devices that include a distribution mechanism (e.g. Hunt and ACD groups). • Group Exclusive Model - the scope of the monitor on the group device includes only the distribution mechanism. This bit applies to group devices that include a distribution mechanism (e.g. Hunt and ACD groups). • Monitor the physical element to report call control events for all appearances associated with a device. (This capability is valid only if an appearanceType of any form of a bridge appearance is supported.) • ACD Device Inclusive - the scope of the monitor on an ACD device includes both the ACD device and the distributed-to devices (including ACD groups). (This capability is valid only for ACD devices). • ACD Device Exclusive - the scope of the monitor on an ACD device only the ACD device. (This capability is valid only for ACD devices). If this parameter is not present, then the monitoring considerations are not known.

Table 13-7 Get Switching Function Capabilities—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
correlatorDataSupported	Boolean	O	Specifies if the switching function supports the correlatorData parameter on service requests and events. The complete set of possible values is: • TRUE - Option supported. • FALSE - Option is not supported. Refer to “Correlator Data” on page 28 for the required events that shall support correlator data if this option is supported.
dynamicFeatureSupported	Enumerated	O	Specifies how the switching function provides the servicesPermitted parameter on events. The complete set of possible values is: • none - servicesPermitted not provided on any events • all - servicesPermitted provided on all events where it is specified • some - servicesPermitted provided on some events. Refer to the logDevEvtsList parameter for the events that support the parameter.
acdModels	Bitmap	O	Specifies the types of ACD models that are supported by the switching function. If a bit is TRUE, then the specified ACD model is supported by the switching function. The following is the list of bits (multiple bits may be set): • Visible ACD-Related Devices • Non-Visible ACD-Related Devices Note that if more than one type of ACD model is present in the switching function, then the Get Logical Device Information service shall be used to determine the particular ACD models supported by for each ACD device or ACD group by the switching function.
agentLogOnModels	Bitmap	O	Specifies the types of agent log on models that are supported by the switching function. If a bit is TRUE, then the specified agent log on model is supported by the switching function. The following is the list of bits (multiple bits may be set): • Log On to an ACD device • Log On to an ACD Group (explicit/one step) • Log On to an ACD Group (explicit/two steps) • Log On to an ACD Group (implicit/one step) Note that if more than one type of model is present in the switching function, then the Get Logical Device Information service shall be used to determine the particular Log On models supported for each device which is or can be associated with an agent.

Table 13-7 Get Switching Function Capabilities—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
agentStateModels	Bitmap	O	Specifies the types of agent models that are supported by the switching function. If a bit is TRUE, then the specified agent model is supported by the switching function. The following is the list of bits (multiple bits may be set): • Agent Multi-State Model • Agent Multi-State Model (Semi-Independent Linked) • Agent Oriented Model Note that if more than one type of model is present in the switching function, then the Get Logical Device Information service shall be used to determine the particular Agent models supported for each ACD device or ACD group by the switching function.
connectionView	Enumerated	M	Specifies the meaning of the primary and secondary old call parameters in the Conferenced and Transferred events. The complete set of possible values is: • fixed view - the contents of the primary and secondary old call parameters are independent of the monitoring type and the role of the device in the conference or transfer. • local view - the contents of the primary and secondary old call parameters are dependent upon which device is being monitored. Refer to the descriptions of the Conferenced and the Transferred events for more information.
maxLengthParameters	List of Values	M	Each value is the switching function's maximum length (in octets/characters) for the corresponding parameters and parameter types. The computing function should not send larger data or the service request will be rejected. The following list provides the different parameters and parameter types for which a maximum value is provided. The number in parenthesis specifies the maximum possible length. • AccountInfo parameter type: (32) • AuthCode parameter type: (32) • AgentID parameter type: (32) • AgentPassword parameter type: (32) • callID in the ConnectionID parameter type: (8) • CorrelatorData parameter type: (32) • CSTAPrivateData parameter type: (any value) • Device Identifiers parameter type: (128) • UserData parameter type: (256) • buttonLabel parameters: (64) • lampLabel parameters: (64) • charsToBeSent parameter: (64) If any of the above values is zero, then the parameter or parameter type is not supported.

Table 13-7 Get Switching Function Capabilities—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
servEvtsList	List	C	Specifies a list of capability bitmap parameter types corresponding to categories of services. Each bitmap entry in the lists represents a service or event that is supported by the switching function. This includes the following categories of services/events: • capExchangeServList (O) CapExchangeServList • systemStatusServList (O) SystemStatusServList • monitoringServList (O) MonitoringServList • snapshotServList (O) SnapshotServList • callControlServList (O) CallControlServList • callControlEvtsList (O) CallControlEvtsList • callAssociatedServList (O) CallAssociatedServList • callAssociatedEvtsList (O) CallAssociatedEvtsList • mediaServList (O) MediaServList • mediaEvtsList (O) MediaEvtsList • routeingServList (O) RouteingServList • physServList (O) PhysServList • physEvtsList (O) PhysEvtsList • logicalServList (O) LogicalServList • logicalEvtsList (O) LogicalEvtsList • deviceMaintEvtsList (O) DeviceMaintEvtsLis • ioServicesServList (O) IOServicesServList • dataCollectionServList (O) DataCollectionServList • voiceUnitServList (O) VoiceUnitServList • voiceUnitEvtsList (O) VoiceUnitServList • cdrServList (O) CDRServList • vendorSpecificServList (O) VendorSpecificServList • vendorSpecificEvtsList (O) VendorSpecificEvtsList This parameter shall be provided if the switching function supports any of these categories of services/ events. If a list entry is not included in the list, then the corresponding category of services/events is not supported by the switching function. The bitmap parameter types are described in Annex C.
privateDataVersionList	List of Values	O	If the switching function supports the private data mechanism, this parameter provides the list of supported private data versions associated with the switching function manufacturer (manufacturerName).

Table 13-7 Get Switching Function Capabilities—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
systemStatusTimer	Value	C	Specifies a timer value indicating how often the switching function sends periodic system status requests to the computing function (i.e. heartbeats). This parameter has a value between 0 and 180 seconds. 0 means that the switching function does not send periodic System Status service requests. This parameter shall be provided if the switching function supports the heartbeat timer via the System Status service.
simpleThreshold	Value	O	Specifies the number of unacknowledged service requests that are allowed at any time for the switching function. 0 means there is no limit. If this parameter is not provided, the simpleThreshold is unknown.
filterThreshold	List of Values	O	Specifies a list of values representing the number of outstanding service requests, for every service defined in this specification, that are allowed at any time for each service. 0 means there is no limit. If this parameter is not provided, the filterThreshold is unknown.

Table 13-7 Get Switching Function Capabilities—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
mediaServiceCapsList	List of Structures	C	Specifies a list of structures of the media service types, version, media service instances, connection modes supported across the entire switching function. This parameter is a list, each element of which contains the following: • mediaServiceType (M) MediaServiceType - A media service type used to identify the media service. • mediaSeviceVersion (O) Value - The version of the media service. • mediaServiceInstance (O) MediaServiceInstanceID - A media service instance associated with the media service. • connectionModeBMap (O) Bitmap - The media service connection modes supported for the media service type, version and instance. The following is the list of bits (multiple bits may be set): • consultationConference • consultationConferenceHold • deflect • directedPickup • join • singleStepConference • singleStepConferenceHold • singleStepTransfer • transfer • direct • mediaStreamIDSupported (M) Boolean - Specifies if the mediaStreamID is supported for the combination of media service type, version, and instance. The complete set of values is: • TRUE - indicates that the switching function shall provide the conditional mediaStreamID parameter where specified. • FALSE - indicates that the conditional mediaStreamID is not provided. This parameter shall be provided if media access is supported by at least one of the devices in the switching function, otherwise it shall not be provided.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

13.1.3.2.2 Negative Acknowledgement

The negative acknowledgement error codes are described in 12.2.12, “ErrorValue”, on page 88.

13.1.3.3 Operational Model

13.1.3.3.1 Connection State Transitions

There are no connection state changes as the result of this service.

13.1.3.3.2 Device-Type Monitoring Event Sequences

There are no events generated as a result of this service.

13.1.3.3.3 Call-Type Monitoring Event Sequences

There are no events generated as a result of this service.

13.1.4 Get Switching Function Devices

C ⟶ S

The Get Switching Function Devices service is used by the computing function to obtain the current set of devices in the application working domain along with their associated device categories and associated device names.

13.1.4.1 Service Request

Table 13-8 Get Switching Function Devices—Service Request

Parameter Name	Type	M/ O/C	Description
requestedDeviceID	DeviceID	O	Specifies the device identifier of the device being queried. If this parameter is not present, the switching function returns a list of all devices (of the requestedDeviceCategory, if that parameter is provided) in the switching sub-domain.
requestedDeviceCategory	Enumerated	O	Specifies that only devices of the requested category be provided. If this parameter is not present, the switching function will return a list of all devices (or just the requested device) in the switching function. The complete set of possible values is: • ACD • Group (ACD) • Group (Hunt) • Group (Pick) • Group (Other) • Network Interface • Park • Routeing Device • Station • Voice Unit • Other
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

13.1.4.2 Service Response

This service follows the multi-step acknowledgement model for this service request.

13.1.4.2.1 Positive Acknowledgement

The positive acknowledgement for the Get Switching Function Devices service indicates that one or more Switching Function Devices services will subsequently be generated by the switching function.

The computing function shall associate subsequent Switching Function Devices services to the original Get Switching Function Devices service by means of the serviceCrossRefID parameter.

Table 13-9 Get Switching Function Devices—Positive Acknowledgement

Parameter Name	Type	M/ O/C	Description
serviceCrossRefID	ServiceCrossRefID	M	Specifies the correlator used to associate subsequent Switching Function Devices services to this service request.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

13.1.4.2.2 Negative Acknowledgement

The negative acknowledgement error codes are described in 12.2.12, “ErrorValue”, on page 88.

13.1.4.3 Operational Model

13.1.4.3.1 Connection State Transitions

There are no connection state changes as the result of this service.

13.1.4.3.2 Device-Type Monitoring Event Sequences

There are no events generated as a result of this service.

13.1.4.3.3 Call-Type Monitoring Event Sequences

There are no events generated as a result of this service.

13.1.4.3.4 Functional Requirements

1. Due to the nature of the switching function configuration, there may be a significant time interval between the generation of the positive acknowledgement for this service and the generation of the first Switching Function Devices service (or between a sequence of Switching Function Device services).
2. If there are no devices that meet the conditions in the service request, the switching function provides a Positive Acknowledgement to the Get Switching Function Devices service request followed by a Switching Function Devices service that includes the deviceList parameter with no devices.
3. The Switching Function Devices service returns all device identifiers that refer to all devices in the application working domain (i.e devices that can be controlled and/or observed).

13.1.5 Switching Function Devices

S $\longrightarrow$ C

The Switching Function Devices service is used by the switching function to provide a list of devices in the application working domain. This service is generated as a result of the Get Switching Function Devices service.

The switching function may generate a sequence of Switching Function Devices services, individually referred to as segments, in response to a single Get Switching Function Devices service request.

13.1.5.1 Service Request

Table 13-10 Switching Function Devices—Service Request

Parameter Name	Type	M/ O/C	Description
serviceCrossRefID	ServiceCrossRefID	M	Specifies the cross reference used to associate the Switching Function Devices service request to the Get Switching Function Devices service request.
segmentID	Value	O	Specifies the segment number of this message. Each successive segment number in the sequence increments the segmentID by one.
lastSegment	Boolean	M	Specifies if this segment is the last one associated with the serviceCrossRefID. The complete set of possible values is: • TRUE - Indicates that this is the last segment • FALSE - Indicates that this is not the last segment in the sequence.

Table 13-10 Switching Function Devices—Service Request (continued)

Parameter Name	Type	M/ O/C	Description
deviceList	List of Structures	M	Specifies the list of device Identifiers representing the devices that can be controlled and/or observed. It includes the following components for each device in the list: • deviceID (M) DeviceID - Specifies a Device Identifier associated with the entry in the list. • deviceCategory (O) Enumerated - Specifies the device category associated with the entry in the list. The complete set of possible values is: • ACD • Group • Network Interface (e.g. trunk, CO Line) • Park • Routeing • Station (default) • Voice Unit • Other • namedDeviceTypes (O) Enumerated - indicates the named device type associated with the device in the service request. The complete set of possible values is: • ACD • ACD Group • Button • Button Group • Conference Bridge • Line • Line Group • Operator • Operator Group • Parking Device • Station • Station Group • Trunk • Trunk Group • Other • Other Group

Table 13-10 Switching Function Devices—Service Request (continued)

Parameter Name	Type	M/ O/C	Description
(continued)	(continued)		• deviceAttributes (O) Bitmap - Specifies additional attributes of the device associated with the entry in the list. This is a bit list of the following set of possible values (multiple bits may be set): • mediaAccessDevice - indicates that the device is also a media access device • routeingDevice - indicates that the device is also a routeing device • Group (ACD) - indicates that the Group device has an ACD attribute (e.g. is an ACD Group) • Group (Hunt) - indicates that the Group device has a Hunt attribute • Group (Pick) - indicates that the Group device has a Pick attribute • deviceName (O) Characters (64) - Specifies the name associated with the entry in the list.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

13.1.5.2 Service Response

There are no service request completion conditions associated with this service.

13.1.5.2.1 Positive Acknowledgement

There is no positive acknowledgement associated with this service request.

13.1.5.2.2 Negative Acknowledgement

The negative acknowledgement error codes are described in 12.2.12, “ErrorValue”, on page 88.

13.1.5.3 Operational Model

13.1.5.3.1 Connection State Transitions

There are no connection state changes as the result of this service.

13.1.5.3.2 Device-Type Monitoring Event Sequences

There are no events generated as a result of this service.

13.1.5.3.3 Call-Type Monitoring Event Sequences

There are no events generated as a result of this service.

13.1.5.3.4 Functional Requirements

1. Due to the nature of the switching function configuration, the switching function may buffer the Switching Function Devices messages to the computing function. The time between these messages may vary significantly between various implementations.
2. The deviceList parameter may be provided without any devices in the list. This could occur when there are no devices that meet the conditions provided in the Get Switching Function Devices service request.
3. The Switching Function Devices service returns all device identifiers that refer to all devices in the application working domain (i.e. devices that can be controlled and/or observed).

14 System Services

This clause consists of:

- System Registration services
- System services

NOTE

This clause describes System Services between the Switching Function and the Computing Function.

14.1 Registration Services

Table 14-1 System Registration Services Summary

System Registration Service	Description	Pg.
14.1.1 Change System Status Filter	Changes the system status filter options for a current system registration.	152
14.1.2 System Register	Registers the computing function for system services with the switching function.	154
14.1.3 System Register Abort	Indicates that the switching function has terminated a system registration.	157
14.1.4 System Register Cancel	Unregisters the computing function for system services with the switching function.	158

14.1.1 Change System Status Filter

C $\longrightarrow$ S

The Change System Status Filter service is used by the computing function to change the filter options for a current system registration.

14.1.1.1 Service Request

Table 14-2 Change System Filter—Service Request

Parameter Name	Type	M/ O/C	Description
sysStatRegisterID	SysStatRegisterID	M	Specifies the system registration identifier for which the status filter should be changed.
requestedStatusFilter	Bitmap	M	Specifies the requested System Status Types to be filtered (not sent) by the switching function. This parameter is a bitmap of the following values: • Initializing • Enabled • Normal • Messages Lost • Disabled • Partially Disabled • Overload Imminent • Overload Reached • Overload Relieved Multiple bits may be set.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

14.1.1.2 Service Response

This service follows the atomic acknowledgement model for this service request.

14.1.1.2.1 Positive Acknowledgement

Table 14-3 Change System Status Filter—Positive Acknowledgement

Parameter Name	Type	M/ O/C	Description
actualStatusFilter	Bitmap	M	Specifies the actual set of System Status Types that will be filtered (not sent) by the switching function. This parameter is a bitmap with the same set of possible values as in the service request. The actualStatusFilter may differ from the requestedStatusFilter parameter in the service request (See Functional Requirement #2).
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

14.1.1.2.2 Negative Acknowledgement

The negative acknowledgement error codes are described in 12.2.12, “ErrorValue”, on page 88.

14.1.1.3 Operational Model

14.1.1.3.1 Connection State Transitions

There are no connection state changes due to this service.

14.1.1.3.2 Device-Type Monitoring Event Sequences

There are no events generated as a result of this service.

14.1.1.3.3 Call-Type Monitoring Event Sequences

There are no events generated as a result of this service.

14.1.1.3.4 Functional Requirements

1. The requestedStatusFilter parameter allows the computing function to choose the System Status Types for which no System Status service requests should be issued by the switching function. This parameter only applies to the System Status service. If the System Status service has not been requested for this system registration, then the switching function shall send a negative acknowledgement to the Change System Status Filter service request.

2. An implementation that does not support all System Status Types will nevertheless accept the Change System Status Filter service even if the requested filter cannot be provided. The service acknowledgement indicates the actual set of System Status Types that will be filtered. This means that the actual set of filtered types returned in the positive acknowledgement may include additional types to be filtered (or fewer types generated by the switching function) than those requested in the service request.

14.1.2 System Register

$C \longrightarrow S$

The System Register service is used by the computing function to register to receive system services from the switching function. The computing function may be required to register for system services before it can receive any system service requests from the switching function.

14.1.2.1 Service Request

Table 14-4 System Register—Service Request

Parameter Name	Type	M/ O/C	Description
requestTypes	Bitmap	M	Specifies the system services that are being registered. This parameter is a bitmap of the following values: • System Status • Request System Status Multiple bits may be set.
requestedStatusFilter	Bitmap	C	Specifies the requested set of System Status Types to be filtered (not sent) by the switching function. This parameter is a bitmap of the following values: • Initializing • Enabled • Normal • Messages Lost • Disabled • Partially Disabled • Overload Imminent • Overload Reached • Overload Relieved Multiple bits may be set. This parameter is mandatory if the requestTypes parameter includes System Status, otherwise it shall not be provided.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

14.1.2.2 Service Response

This service follows the atomic acknowledgement model for this service request.

14.1.2.2.1 Positive Acknowledgement

Table 14-5 System Register—Positive Acknowledgement

Parameter Name	Type	M/ O/C	Description
sysStatRegisterID	SysStatRegisterID	M	Specifies the system registration identifier for this registration.
actualStatusFilter	Bitmap	C	Specifies the actual set of System Status Types that will be filtered (not sent) by the switching function. This parameter is a bitmap with the same set of possible values as in the service request. The actual types filtered may differ from what was requested in the service request (See Functional Requirement #5). If the requestType parameter in the service request includes System Status, then this parameter is mandatory, otherwise it shall not be provided.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

14.1.2.2.2 Negative Acknowledgement

The negative acknowledgement error codes are described in 12.2.12, “ErrorValue”, on page 88.

14.1.2.3 Operational Model

14.1.2.3.1 Connection State Transitions

There are no connection state changes due to this service.

14.1.2.3.2 Device-Type Monitoring Event Sequences

There are no events generated as a result of this service.

14.1.2.3.3 Call-Type Monitoring Event Sequences

There are no events generated as a result of this service.

14.1.2.3.4 Functional Requirements

1. The sysStatRegisterID parameter returned in the positive acknowledgement is used to identify the registration over which system services will be sent. The sysStatRegisterID is also used when cancelling the system registration.

2. The number of simultaneous system registrations allowed is switching function dependent. When the limit is reached, subsequent System Register service requests shall result in negative acknowledgements from the switching function.

3. The requestTypes parameter specifies which system services are to be issued by the switching function to the registering computing function.

4. The requestedStatusFilter parameter allows the computing function to choose the System Status Types for which no System Status service requests should be issued by the switching function. If the System Status service is not being requested (i.e., in the requestTypes parameter), then the actualStatusFilter parameter does not apply and shall not be provided. The actual system status filter is provided in the actualStatusFilter parameter in the positive acknowledgement.

5. An implementation that does not support all System Status Types will nevertheless accept the System Register service even if the requestedStatusFilter cannot be provided. The service acknowledgement indicates the actual set of System Status Types that will be filtered. This means that the actual set of filtered types returned in the positive acknowledgement may include additional types to be filtered (or fewer types generated by the switching function) than what was requested in the service request.

6. If explicit registration is not supported, all system services (e.g., System Status, Request System Status) and System Status Types (e.g., Initializing, Enabled) supported by the switching function shall be provided to the computing functions. Note that the computing function *shall* be prepared to respond to a System Status service

request from the switching function in such cases (because it has no way of specifying that it should *not* receive such requests).

7. Note that if a computing function registers for system services it shall support the ability to respond to any switching function System Status service requests it may receive. In particular, for implicit registrations, a CSTA-conformant computing function shall always be able to support such requests from the switching function.

14.1.3 System Register Abort S $\longrightarrow$ C

The System Register Abort service is used by the switching function to asynchronously cancel an active system registration. This service invalidates a current systems status registration. There is no positive acknowledgement defined for this service.

14.1.3.1 Service Request

Table 14-6 System Register Abort—Service Request

Parameter Name	Type	M/ O/C	Description
sysStatRegisterID	SysStatRegisterID	M	Specifies the system registration identifier for the system registration that was aborted.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

14.1.3.2 Service Response

There are no service completion conditions for this service.

14.1.3.2.1 Positive Acknowledgement

There is no positive acknowledgement defined for this service.

14.1.3.2.2 Negative Acknowledgement

The negative acknowledgement error codes are described in 12.2.12, “ErrorValue”, on page 88.

14.1.3.3 Operational Model

14.1.3.3.1 Connection State Transitions

There are no connection state changes due to this service.

14.1.3.3.2 Device-Type Monitoring Event Sequences

There are no events generated as a result of this service.

14.1.3.3.3 Call-Type Monitoring Event Sequences

There are no events generated as a result of this service.

14.1.3.3.4 Functional Requirements

1. The switching function may issue this service at any time when it can no longer maintain the system registration.
2. The computing function may send a negative acknowledgement to this service request, but no positive acknowledgement is defined.

14.1.4 System Register Cancel

$C \longrightarrow S$

The System Register Cancel service is used to cancel a previous system registration. This request terminates the system registration and the computing function receives no further system service requests for that system registration once it receives the positive acknowledgement to the System Register Cancel request.

14.1.4.1 Service Request

Table 14-7 System Register Cancel—Service Request

Parameter Name	Type	M/ O/C	Description
sysStatRegisterID	SysStatRegisterID	M	Specifies the system registration identifier for which the system registration is to be cancelled.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

14.1.4.2 Service Response

This service follows the atomic acknowledgement model for this service request.

14.1.4.2.1 Positive Acknowledgement

Table 14-8 System Register Cancel—Positive Acknowledgement

Parameter Name	Type	M/ O/C	Description
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

14.1.4.2.2 Negative Acknowledgement

The negative acknowledgement error codes are described in 12.2.12, “ErrorValue”, on page 88.

14.1.4.3 Operational Model

14.1.4.3.1 Connection State Transitions

There are no connection state changes due to this service.

14.1.4.3.2 Device-Type Monitoring Event Sequences

There are no events generated as a result of this service.

14.1.4.3.3 Call-Type Monitoring Event Sequences

There are no events generated as a result of this service.

14.1.4.3.4 Functional Requirements

1. The computing function shall continue to process outstanding system requests from the switching function until it receives a positive acknowledgement for the System Register Cancel service request. The switching function shall not send any further system requests for a registration once it has sent the positive acknowledgement.

14.2 Services

Table 14-9 System Services Summary

System Service	Description	Pg.
14.2.1 Request System Status	Request to query the system status of the function receiving the request (bi-directional).	160
14.2.2 System Status	Request that reports the status of the function issuing the request to the function receiving the request (bi-directional). The indicated status may or may not have changed since the last System Status request was issued.	162
14.2.3 Switching Function Capabilities Changed	Request that reports that switching function level capability information (available via the Get Switching Function Capability service) has changed.	164
14.2.4 Switching Function Devices Changed	Request that reports that information associated with the current set of devices that can be controlled and observed in the switching sub-domain (available via the Get Switching Domain Devices service) has changed.	165

14.2.1 Request System Status

C ◄► S

The Request System Status service is used by the computing function or switching function to obtain (i.e., query) the system status of its peer function.

14.2.1.1 Service Request

Table 14-10 Request System Status—Service Request

Parameter Name	Type	M/ O/C	Description
sysStatRegisterID	SysStatRegisterID	C	Specifies the system registration identifier associated with the system registration for this request. This parameter is mandatory if the switching function is issuing the request and supports system registration, and shall not be provided otherwise.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

14.2.1.2 Service Response

This service follows the atomic acknowledgement model for this service request.

14.2.1.2.1 Positive Acknowledgement

Table 14-11 Request System Status—Positive Acknowledgement

Parameter Name	Type	M/ O/C	Description
systemStatus	Enumerated	M	Specifies the status of the function issuing the service request. The complete set of possible values is: • Initializing • Enabled • Normal • Messages Lost • Disabled • Partially Disabled • Overload Imminent • Overload Reached • Overload Relieved See 12.2.20, “SystemStatus”, on page 105 for a description of these values.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

14.2.1.2.2 Negative Acknowledgement

The negative acknowledgement error codes are described in 12.2.12, “ErrorValue”, on page 88.

14.2.1.3 Operational Model

14.2.1.3.1 Connection State Transitions

There are no connection state changes due to this service.

14.2.1.3.2 Device-Type Monitoring Event Sequences

There are no events generated as a result of this service.

14.2.1.3.3 Call-Type Monitoring Event Sequences

There are no events generated as a result of this service.

14.2.1.3.4 Functional Requirements

1. The systemStatus parameter in the positive acknowledgement provides the requesting function with information regarding the state of the overall system of the responding function. This information is important for proper system operation and should be processed accordingly. If the responding function has informed the requesting function that an overload condition is imminent then the requesting function should attempt to decrease the overall traffic to the responding function.

14.2.2 System Status

C ◄► S

The System Status service is used by the computing function or switching function to report its system status to its peer function. The indicated status may or may not have changed since the last System Status request was issued. This service can also be used to implement a heartbeat mechanism between the two functions.

14.2.2.1 Service Request

Table 14-12 System Status—Service Request

Parameter Name	Type	M/ O/C	Description
sysStatRegisterID	SysStatRegisterID	C	Specifies the system registration identifier associated with the system registration for this request. This parameter is mandatory if the switching function is issuing the request and supports system registration, and shall not be provided otherwise.
systemStatus	Enumerated	M	Specifies the status of the function issuing the service request. The complete set of possible values is: • Initializing • Enabled • Normal • Messages Lost • Disabled • Partially Disabled • Overload Imminent • Overload Reached • Overload Relieved See 12.2.20, “SystemStatus”, on page 105 for a description of these values.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

14.2.2.2 Service Response

This service follows the atomic acknowledgement model for this service request.

14.2.2.2.1 Positive Acknowledgement

Table 14-13 System Status—Positive Acknowledgement

Parameter Name	Type	M/ O/C	Description
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

14.2.2.2.2 Negative Acknowledgement

The negative acknowledgement error codes are described in 12.2.12, “ErrorValue”, on page 88.

14.2.2.3 Operational Model

14.2.2.3.1 Connection State Transitions

There are no connection state changes due to this service.

14.2.2.3.2 Device-Type Monitoring Event Sequences

There are no events generated as a result of this service.

14.2.2.3.3 Call-Type Monitoring Event Sequences

There are no events generated as a result of this service.

14.2.2.3.4 Functional Requirements

1. The systemStatus parameter in the request provides the responding function with information regarding the state of the overall system of the requesting function. This information is important for proper system operation and should be processed accordingly. If the requesting function has informed the other function that an overload condition is imminent then the responding function should attempt to decrease the overall traffic to the requesting function.

2. The computing function can determine if the switching function uses the System Status service for periodic status reporting (i.e., heartbeats) using the capabilities exchange services. The Get Switching Function Capabilities service positive acknowledgement defines a parameter (systemStatusTimer) that is used to indicate whether periodic status reporting is used and if so, how often the computing function should expect the reports. The recovery action to be taken by the computing function in the event of a loss of heartbeats is implementation specific.

14.2.3 Switching Function Capabilities Changed

S ⟶ C

The Switching Function Capabilities Changed service is used to indicate that switching function level capability information available via the Get Switching Function Capability service has changed.

The Get Switching Function Capability service may be used to obtain the revised information.

14.2.3.1 Service Request

Table 14-14 Switching Function Capabilities Changed—Service Request

Parameter Name	Type	M/ O/C	Description
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

14.2.3.2 Service Response

This service follows the atomic acknowledgement model for this service request.

14.2.3.2.1 Positive Acknowledgement

Table 14-15 Switching Function Capabilities Changed—Positive Acknowledgement

Parameter Name	Type	M/ O/C	Description
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

14.2.3.2.2 Negative Acknowledgement

The negative acknowledgement error codes are described in 12.2.12, “ErrorValue”, on page 88.

14.2.3.3 Operational Model

14.2.3.3.1 Connection State Transitions

There are no connection state changes due to this service.

14.2.3.3.2 Device-Type Monitoring Event Sequences

There are no events generated as a result of this service.

14.2.3.3.3 Call-Type Monitoring Event Sequences

There are no events generated as a result of this service.

14.2.3.3.4 Functional Requirements

1. If supported by the switching function, the Switching Function Capabilities Changed service shall be sent whenever switching function level capability information has changed, whether or not the Get Switching Function Capabilities service has been previously issued.

14.2.4 Switching Function Devices Changed

S ⟶ C

The Switching Function Devices Changed service is used to indicate that information associated with the current set of devices that can be controlled and observed in the switching sub-domain has changed.

The Get Switching Function Devices service may be used to obtain the revised information.

14.2.4.1 Service Request

Table 14-16 Switching Function Devices Changed—Service Request

Parameter Name	Type	M/ O/C	Description
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

14.2.4.2 Service Response

This service follows the atomic acknowledgement model for this service request.

14.2.4.2.1 Positive Acknowledgement

Table 14-17 Switching Function Devices Changed—Positive Acknowledgement

Parameter Name	Type	M/ O/C	Description
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

14.2.4.2.2 Negative Acknowledgement

The negative acknowledgement error codes are described in 12.2.12, “ErrorValue”, on page 88.

14.2.4.3 Operational Model

14.2.4.3.1 Connection State Transitions

There are no connection state changes due to this service.

14.2.4.3.2 Device-Type Monitoring Event Sequences

There are no events generated as a result of this service.

14.2.4.3.3 Call-Type Monitoring Event Sequences

There are no events generated as a result of this service.

14.2.4.3.4 Functional Requirements

1. If supported by the switching function, the Switching Function Devices Changed service shall be sent whenever switching function device information has changed, whether or not the Get Switching Function Devices service has been previously issued.

15 Monitoring Services

NOTE

This clause describes Monitoring Services between the Switching Function and the Computing Function.

15.1 Services

Table 15-1 Monitoring Services Summary

Monitoring Service	Description	Pg.
15.1.1 Change Monitor Filter	Modifies the event filter for an existing monitor.	167
15.1.2 Monitor Start	Initiates an event monitor on a specified device or call.	169
15.1.3 Monitor Stop	Terminates an existing monitor.	173

15.1.1 Change Monitor Filter

C ⟶ S

The Change Monitor Filter service is used to modify the set of event reports that are filtered out (not sent) over an existing monitor.

The new set of filtered out (not sent) event reports may be listed in the service acknowledgement.

15.1.1.1 Service Request

Table 15-2 Change Monitor Filter—Service Request

Parameter Name	Type	M/ O/C	Description
crossRefIdentifer	MonitorCrossRefID	M	This indicates the monitor for which to change the filter.
requestedFilterList	MonitorFilter	M	This parameter specifies the requested set of events to be filtered out (not sent) by the server. It is a bitmap of all events defined in this Standard.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

15.1.1.2 Service Response

This service follows the atomic acknowledgement model for this service request.

15.1.1.2.1 Positive Acknowledgement

Table 15-3 Change Monitor Filter—Positive Acknowledgement

Parameter Name	Type	M/ O/C	Description
actualFilterList	MonitorFilter	C	This parameter specifies the actual set of events that will be filtered out (not sent) by the server. It is a bitmap of all events defined in this Standard. The actual events filtered may differ from the requestedFilterList parameter on the service request (See Functional Requirement #1). This parameter is optional if the actualFilterList is the same as the requestedFilterList parameter on the service request, otherwise it is mandatory.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

15.1.1.2.2 Negative Acknowledgement

The negative acknowledgement error codes are described in 12.2.12, “ErrorValue”, on page 88.

15.1.1.3 Operational Model

15.1.1.3.1 Connection State Transitions

There are no connection state changes due to this service.

15.1.1.3.2 Monitoring Requirements

1. Once a request has been acknowledged, a new set of events will be filtered out (not sent) by the server.

15.1.1.3.3 Functional Requirements

1. An implementation that does not support all event reports, or that does not support filtering will nevertheless accept the Change Monitor Filter service even if the requested filter cannot be provided. In this case, the service acknowledgement indicates the actual set of events that will be filtered out (not sent). This means that the actual set of filtered events returned in the positive acknowledgement may include additional events to be filtered out (or fewer monitored events supported for the monitor) than those requested in the service request. For example, an implementation that does not support event filtering responds to the Change Monitor Filter service with a filter that shows provided events as unfiltered and unimplemented events as filtered out.

Similarly, an implementation that does not support, for example, Delivered events, shall always respond with a filter indicating that Delivered events will not be reported.

15.1.2 Monitor Start

C ⟶ S

The Monitor Start service initiates event reports (otherwise known as events) for a call, device, or for one or more calls involving a device.

The server starts a monitor, allocates a Monitor Cross Reference Identifier that uniquely identifies the monitor, and then positively acknowledges the request. All activities satisfying the filter provided (for example: call, feature, agent, private) trigger events which are delivered as a stream of event reports to the server. Each event contains the Monitor Cross Reference Identifier that correlates the event back to the Monitor Start service that established the monitor.

These event reports cease after the switching function terminates the monitor. Service termination can result from a client request (15.1.3, “Monitor Stop” on page -173) or it can be initiated by the server. The switching function shall terminate the monitor if the monitorObject ceases to exist, or if the monitorObject leaves the switching sub-domain. There may be other conditions that cause the server to terminate the monitor.

Once the monitor is terminated, the monitor cross reference ID is no longer valid.

Please refer to 6.7.2, “Monitoring”, on page 48 for an overview of monitoring and related concepts such as monitor objects, monitor types, monitor call types, and monitor filters.

15.1.2.1 Service Request

Table 15-4 Monitor Start—Service Request

Parameter Name	Type	M/ O/C	Description
monitorObject	Choice Structure	M	Specifies the monitor object of a call or device to be monitored. This shall be one of the following choices: • call (ConnectionID) - Specifies a call (connection) object. • device (DeviceID) - Specifies a device object. See Functional Requirement #5.
requestedMonitorFilter	MonitorFilter	O	This parameter specifies the requested set of events to be filtered out (not sent) by the switching function. It is a bitmap of all events defined in this standard. If this parameter is not provided (or if the parameter is not supported by the switching function), then it shall mean that no filtering of events is requested (all events are requested).
monitorType	Enumerated	O	Specifies the type of monitor requested. The complete set of possible values is: • call-type • device-type If this parameter is not provided (or if the parameter is not supported by the switching function), then the monitor type shall be selected by the switching function (as indicated by the capabilities exchange services).
requestedMonitorMediaClass	Bitmap	O	Specifies the media classes (voice, digital data, etc.) of calls that are being requested to be monitored for the monitorObject. Refer to the mediaClass component in 12.2.15, “MediaCallCharacteristics”, on page 101 for the complete set of possible values. Note that multiple bits may be set. If this parameter is not provided (or if the parameter is not supported by the switching function), it is switching function dependent which media classes of calls are monitored.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.

Table 15-4 Monitor Start—Service Request (continued)

Parameter Name	Type	M/ O/C	Description
privateData	CSTAPrivateData	O	Specifies non-standardized information.

15.1.2.2 Service Response

This service follows the atomic acknowledgement model for this service request.

15.1.2.2.1 Positive Acknowledgement

Table 15-5 Monitor Start—Positive Acknowledgement

Parameter Name	Type	M/ O/C	Description
crossRefIdentifer	MonitorCrossRefID	M	This indicates a value that is unique within the association for the duration of the monitor and that can be used to relate subsequent events to the monitor request that initiated them. It shall also allow correlating Monitor Stop and subsequent Change Monitor Filter services with the original Monitor Start service on which they act.
actualMonitorFilter	MonitorFilter	C	This parameter specifies the actual set of events that will be filtered (not sent) by the switching function. It is a bitmap of all events defined in this standard. The actual events filtered out may differ from the filterList parameter on the service request (See Functional Requirement #1). If this parameter is supported by the switching function, it may be omitted if the requested and actual monitor filters are the same, otherwise it shall be provided. If the parameter is not supported by the switching function, then the switching function does not filter events and all events supported (as indicated by the capability exchange services) shall be sent for this monitor.
actualMonitorMediaClass	Bitmap	C	This parameter specifies the actual media classes of calls that are monitored by the switching function for this monitor. The actual media classes of calls monitored may be the same or a subset of what was requested on the service request. If this parameter is supported by the switching function, it may be omitted if the requested and actual monitor media class parameters are the same otherwise it shall be provided. If the parameter is not supported by the switching function, then the switching function does not filter call types for specific monitors. The capability exchange services indicates the media classes of calls that can be monitored.

Table 15-5 Monitor Start—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
monitorExistingCalls	Boolean	O	Indicates whether or not the computing function will receive event reports regarding calls that are currently existing at the device at which the monitor was started. The complete set of possible values is: • TRUE - Indicates event reports will be provided for calls that are at the device at the time of the acknowledgement [Default]. • FALSE - Indicates event reports will not be provided for calls that are at the device at the time of the acknowledgement. This parameter is applicable to monitors that have devices as their object. For such monitors, if this parameter is not present (or the parameter is not supported), it means that the switching function always provides event reports for calls that are currently present at the device when the monitor was started.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

15.1.2.2.2 Negative Acknowledgement

The negative acknowledgement error codes are described in 12.2.12, “ErrorValue”, on page 88.

15.1.2.3 Operational Model

15.1.2.3.1 Connection State Transitions

There are no connection state changes due to this service.

15.1.2.3.2 Monitoring Requirements

1. For call related events, events are provided for all devices associated with the call or device being monitored. For non-call related events, events are provided for the monitored device.
2. Once a call is monitored (irrespective of the monitor type or monitor object), all connection state changes that are known by the switching function are reported (subject to the Monitor Filter).
   - For example, if device A is being monitored (with a device-type monitor) and a call is placed to device B (no monitor on B), then any connection state changes for either device A or B (such as when B receives the Delivered event and answers the call) will be reported through device A’s monitor.
   - Monitoring is only guaranteed for devices in the switching sub-domain. Activity related to devices outside the switching sub-domain may be only partially available or completely unreported.
3. Since some devices do not support all events, when a device enters a call that is being monitored with a call-type monitor, there may be a reduced set of event reporting associated with that monitor.
4. Physical Device, Logical Device, and Maintenance events are only reported for device-type monitors.

15.1.2.3.3 Functional Requirements

1. An implementation that does not support all event reports, or that does not support filtering will nevertheless accept the Monitor Start service even if the requested filter cannot be provided. In this case (if the actualMonitorFilter parameter is supported), the service acknowledgement indicates the actual set of events that will be filtered out. This means that the actual set of filtered events returned in the positive acknowledgement may include additional events to be filtered out (or less monitored events supported for the monitor) than those requested in the service request. For example, an implementation that does not support event filtering responds to the Monitor Start service with a filter that shows provided events as unfiltered and unimplemented events as filtered out. Similarly, an implementation that does not support, for example, Delivered events, shall respond with a filter indicating that Delivered events will not be reported.

2. If the switching function does not support the requestedMonitorFilter parameter (and the switching function is ignoring unsupported parameters, as defined by the capability exchange services), the computing function may receive events that it had requested to be filtered out.
3. When monitoring device configurations, refer to 6.1.7, “Referencing Devices, Elements, Appearances and Device Configurations”.
4. Events that occurred prior to the Monitor Start positive acknowledgement are not reported.
5. When a call-type monitor is requested and a connection identifier is being provided, the connection identifier may consist of only the callID. This is an exception to the general rule that deviceIDs are always provided in connectionIDs (refer to section 12.3.9, “ConnectionID”).
6. The Service Completion Failure event is *only* reported to device-type monitors on the device which has or had connection(s) that were used in the particular service request which is associated with this event (i.e., this event is not reported to any call-type monitors or device-type monitors for other devices in the call(s) associated with the service request).
7. If a computing function starts a monitor on a device, then issues another Monitor Start on the same device, the switching function will start a new monitor and will create a new Monitor Cross Reference Identifier. The new Monitor Start service request can be the same as the original or it may include a different Monitor Type or a different Monitor Filter.
8. The event reporting of the Failed event may vary depending on the given switching function. For the conditions under which the event reporting is limited, see section 6.8, “Additional Services, Features & Behaviour” and the Failed event section.
9. If filtering of the individual Private Events is desired, then the CSTA Private Data Information (privateData parameter) shall be used.

15.1.3 Monitor Stop

C ◄► S

The Monitor Stop service is used to cancel a previously initiated Monitor Start service.

The Monitor Stop service can be issued by a function to terminate or signal the termination of a corresponding Monitor Start service.

A positive acknowledgement to the service request indicates that the Cross Reference ID used by the Monitor Start service has become invalid.

15.1.3.1 Service Request

Table 15-6 Monitor Stop—Service Request

Parameter Name	Type	M/ O/C	Description
crossRefIdentifer	MonitorCrossRefID	M	This indicates the Cross Reference Identifier provided in the original Monitor Start service positive acknowledgement.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

15.1.3.2 Service Response

This service follows the atomic acknowledgement model for this service request.

15.1.3.2.1 Positive Acknowledgement

Table 15-7 Monitor Stop—Positive Acknowledgement

Parameter Name	Type	M/ O/C	Description
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

15.1.3.2.2 Negative Acknowledgement

The negative acknowledgement error codes are described in 12.2.12, “ErrorValue”, on page 88.

15.1.3.3 Operational Model

15.1.3.3.1 Connection State Transitions

There are no connection state changes due to this service.

15.1.3.3.2 Monitoring Requirements

1. Once a request has been acknowledged, event reports are no longer sent.

15.1.3.3.3 Functional Requirements

1. This service can be sent to the function that requested a monitor to report that the monitor has been terminated.
2. The switching function may issue a Monitor Stop service when it can no longer provide information. Examples of when this may occur are:
   - For monitors on calls that have ended (call monitoring).
   - For load management reasons.
   - If the monitor object leaves the switching sub-domain (via configuration, for example).

16 Snapshot Services

NOTE

This clause describes Snapshot Services between the Switching Function and the Computing Function.

16.1 Services

Table 16-1 Snapshot Services Summary

Snapshot Service	Description	Pg.
16.1.1 Snapshot Call	Provides information about the devices participating in a specified call.	175
16.1.2 Snapshot Device	Provides information on the status of calls at a specific device.	178
16.1.3 Snapshot CallData	Provides Snapshot Call Information in segmented messages.	181
16.1.4 Snapshot DeviceData	Provides Snapshot Device Information in segmented messages.	183

16.1.1 Snapshot Call

C $\longrightarrow$ S

The Snapshot Call service provides information about the devices participating in a specified call. The information provided includes device identifiers, their connections in the call, and local connection states of the devices in the call as well as call related information.

Information that applies to the entire call shall be provided in the Snapshot Call positive response.

Information that is specific to each endpoint in the call (snapshotData parameter) shall be provided using one of two possible mechanisms (as indicated by the capability exchange services): either in the Snapshot Call positive acknowledgement or in one or more messages using the Snapshot CallData Service. Note that both mechanisms cannot be used at the same time.

If the switching function supports the Dynamic Feature Availability option (as indicated through the capabilities exchange services), then for each connection in the call, this service provides the list of permitted call control services.

16.1.1.1 Service Request

Table 16-2 Snapshot Call—Service Request

Parameter Name	Type	M/ O/C	Description
snapshotObject	ConnectionID	M	This indicates the connectionID of the call to be snapshot. See Functional Requirement #2.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

16.1.1.2 Service Response

This service follows the atomic acknowledgement model for this service request.

16.1.1.2.1 Positive Acknowledgement

Table 16-3 Snapshot Call—Positive Acknowledgement

Parameter Name	Type	M/ O/C	Description
serviceCrossRefID	ServiceCrossRefID	C	Specifies the reference used to associate subsequent Snapshot CallData services to this service request. This parameter is mandatory if the switching function is providing the snapshot information using the Snapshot CallData service, otherwise it shall not be provided.

Table 16-3 Snapshot Call—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
snapshotData	List of Structures	C	Specifies information for each endpoint in a call. This parameter is mandatory if the switching function is providing all of the response information in this message. This parameter shall not be provided if the switching function is providing the snapshot information using the Snapshot CallData Service. The complete set of possible information is: • deviceOnCall (M) DeviceID - Of a device involved with the endpoint. • connectionIdentifer (C) ConnectionID - For the endpoint. This is mandatory if the endpoint is in the switching sub-domain, otherwise it is optional. • localConnectionState (O) LocalConnectionState - For the endpoint. • servicesPermitted (C) ServicesPermitted - This is mandatory if the switching function supports the Dynamic Feature Availability option (as indicated through the capabilities exchange service). • mediaServiceInformationList (O) List of Structure - Specifies information about the media services that are attached (bound to) the connection in the call. The complete set of possible information is: • mediaServiceType (M) MediaServiceType - A media service type that has been bound to the connection. • mediaServiceVersion (O) Value. The version of the media services. • mediaServiceInstance (O) MediaServiceInstanceID - A media service instance associated with the media service bound to the connection. • mediaStreamID (C) MediaStreamID - The media stream identifier for the media service binding. This shall be provided if the switching function supports providing the mediaStreamID as indicated by the capability exchange services. • connectionInformation (O) ConnectionInformation - The connection information associated with the callIdentifier connection.
mediaCallCharacteristics	MediaCallCharacteristics	O	This specifies the media class and data characteristics of the call. See 12.2.15, “MediaCallCharacteristics”, on page 101 for the list of possible values.
callCharacteristics	CallCharacteristics	O	Specifies the high level characteristics (ACD call, Priority call, etc.) associated with the call. See 12.2.4, “CallCharacteristics”, on page 84 for the complete set of possible values.
callingDevice	CallingDeviceID	O	Specifies the calling device.
calledDevice	CalledDeviceID	O	Specifies the called device.
associatedCallingDevice	AssociatedCallingDeviceID	C	Specifies the Network Interface Device associated with the calling device if the call is an external incoming call. This parameter is mandatory for all external incoming calls and shall not be provided otherwise.

Table 16-3 Snapshot Call—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
associatedCalledDevice	AssociatedCalledDeviceID	C	For outgoing external calls, this parameter specifies the Network Interface Device associated with the originally called device. For incoming external calls, this parameter specifies a device within the switching sub-domain associated with the originally called device. This parameter is mandatory for all external outgoing calls and it is optional for external incoming calls.
correlatorData	CorrelatorData	C	Specifies the correlator data associated with the call. This parameter is mandatory if there is correlator data associated with the call.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

16.1.1.2.2 Negative Acknowledgement

The negative acknowledgement error codes are described in 12.2.12, “ErrorValue”, on page 88.

16.1.1.2.3 Operational Model

16.1.1.2.4 Connection State Transitions

There are no connection state changes due to this service.

16.1.1.2.5 Monitoring Requirements

This service does not affect existing monitors.

16.1.1.2.6 Functional Requirements

1. The Snapshot Call service is intended to provide information about devices in calls that makes further monitoring more meaningful. For example, if the computing function started working with a call, the event reports needed to provide synchronization may not occur for some time. To facilitate operations before an event report is available to synchronize the monitor, it is necessary to be able to query the current state of devices. Snapshot Call service provides this function.

2. The connection ID provided in the service may consist of only a callID portion. This is an exception to the rule of always providing deviceIDs in connectionIDs of service requests as described in 12.3.9, “ConnectionID”, on page 110.

3. If the switching function is providing the snapshot response information in multiple messages, it shall support the Snapshot CallData service. In this case, the computing function can associate subsequent Snapshot CallData services to the original Snapshot Call service by means of the serviceCrossRefID parameter.

16.1.2 Snapshot Device S ⟶ C

The Snapshot Device service provides information about calls associated with a given device. The information provided identifies each call the device is participating in and the local connection state of the device in that call.

The switching function shall provide the response information using one of two possible mechanisms (as indicated by the capability exchange services): either in the Snapshot Device positive acknowledgement or in one or more messages using the Snapshot DeviceData Service. Note that both mechanisms cannot be used at the same time.

If the switching function supports the Dynamic Feature Availability option (as indicated through the capabilities exchange services), then for each connection at the device, this service provides the list of permitted call control services.

16.1.2.1 Service Request

Table 16-4 Snapshot Device—Service Request

Parameter Name	Type	M/ O/C	Description
snapshotObject	DeviceID	M	This indicates the deviceID of the device to be snapshot.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

16.1.2.2 Service Response

This service follows the atomic acknowledgement model for this service request.

16.1.2.2.1 Positive Acknowledgement

Table 16-5 Snapshot Device—Positive Acknowledgement

Parameter Name	Type	M/ O/C	Description
serviceCrossRefID	ServiceCrossRefID	C	Specifies the reference used to associate subsequent Snapshot DeviceData services to this service request. This parameter is mandatory if the switching function is providing the snapshot device information using the Snapshot DeviceData service, otherwise it shall not be provided.

Table 16-5 Snapshot Device—Positive Acknowledgement (continued)

Parameter Name	Type	M/ O/C	Description
snapshotData	List of Structures	C	Specifies information for each call at a device. This parameter is mandatory if the switching function is providing all of the response information in this message. This parameter shall not be provided if the switching function is providing the response information using the Snapshot DeviceData Service. This complete set of information is: • connectionIdentifier (M) ConnectionID • localCallState (M) Choice Structure - This shall be one of the following choices: • compoundCallState (List of LocalConnectionStates) - This consists of a sequence of local connection states. • simpleCallState (SimpleCallState) - the simple call state. • unknown • servicesPermitted (C) ServicesPermitted - This is mandatory if the switching function supports the Dynamic Feature Availability option (as indicated through the capabilities exchange services). • mediaServiceInformationList (O) List of Structures - Specifies information about the media services that are attached (bound to) the connection in the call. The complete set of possible components is: • mediaStreamID (C) MediaStreamID - The media stream identifier for the media service binding. This shall be provided if the switching function supports providing the mediaStreamID as indicated by the capability exchange services. • connectionInformation (O) ConnectionInformation - The connection information associated with the callIdentifier connection. • mediaCallCharacteristics (O) MediaCallCharacteristics - specifies the media class and data characteristics of the call. See 12.2.15, “MediaCallCharacteristics”, on page 101 for the list of possible values.
security	CSTASecurityData	O	Specifies the timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

16.1.2.2.2 Negative Acknowledgement

The negative acknowledgement error codes are described in 12.2.12, “ErrorValue”, on page 88.

16.1.2.3 Operational Model

16.1.2.3.1 Connection State Transitions

There are no connection state changes due to this service.

16.1.2.3.2 Monitoring Requirements

This service does not affect existing monitors.

16.1.2.3.3 Functional Requirements

1. The Snapshot Device service is intended to provide information about devices to make further monitoring more meaningful. For example, when a computing function starts working with a device, the event reports needed to provide synchronization may not occur for some time. To facilitate operations before event reports synchronize the monitor, it is necessary to be able to query the current state of devices. Snapshot Device service provides that function.

2. If the switching function is providing the snapshot device response information in multiple messages, it shall support the Snapshot DeviceData service. In this case, the computing function can associate subsequent Snapshot DeviceData services to the original Snapshot Device service by means of the serviceCrossRefID parameter.

16.1.3 Snapshot CallData

S $\longrightarrow$ C

This service is generated as a result of the Snapshot Call service. It is used when the switching function is providing snapshot call information in multiple messages (otherwise the switching function provides the snapshot call information in the Snapshot Call positive acknowledgement).

The information provided includes information about the endpoints in the call (information about the entire call is provided in the Snapshot Call positive response).

The switching function may generate a sequence of Snapshot CallData services, individually referred to as segments, in response to a single Snapshot Call service request.

16.1.3.1 Service Request

Table 16-6 Snapshot CallData—Service Request

Parameter Name	Type	M/ O/C	Description
serviceCrossRefID	ServiceCrossRefID	M	Specifies the reference used to associate the Snapshot CallData service messages to the Snapshot Call service request.
segmentID	Value	O	Specifies the segment number of this message. Each successive segment number in the sequence increments the segmentID by one.
lastSegment	Boolean	M	Specifies if this segment is the last one associated with the serviceCrossRefID. The complete set of possible values is: • TRUE - Indicates that this is the last segment • FALSE - Indicates that this is not the last segment in the sequence.
snapshotData	List of Structures	M	Specifies information for each endpoint in a call. The complete set of possible information is: • deviceOnCall (M) DeviceID - Of a device involved with the endpoint. • connectionIdentifer (C) ConnectionID - For the endpoint. This is mandatory if the endpoint is in the switching sub-domain, otherwise it is optional. • localConnectionState (O) LocalConnectionState - For the endpoint. • servicesPermitted (C) ServicesPermitted - This is mandatory if the switching function supports the Dynamic Feature Availability option (as indicated through the capabilities exchange service). • mediaServiceInformationList (O) List of Structures - Specifies information about the media services that are attached (bound to) the connection in the call. The complete set of components is: • mediaServiceType (M) MediaServiceType - A media service type that has been bound to the connection. • mediaServiceVersion (O) Value. The version of the media services. • mediaServiceInstance (O) MediaServiceInstanceID - A media service instance associated with the media service bound to the connection. • mediaStreamID (C) MediaStreamID - The media stream identifier for the media service binding. This shall be provided if the switching function supports providing the mediaStreamID as indicated by the capability exchange services. • connectionInformation (O) ConnectionInformation - The connection information associated with the callIdentifier connection.

Table 16-6 Snapshot CallData—Service Request

Parameter Name	Type	M/ O/C	Description
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.
privateData	CSTAPrivateData	O	Specifies non-standardized information.

16.1.3.2 Service Response

There are no service request completion conditions associated with this service.

16.1.3.2.1 Positive Acknowledgement

There is no positive acknowledgement associated with this service request.

16.1.3.2.2 Negative Acknowledgement

The negative acknowledgement error codes are described in 12.2.12, “ErrorValue”, on page 88.

16.1.3.3 Operational Model

16.1.3.3.1 Connection State Transitions

There are no connection state changes due to this service.

16.1.3.3.2 Device-Type Monitoring Event Sequences

There are no events generated as a result of this service.

16.1.3.3.3 Call-Type Monitoring Event Sequences

There are no events generated as a result of this service.

16.1.3.3.4 Functional Requirements

1. Due to the nature of the switching function configuration, the switching function may buffer the Snapshot CallData messages to the computing function. The time between these messages may vary significantly between various implementations.
2. The switching function may include information for one or more endpoints in each segment.
3. The information reported in the sequence of segments generated from the Snapshot Call service request represents the state of the call at the time the Snapshot Call service is positively acknowledged.

16.1.4 Snapshot DeviceData

S $\longrightarrow$ C

This service is generated as a result of the Snapshot Device service. It is used when the switching function is providing snapshot device response information in multiple messages (otherwise the switching function provides the snapshot device response in the Snapshot Device positive acknowledgement).

This includes information about calls associated with a given device. The information provided identifies each call the device is participating in and the local connection state of the device in that call.

The switching function may generate a sequence of Snapshot DeviceData services, individually referred to as segments, in response to a single Snapshot Device service request.

16.1.4.1 Service Request

Table 16-7 Snapshot DeviceData—Service Request

Parameter Name	Type	M/ O/C	Description
serviceCrossRefID	ServiceCrossRefID	M	Specifies the reference used to associate the Snapshot DeviceData service messages to the Snapshot Device service request.
segmentID	Value	O	Specifies the segment number of this message. Each successive segment number in the sequence increments the segmentID by one.
lastSegment	Boolean	M	Specifies if this segment is the last one associated with the serviceCrossRefID. The complete set of possible values is: • TRUE - Indicates that this is the last segment • FALSE - Indicates that this is not the last segment in the sequence.
snapshotData	List of Structures	M	Specifies information for each call at a device. This complete set of information is: • connectionIdentifier (M) ConnectionID • localCallState (M) Choice Structure - This shall be one of the following choices: • compoundCallState (List of LocalConnectionStates) - This consists of a sequence of local connection states. • simpleCallState (SimpleCallState) - The simple call state. • unknown • servicesPermitted (C) ServicesPermitted - This is mandatory if the switching function supports the Dynamic Feature Availability option (as indicated through the capabilities exchange services). • mediaServiceInformationList (O) List of Structures - Specifies information about the media services that are attached (bound to) the connection in the call. The complete set of possible components include: • mediaStreamID (C) MediaStreamID - The media stream identifier for the media service binding. This shall be provided if the switching function supports providing the mediaStreamID as indicated by the capability exchange services. • connectionInformation (O) ConnectionInformation - The connection information associated with the callIdentifier connection • MediaCallCharacteristics (O) MediaCallCharacteristics - specifies the media class and data characteristics of the call.
security	CSTASecurityData	O	Specifies timestamp information, message sequence number, and security information.

Table 16-7 Snapshot DeviceData—Service Request

Parameter Name	Type	M/ O/C	Description
privateData	CSTAPrivateData	O	Specifies non-standardized information.

16.1.4.2 Service Response

There are no service request completion conditions associated with this service.

16.1.4.2.1 Positive Acknowledgement

There is no positive acknowledgement associated with this service request.

16.1.4.2.2 Negative Acknowledgement

The negative acknowledgement error codes are described in 12.2.12, “ErrorValue”, on page 88.

16.1.4.3 Operational Model

16.1.4.3.1 Connection State Transitions

There are no connection state changes due to this service.

16.1.4.3.2 Device-Type Monitoring Event Sequences

There are no events generated as a result of this service.

16.1.4.3.3 Call-Type Monitoring Event Sequences

There are no events generated as a result of this service.

16.1.4.3.4 Functional Requirements

1. Due to the nature of the switching function configuration, the switching function may buffer the Snapshot DeviceData messages to the computing function. The time between these messages may vary significantly between various implementations.
2. The switching function may include information for one or more connections in each segment.
3. The information reported in the sequence of segments generated from the Snapshot Device service request represents the state of the device at the time the Snapshot Device service is positively acknowledged.