



**CSTA FAQ**

---

# **Frequently Asked Questions (updated 21 Dec. 1998)**

The answers to the following questions do not represent an official ECMA position, but are provided to be helpful. Further questions can be directed to the ECMA Secretariat ([helpdesk@ecma.ch](mailto:helpdesk@ecma.ch)) for further disposition.

---

## **Questions**

**Is CSTA Y2K (millennium "bomb") safe?**

**Why** is the heading "( iso(1) ....., icd-ecma(0012).... make-call(10) )" defined in the protocol (example for Make-call service(ecma-218)?

**How** does the CSTA application relate to the Computing and Switching functions, and also to a telephony server?

**Does** CSTA assume that a telephony driver exists or do we have to make some changes in our switch to support CSTA?

**To be** CSTA compliant do we have to support all the switching functionality defined for CSTA?

**What** is the "CSTA Link" between the switch and a server?

**Why** does the MakeCall service in ECMA 217 show the caller connected to a call before the service is used?

**How** to clear an incoming call before it rings

---

## **Why is the heading "( iso(1) ....., icd-ecma(0012).... make-call(10) )" defined in the protocol (example for Make-call service(ecma-218)?**

What you are looking at is an ASN.1 definition of the protocol data unit (PDU) for the Make Call service. ASN.1 is "Abstract Syntax Notation One", a data structure representation methodology developed by the International Telecommunications Union (ITU). It consists of a declarative language (the header is written in ASN.1, for example), and an agreed-upon set of encoding rules for translating the ASN.1 source code into a byte stream. The syntax and encoding rules allow protocol developers to guarantee that they are encoding and decoding the byte streams the same way.

Standards organisations such as ECMA, ITU, ISO develop protocols that have very complex PDUs, and in general multiple protocols may coexist on the same transport, so ensuring that the byte streams can be recognised correctly is very important.

To use the definition of MakeCall in ECMA-218, for example, you must write code that encodes a MakeCall command into a byte stream according to the ASN.1 definition (this code runs in your application stack), and code that decodes the MakeCall byte stream (this code resides in your switch) into an internal data structure that your switch can interpret and execute. For "simple" pieces of ASN.1, this can be done by manual compilation, much like translating a fragment of C into object code. But for complex ASN.1 definitions, this can become very difficult. For example, the fragment:

```
( iso(1) ....., icd-ecma(0012).... make-call(10) )
```

is an object identifier, a byte stream that encodes the Make Call command uniquely among all possible commands from all possible sources from all possible standards (*whew!*). The ISO (i.e., iso(1)), ECMA (i.e., icd-ecma(0012), etc.), tags uniquely identify this command as being defined by ECMA and nobody else. To compile this manually, you need to find the ASN.1 definitions that define the header, and apply the so-called Basic Encoding Rules (BER) to translate it. Among the definitions you will need are ROSE (Remote Operation Service Element) and ACSE (Association Control Service Element), which are defined in other standards documents (see the Bibliography).

There are some developers who either know ASN.1 very well, or are only using a very few PDUs, who write encoding and decoding functions manually. Most people use ASN.1 compilers, which take ECMA-218, ROSE, ACSE, etc. definitions as input, and generate encoding and decoding functions

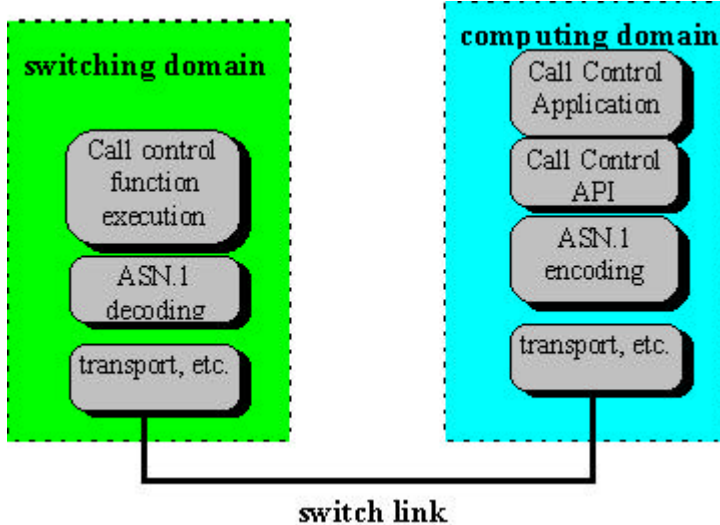
(in your favourite programming language) as output.

ASN.1 was defined in the early 1980's (documents X.208, X.209), and updated in 1994 to include such features as Unicode encoding (X.680, X.690). CSTA Phase II and CSTA Phase III assume the 1994 definition.

The [bibliography](#) includes citations of the official specifications of ASN.1, and a tutorial explaining ASN.1 and the Basic Encoding Rules.

**How does the CSTA application relate to the Computing and Switching functions, and also to a telephony server?**

This picture, which depicts one example, may make the situation clearer:



The bottom of each stack, whether it is in the switching domain or computing domain, is a transport layer. The top of the computing domain is an application. By some means, typically an API, the application generates CSTA PDUs encoded as defined by ECMA-218, to invoke the services of ECMA-217. In the switching domain, there is a call control function that performs the services of ECMA-217, which it receives as ECMA-218-encoded PDUs.

Currently, CSTA doesn't specify these stacks, other than ASN.1 encoding/decoding (that is, the application and presentation layers) and the semantics of "Call Control Function".

## **Does CSTA assume that a telephony driver exists or do we have to make some changes in our switch to support CSTA?**

Unless your switch is CSTA ready, you will have to probably do one or both of these. Your switch will have to be able to have an ASN.1 encoding/decoding layer for CSTA, and have a transport layer that will communicate to the client over the "switch link". On the client side, you will need to provide similar functions. This may be done via a driver, but that is an implementation detail.

## **To be CSTA compliant do we have to support all the switching functionality defined for CSTA?**

No. There is a PICS Performa (a sort of template) at the end of the protocol in ECMA 218. A switch manufacturer may provide a PICS statement describing what functions they provide.

## **What is the "CSTA Link" between the switch and a server?**

The above [figure](#) should cover this - it is a transport layer between a client and the server. It could be a serial cable, a LAN, the Internet, whatever you provide in your switch.

## **Why does the MakeCall service in ECMA 217 show the caller connected to a call before the service is used?**

At first sight it may not seem logical to have the call connected before using the MakeCall service,

which you would expect to create the call. However, the condition of having the call already connected is an option, the connection may exist or it may be Null (because the call doesn't yet exist). We need to allow for the call to exist to allow for cases where MakeCall is allowed whilst the phone is off-hook and providing dial tone, e.g. for manual number prefixing

## **How to clear an incoming call before it rings**

A developer writes:

I have a question concerning the timing between the switching domain and the application domain. The following example might help to illustrate the question:

Assume that there is an incoming call to a local device of the switching sub-domain and that the device is monitored by an application. The switching function will generate a "DeliveredEvent" according to the CSTA standard and will deliver this event report to the application domain. Now, assume that the application wants - for whatever reasons - to prevent the called device from being alerted. The problem is that the application only discovers that the device is about to get a call when it receives the event and hence cannot issue a "ClearCall" request until this occurs, meanwhile the switching domain could already have started alerting the device.

Is there any way to allow the application to get in and stop the call before the device is alerted?

## **Answer**

In phase I and phase II of CSTA you will have to use the Routing Services to intercept the call and re-route it to an alternative device where it may be cleared after it has started to alert the alternative device.

However, there is a new call control event, called Offered, in ECMA-269 (CSTA Phase III) that may provide a better solution to your problem. If supported on the device by the switching function, the Offered event is generated when the call arrives at the device but before the device is alerted. Typically the device stays in this "offered" state until it receives instructions from an application on what to do with the call (CSTA service requests like Divert Call, Accept Call, or Clear Connection, for example) or, in the absence of any such instructions, after a certain (switch determined) time period the device would typically be alerted.

We think that this may provide the switch/application synchronisation that you are looking for as it will allow the application an opportunity to clear the call before the device is alerted.

For more information refer to the Offered event and the Accept Call service in [ECMA-269](#).

## **Is CSTA Y2K safe?**

For ECMA-180, Phase I, and ECMA-218, Phase II, (which use UTCTime defined by ITU-T) it has been decided by ECMA TC32-TG11 that if the year value is reported as:

- 00-49 then this means that the century is 20 (i.e. the year is 2000 to 2049), whilst
- 50-99 means that the century is 19 (i.e. the year is 1950 to 1999)

A usage note has been added into ECMA-180 and ECMA-218 to report the chosen solution to this problem.

This gives a 50 year period at the start of the new millennium during which implementations based upon phase I and phase II of CSTA will be unaffected by the use of UTCTime in the CSTA protocol. TG11 feels that should be ample time to upgrade to the latest CSTA.

CSTA Phase III, ECMA-285, uses GeneralizedTime which is not susceptible to the Y2K problem.

---



## Bibliography

ISO 6523 Data interchange -- Structures for the identification of organizations (1984)

ISO/IS 7498 International Standard: Information Processing Systems - Open Systems Interconnection - Basic Reference Model

ISO/IS 8649 International Standard: Information Processing Systems - Open Systems Interconnection - Association Control Service Element

IS/ISO 8824 International Standard: Information Processing Systems - Open Systems Interconnection - Specification Of Abstract Syntax Notation One (ASN.1) (This corresponds to CCITT X.208, which was updated in 1994 to X.680)

IS 8825 International Standard: Information Processing Systems - Open Systems Interconnection - Specification Of Basic Encoding Rules For ASN.1 (This corresponds to CCITT X.209, which was updated in 1994 to X.690)

DIS 9072-1 Draft International Standard: Information Processing Systems - Text Processing - Remote Operations Part 1: Model, Notation And Service Definition

DIS 9072-2 Draft International Standard: Information Processing Systems - Text Processing - Remote Operations Part 2: Protocol Specification

---

[ASN1-A] "Introduction to ASN.1 and the Packed Encoding Rules,"

[ASN1-B] "Abstract Syntax Notation One",

<http://renoir.vill.edu/faculty/schragge/html/CSC8560/asn1.html>

SNACC "Index of /pub/languages/asn/snacc/",

Updated 20 March 1997