行政院及所屬各機關出國報告
（出國類別：考察）

參加『Natural MicroSystem 亞洲工程師研討會議』心得報告

服務機關：中山科學研究院
出國人職稱：荐聘技正　　委聘技佐
姓　　名：陳正浩　　劉光漢

出國地區：印尼巴里島
出國期間：89 年 10 月 28 日至 89 年 11 月 4 日
報告日期：90 年 1 月 15 日

隨者科技的日欣月異，訊息的傳遞由電報、電話、傳真到網路，網路的發展也由最初單純的 data、圖形的傳遞，發展到現在能夠傳遞即時語音、影像。且將電信網路與 internet 整合成多媒體傳遞的網路已是未來之趨勢。本組參與之『行動電話語音監察系統』計劃，其計劃之目的為對犯罪之監察，隨著信息傳遞之整合發展，未來建案之方向必定朝向網路及電信整合網路之監察，本次參與 Natural MicroSystem 亞洲工程師會議之目的即為研討並蒐集最新通訊相關之技術資料，以備未來建案及執行之需，Natural MicroSystem 為國際通訊介面卡設備領導大廠，在通訊的各領域或有產品發表，或與其他領導廠商合作，隨時掌握最先進之通訊資訊；借由參加 Natural MicroSystem 亞洲工程師會議的機會，收集 Natural MicroSystem 及其合作廠商有關未來通訊發展趨勢及相關產品之資訊，實已達到達成參與研討會之目地。

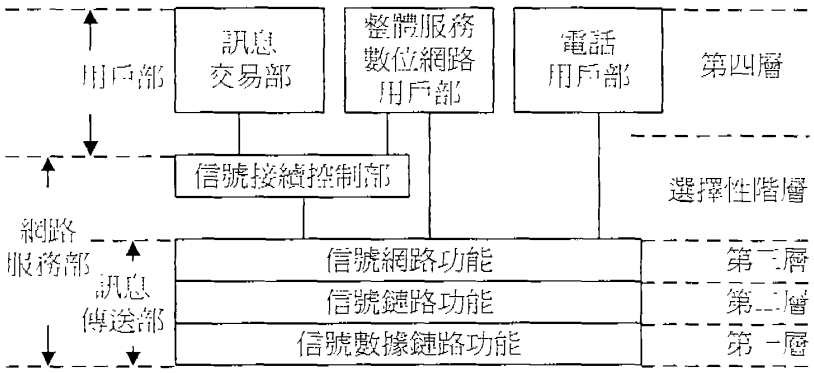
一、前言

未來電信網路之發展趨勢應為寬頻網路，就是由原來單純之data、圖片之非即時傳送，聲音的窄頻即時傳送，推展到需寬頻的影音即時傳送及其它先進之加值服務。在既有的通信網路(如ATM(Asynchronous Transfer Mode)、SS7(Signaling System No.7))及網際網路（如 IP(Internet Protocol)）優良技術基礎上，整合電話網路及行動通信網路、公眾網路及企業網路、Data Service 及 Voice Service 成為具備 Multi-service 功能之單一型式(Unified)網路。由於未來對電信網路之需求，需具備提供各種頻寬及不同服務等級需求之語音、數據、及多媒體訊務高效率載送能力，導致於下一代電信網路中以分封電話(Packet Telephony)技術提供電信級之VOIP(Voice Over IP)及 VTOA(Voice and Telephony Over ATM)服務之發展，此一發展方向將成為未來電信網路技術發展之主要研究議題。

二、說明

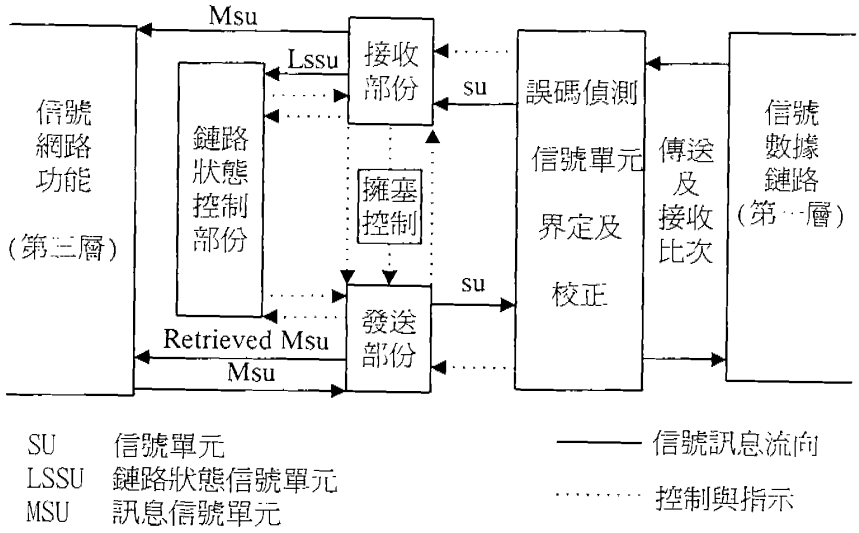
1 SS7 信號：

SS7 信號也可稱為 C7 信號及第七號信號系統。SS7 信號共分四層，第一層為信號數據鏈路功能層、第二層信號鏈路功能層、第三層信號網路功能層及第四層用戶部，其中信號網路功能層又分為信號訊息處理及信號網路管理兩個部份，用戶部又分電話用戶部(TUP)及 ISDN 用戶部(ISUP)。SS7 信號系統架構如下圖所示：



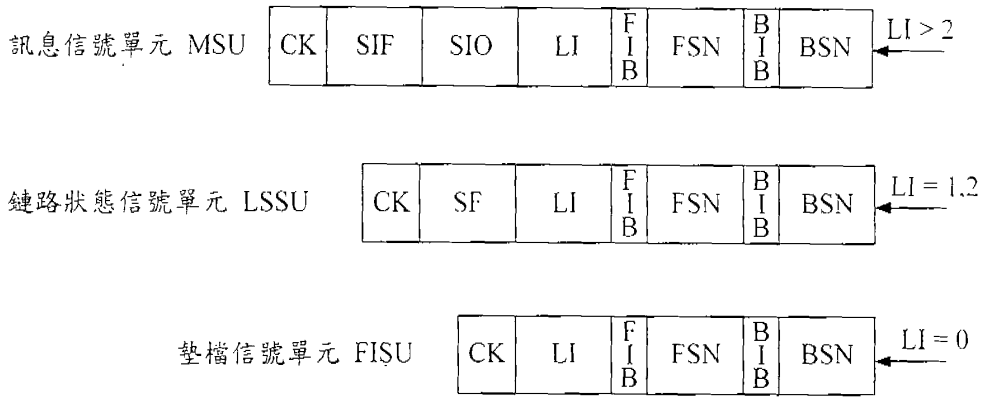
SS7 信號系統架構

第一層信號數據鏈路功能層(Signalling Data Link Function)定義信號數據鏈路的機械、電氣、功能及程序等特性，機械特性包含連接頭的大小、形狀及接腳數目等。電氣特性包含傳輸速度、電壓和電流大小及時序等。功能特性則定義各種電路的意義及信號之間的關係、程序則是依據功能特性，規定資料傳送的流程。因此，第一層是負責資料傳送的實體鏈路，而在數位化環境中，通常使用 64k bit/s 的信號數據鏈路。



信號鏈路功能方塊圖

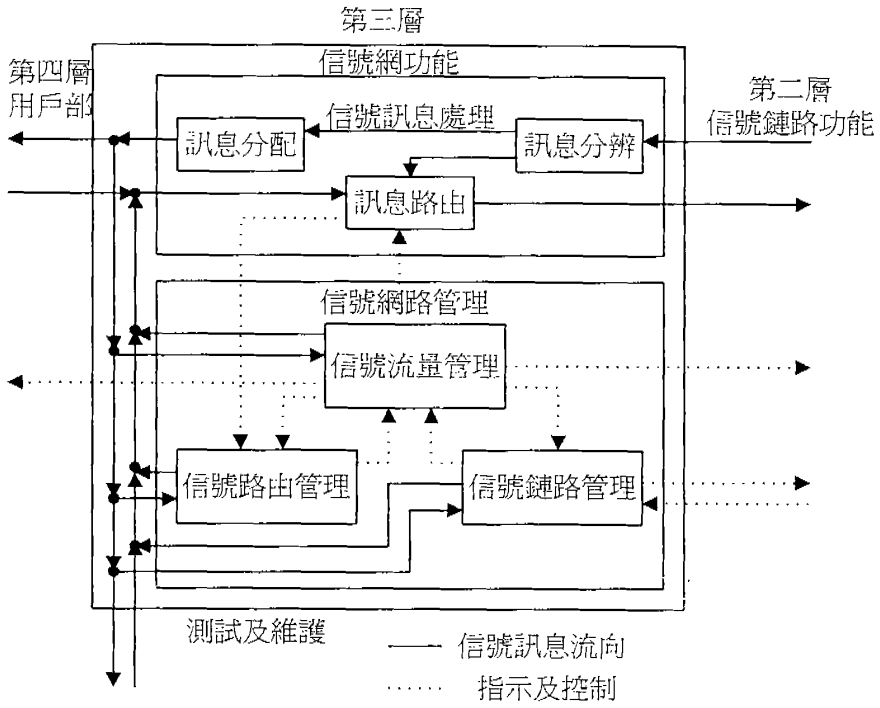
第二層信號鏈路功能層主要功能可細分為信號單元的界定(DELIMITATION)、信號單元的校驗(ALIGNMENT)、信號單元誤碼偵測(ERROR DETECTION)、信號單元誤碼更正(ERROR CORRECTION)、初始校驗(INITIAL ALIGNMENT)、信號單元誤碼率監控(SUERM)及流量控制(FLOW CONTROL)。信號單元區分為訊息信號單元(MSU，Message Signal Unit)、鏈路狀態信號單元(LSSU，Link Status Signal Unit) 及墊檔信號單元(FISU，File In Signal Unit)三種，訊息信號單元為實際承載信號訊息資訊之 SU；鏈路狀態信號單元為表示鏈路狀態之 SU；墊檔信號單元為無訊息信號單元及鏈路狀態信號單元可送時所送之 SU。下圖即為三種信號單元之碼框結構。



信號單元的識別

誤碼更正依使用距離與環境的不同，可採用基本誤碼更正法或預防性循環重送誤碼更正法。當短距離通訊時可採用基本誤碼更正法，用正、負確認及重送方式來更正錯誤；於長距離通訊、衛星通訊時，採用預防性循環重送誤碼更正法，當發送端無新的 SU 可送時，將未收到正確認可之 SU 循環重送。

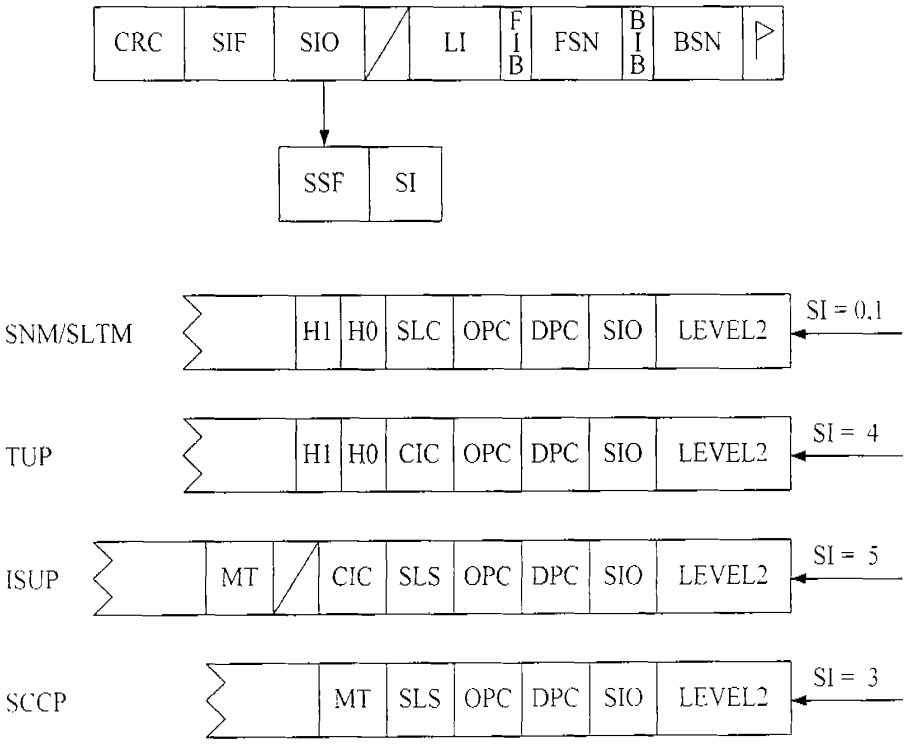
第三層信號網路功能層分為信號訊息處理及信號網路管理。信號訊息處理主要在處理信號訊息的流程；經由訊息分辨、訊息分配及訊息路由功能，將訊息正確送到目的地。信號網路管理主要在網路發生障礙時，提供或管制信號訊息的流向，以提高信號網路訊息疏通的能力，維持網路正常的運作。



圖四信號網路功能方塊圖

信號訊息處理的訊息分辨功能用於分辨信號訊息目的地為本信號點或其他信號點；訊息分配功能依服務指示將信號訊息傳送至用戶部；訊息路由功能決定輸出鏈路，將信號訊息送至第二層，以送達目的地信號點。

信號網路管理分為信號話務管理、信號路由管理及信號鏈路管理。信號話務管理用於在鏈路發生故障時，分散路由話務量，使信號流量暢通；信號路由管理決定信號訊息是否可經由信號轉接點轉接；信號鏈路管理負責鏈路的啟用、恢復、停用等功能。下圖為信號網路信息碼框結構。



信號網路訊息格式圖

信號點編號計劃：在網路內，任一信號點(SP，STP)均有其特定的信號點碼。

ITU 國際信號點編號規定如下：

國際信號點碼為 14 比次

區帶	區／網路名	信號點
3	8	3

區帶：將全球分成六個區帶：歐洲、北美洲、亞洲、南太平洋洲、南美洲、非洲。

區／網路名：區帶內再依地區／ 國家／網路予以區分。

信號點：每個網路最多可有 8 個國際信號點。

信號接續控制部（SCCP，Signalling Cconnection Control Part）介於第三、四層間，用於傳送電路相關及非電路相關信號訊息之接續導向網路服務及非接續型之網路服務。

第四層用戶部細分為整體服務數位網路應用部（ISUP，Intergrated Sservice digital network Uuser Part）、電話用戶部（TUP，Telephone User Part）及訊息交易部（TC，Transaction Capabilities）

電話用戶部（TUP，Telephone User Part）中較值得注意的是服務資訊位元組（SIO，Service Information Octet）、路由標記（Label）及電路識別碼（CIC，Circuit Intentification Code）。服務資訊位元組之格式如下。

D C B A	1 0 0	
次服務欄（Subservice）	服務指標（Service Indicator）	首次傳送之比次

服務指標（SI）編成)0100

次服務欄

- 比次 BA：為未來發展所預留，其對於國際應用部及 MTP 第三層需共同的解決方式，係編碼成"00"
- 比次 DC：網路指標：
  - 00：網路指標
  - 01：保留供國際使用
  - 10：國內網路
  - 11：保留供國內使用

標準的電話路由標記長度為 40 比次，並置放於信號茲訊欄之起頭，其格式如下：依據標準之路由標記結構，每一個電話教換局均為一信號點且付予唯一之信號點碼。

CIC	OPC	DPC
12 比次	14 比次	14 比次

DPC：受信號碼（Destination Point Code）

OPC：發信號碼（Originating Point Code）

CIC：電路識別碼（CIC，Circuit Intentification Code）

個別電話電路間之電路識別碼指定係依據雙方協議或事先訂定的規則，規劃原則依特定應用而不同：

- 2048KBPS 數位路由：導引自 E1 數位系統，電路識別碼在最低效的 5 個位元，代表設定於通話電路時槽的實際號碼。
- 8448KBPS 數位路由：電路識別碼在最低效的 7 個位元，代表設定通話路由之通道 chennel，其編碼情況如下：

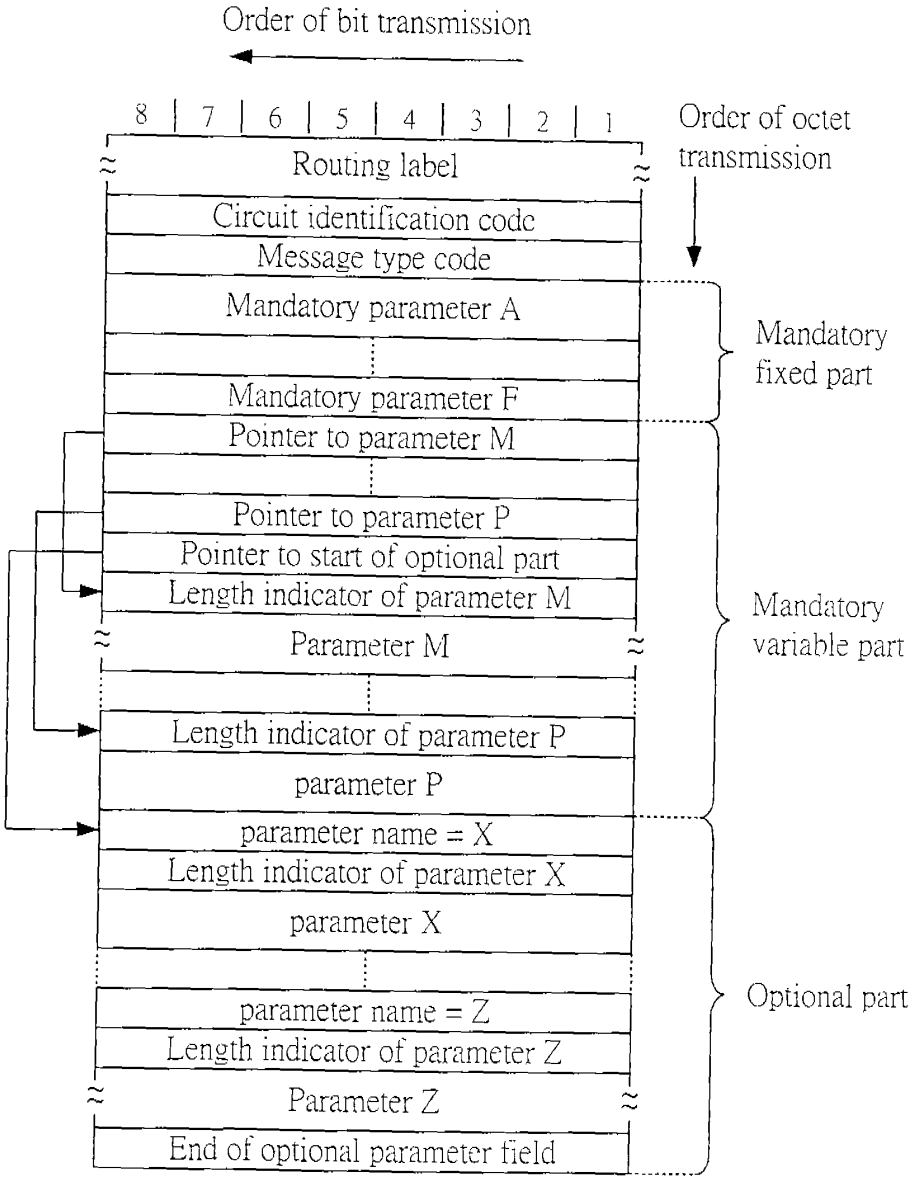
0000000	chennel 1
0000001	chennel 2
:	:
:	:
0011111	chennel 32
0100000	chennel 33
:	:
:	:
1111110	chennel 127
1111111	chennel 128

- 劃頻多工（FDM，Frequency Division Multiplex）係利用2048KBPS 碼調變標準，CIC 在最低的 6 個位元，代表 60 通道的一個通道，分成 5 基本 FDM 群，設定如下：

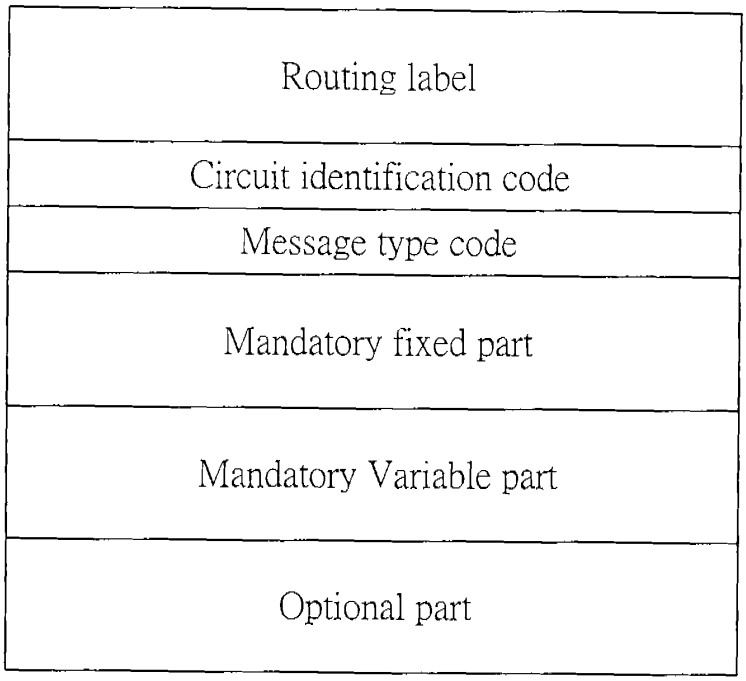
000001 : 001100	1st basic（FDM）group
001101 : 011001	2st basic（FDM）group
011010 : 100110	3st basic（FDM）group
100111 : 110011	4st basic（FDM）group
110100 : 111111	5st basic（FDM）group

ISUP 信息被承載於信息鏈路(Signaling Link)時係以 MSU 格式來傳送，其 SIO 之 SI 值為 0101，其信息格式如下：

表二 ISUP 訊息格式



上表所帶有之ISUP信息長度為位元組之整數倍（8xn 比次，n為整數），可包括下列部份：



表一 ISUP 信訊息格式分類

2 H.323 規範：

國際電信聯盟（ITU-T）於 1996 年制定了 H.323 規範，H.323 標準訂立了一套適用於 IP 網路架構上多媒體系統之溝通模式，係定義在一個沒有提供服務品質（Quility of Service）保證的區域網路環境中，各類資料型態所使用的標準與轉換。有了這個網際網路電話的世界性標準，互通性已穫得解決。H.323 規範提供了網際網路電話間（包含網際網路視訊即時傳輸）間互通之整體解決方案，也定義了讓網際網路電話與公眾服務電話網路（PSTN）上傳統電話機互通之規範，包括：

- 透過 B-ISDN（寬頻整合服務數位網）與 H.310/H.321 終端機溝通。
- 透過 N-ISDN(窄頻整合服務數位網)與 H.320 終端機溝通。
- 透過提供服務品質（QoS）保證的區域網路（LAN）環境與 H.322 終端機溝通。
- 透過 GSTN 與 H.324 終端機溝通。

在 H.323 規範定義了各種所有可終端設備及特性，包括終端機(Terminal)、閘道（Gateway）閘道管理者（GateKeeper）、多方會談控制單元（MCU）及這些元件間或這些元件與外界元件之間彼此溝通所需之訊息與程序，即 RAS 協定、Q.931 通話件立協定及 H.245 控制協定及媒體傳輸協定。現介紹各 H.323 規範元件及元件間之相關通話協定如下（H.323 規範為主從式架構。）

2.1 終端機(Terminal)：

終端機主要功能是在網路上傳遞即時性的語言（須具有 G.711、G.723、G.729 等語音壓縮解壓縮（Codec）能力）、視訊（須具有 H.261、H.263 視訊壓縮解壓縮能力）以及非即時性之資料（須具有 T.120 資料傳輸協定(Option)），如網際網路電話即是一種終端機(Terminal)。為了讓終端機與其它 H.323 元件能有標準之溝通方式，終端機必須支援 ITU 另外所制定的 H.245 控

制協定，Q.931 通話建立協定及 RAS 協定。

2.2 閘道（Gateway）：

只有當終端機(Terminal) 與外界元件（如 GSTN 之 H.324 Terminal、N-ISTN 之 H.320 Terminal、B-ISTN 之 H.321 Terminal）通話時才用得到，它是連接分封交換網路（Packet Switch Network）與線路交換網路（Circuit Switch Network）的橋樑，負責兩相異網路間傳輸格式（包括視訊、語音壓縮解壓縮格式）及溝通程序上差異之轉換，由相對的角度看來，可把閘道（Gateway）看成一個 H.323 Terminal 或 B-ISTN 之 H.321 Terminal。

2.3 閘道管理者（GateKeeper）：

閘道管理者負責終端機(Terminal)、閘道（Gateway）及多方會談控制單元（MCU）等位址之轉換及頻寬之管理。

這些由閘道管理者所管理元件所成的集合稱為區域(Zone)，於區域(Zone)內所有 H.323 元件在啟動時必須向閘道管理者註冊其別名（Alias）與網路位址，當一區域(Zone)內之 H.323 元件想與其它元件建立通話管道；如區域(Zone)內終端機(Terminal)A 想透過閘道（Gateway）B 與區域外之 B-ISTN 之 H.321 Terminal 聯絡，終端機 A 必須先和閘道 B 建立通話管道，終端機 A 可根據閘道 B 之別名向閘道管理者提出申請，包括查詢閘道 B 之網路位址及提出使用頻寬之申請，閘道管理者可以根據目前頻寬之使用狀況批准或拒絕其請求，若批准使用頻寬，雙方即可依此建立通話管道。是以閘道管理者為中心之主從式架構。

2.4 多方會談控制單元（MCU）：

H.323 規範也提供了主從式架構的多方會談的功能，於三個以上的終端機彼此同時交談，即使用多方會談控制元件為中心來控制其它參與會談之元件，使多方會談能順利進行。

多方會談控制單元包含多點控制器（Multipoint Controller）及媒體處理器（Media Processor），多點控制器負責協調與控制參與多方會談的元件，包括決定多方會談中元件間資料、語音及視訊傳輸方式採用單點或多點傳輸，及每一元件所應採用的壓縮解壓縮格式。媒體處理器負責處理媒體，如混音、視訊合成及資料處理，若於一多方會談中沒有媒體處理需要時，則不需要媒體處理器。所以一個多方會談控制單元包含一個多點控制器（Multipoint Controller）及零或多個媒體處理器（Media Processor）。

2.5 RAS 協定：

RAS 協定即為註冊（Registration）、加入（Admission）及狀態查詢（Status）協定，為各 H.323 元件與閘道管理者之間信號交換所使用之協定。H.323 元件在啟動時必須使用 RAS 協定向閘道管理者註冊其別名（Alias）與網路位址，此即為註冊。H.323 元件在與其它元件建立通話管道前必須使用 RAS 協定告知閘道管理者通話所需之頻寬及對方的別名，閘道管理者可以根據目前頻寬之使用狀況批准或拒絕其請求，並告知對方的網路位址，此即為加入（Admission）。閘道管理者可以透過 RAS 協定查詢 H.323 元件是否正常運作及其通話狀態，此即為狀態查詢（Status）。除此三大功能外，H.323 元件也可以透過 RAS 協定找尋閘道管理者或是更改通話所需之頻寬。

RAS 協定依照 H.225.0 之資料格式標準提供訊息(Message)封包，分類條列如下：

- 註冊（Registration）：
  - RRQ：要求註冊（Registration Reguest）為終端機=>閘道管理員
  - RCF：同意註冊（Registration Confirm）為閘道管理員=>終端機

- RRJ：拒絕註冊（Registration Reject）為閘道管理員=>終端機
- URQ：要求取消註冊（Unegistration Reguest）為終端機=>閘道管理員
- UCF：同意取消註冊（Unegistration Confirm）為閘道管理員=>終端機
- URJ：拒絕取消註冊（Unegistration Reject）為閘道管理員=>終端機

■ 加入（Admission）：

- ARQ：要求加入（Admission Reguest）為終端機=>閘道管理員
- ACF：同意加入（Admission Confirm）為閘道管理員=>終端機
- ARJ：拒絕加入（Admission Reject）為閘道管理員=>終端機

■ 狀態查詢（Status）：

- IRQ：查詢狀態（Information Reguest）為閘道管理員=>終端機
- IRR：回報狀態（Information Rerun）為終端機=>閘道管理員
- IACK：為閘道管理員=>終端機
- INAK：為閘道管理員=>終端機
- DRQ：要求中斷(Disconnect Reguest) 為閘道管理員<=>終端機
- DCF：同意中斷(Disconnect Confirm) 為閘道管理員<=>終端機
- DRJ：拒絕中斷(Disconnect Reject) 為閘道管理員<=>終

端機

- LRQ：（Location Reguest）為閘道管理員=>終端機
- LCF：（Location Confirm）為閘道管理員=>終端機
- LRJ：（Location Reject）為閘道管理員=>終端機

■ 其它：

- NSM：為閘道管理員<=>終端機
- RAI：為終端機=>閘道管理員
- XRS：為終端機=>閘道管理員
- RIP：為閘道管理員=>終端機
- RAC：為閘道管理員=>終端機
- XRS：為閘道管理員=>終端機

2.6 Q.931 通話建立協定：

H.323 協定使用的 Q.931 通話建立協定為原用於線路交換網路的修正版，於訊息格式及 Terminal 方面因應線路與分封網路之差異作修正外，其它如運作方式與原線路交換網路的 Q.931 通話建立協定大致相同。

Q.931 通話建立協定的主要訊息(Message)包括：

- Setup：要求建立通話管道
- Call Proceeding：通話管道建立中
- Alerting：振鈴中
- Connect：通話管道建立成功
- Release Complete：中斷通話管道

Q.931 訊息有兩種傳遞方式：

- Direct Call Signaling：呼叫端（H.323 元件）與被呼叫端（H.323 元件）直接傳遞
- Gatekeeper Routed Call Signaling：呼叫端（H.323 元件）與被呼叫端（H.323 元件）透過閘道管理者（GatekKeeper）傳遞

由閘道管理者決定採用何者。就兩種傳遞方式舉例如下：

2.7 H.245 控制協定：

H.245 控制協定主要包括交換能力（Capability Exchange）、決定主從關係（Master/Slave Determination）及開關邏輯通道（Logical Channel）。

- 交換能力（Capability Exchange）：於 H.323 系統中，每個 H.323 元件因設計目地不同，可能都擁有許多但彼此不盡相同之壓縮解壓縮能力，因此於傳輸語音、視訊及資料前，必須先協調出雙方都能接受之壓縮解壓縮格式，這即是運用 H.245 控制協定之交換能力（Capability Exchange）來達成。
- 決定主從關係（Master/Slave Determination）：於 H.323 協定中，一個多方會談只包含一個多點控制器（Multipoint Controller）來協調及零或多個媒體處理器（Media Processor）來處理媒體。當多方會談成員有二個以上之多點控制器時，就需依賴 H.245 控制協定來決定主從關係，然後由主多點控制器來掌控多方會談。
- 開關邏輯通道（Logical Channel）：媒體的傳輸需使用媒體管道（Media Channel），但因媒體管道之傳輸位址（Transport address）是不固定的，為了讓傳送端瞭解接收端所使用的媒體管道傳輸位址，於是就有開啟關邏輯通道之需求產生，開啟邏輯通道之目地是讓彼此了解對方媒體管道之傳輸位址；如接收端回應傳送端開啟邏輯通道的請求時，必須告訴其媒體管道傳輸位址。媒體傳輸協定：

  於 H.323 規範中，媒體傳輸協定有兩類：

  - RTP（Real-Time Transport Protocol，即時傳輸協定）/RTCP（RTP Control Protocol，即時傳輸控制協定）：用

來傳輸語音及視訊等即時性媒體。RTP 協定可以應用網路上如 UDP 等不可靠但快速之通訊協定即時一對一或一對多傳輸語音及視訊等允許少量誤差之媒體，再介由 RTCP 協定由送收端統計出資料遺失量，進而了解網路目前的服務品質，作為調整網路頻寬來達到服務品質之參考。

- T.120 資料傳輸協定：傳輸非即時性的資料，可一對一或一對多傳輸非即時性但可靠的資料。

2.8 H.323 元件通話模型之一：

以 H.323 元件（如 Terminal A）與另一同區域(Zone)內終端機(Terminal B)通話為例，其可以使用下列之程序為建立通話之方法之一：

- Terminal A 使用 RAS 協定向閘道管理者（GatekKeeper）要求加入；由 Terminal A 向閘道管理者傳送 H.225 格式之 ARQ 封包。
- 閘道管理者用 RAS 協定回復 Terminal A 同意加入；由閘道管理者向 Terminal A 傳送 H.225 格式之 ACF 封包。
- Terminal A 使用 Q,931 協定向閘道管理者要求與 Terminal B 建立通話管道；為 Q.931 開啟 TCP 通道。
- 閘道管理者使用 Q,931 協定向 Terminal B 提出 Terminal A 要求建立通話管道。
- Terminal B 使用 RAS 協定向閘道管理者要求加入；由 Terminal B 向閘道管理者傳送 H.225 格式之 ARQ 封包。
- 閘道管理者用 RAS 協定回復 Terminal B 同意加入；由閘道管理者向 Terminal B 傳送 H.225 格式之 ACF 封包。
- Terminal B 使用 Q.931 協定通知閘道管理者通話管道建立成功；傳送 H.225 格式之 Q.931 connect 封包。
- 閘道管理者使用 Q,931 協定告知 Terminal A 通話管道建

立成功；傳送 H.225 格式之 Q.931 connect 封包。

- Terminal A 使用 H.245 協定直接向 Terminal B 交換能力資訊及決定主從關係，及開啟邏輯通道；使用 H.245 格式傳送 Terminal Capacity 封包。
- Terminal A 及 Terminal B 各自開啟 RTP 及 RTCP 管道；傳送 H.225 格式之 RTP 及 RTCP 封包。
- 雙方通話

3 SIP（Session Initiation Protocol；議程起始協定）

SIP 是由 IETF（Internet Engineering Task Force）所制定，也是網際網路會議或網際網路電話之間溝通的一種信號傳送協定，用於邀請使用者加入網路上的多媒體系統，具有將會議之邀約轉送使用者目前位址之功能。SIP 具有如 H.323 協訂之相類似功能，但有較簡潔之內容定義方式。

SIP 是設立於網路應用層的協定，其功能包括建立、修改、及終結多媒體如多媒體會議、遠距教學及網路電話等的交流或通話。SIP 可邀約加入單播式（unicast）或群播式（multicast）的會議，且邀約的一方不需是會議中的一員，受邀者可包括通話的人或機器，如邀請媒體儲存裝置製目前的議程，或請求隨選視訊伺服器於會議中播放視訊資料，但 SIP 並不直接控制此類服務。

SIP 可根據會議公告發起議程或邀約成員，會議公告的方式可利用群播協定，如會議公告協定（Session Announcement Protocol；SAP）、電子郵件（electronic mail）、新聞群組（news groups）、網際網業（web pages）或目錄服務系統（Lightweight Directory Access Protocol；LDAP）等。SIP 支援以下幾種多媒體通訊的建立與終結：

- User Location：決定通訊的終端系統。
- User Capabilities：決定使用的媒體及媒體參數。
- User Availability：決定受話方是否有意願加入通訊。
- Call Setup：建立撥號方與受話方兩端之通話參數。
- Call Handling：包括通話的傳送與終結。

SIP 網路結構包括如下：

- Clients（用戶端）：即傳送 SIP Requests（請求）的應用

程式，用戶端也可兼 UAC（User Agent Client（使用者代理用戶端））及 SIP Proxies（代理主機）

Servers（伺服器）：即接受請求、服務請求並且回答（Response）請求的應用程式，伺服器也可兼 UAS（User Agent Server（使用者代理伺服器））、SIP Proxies（代理主機）及 Redirect Server（改向伺服器）

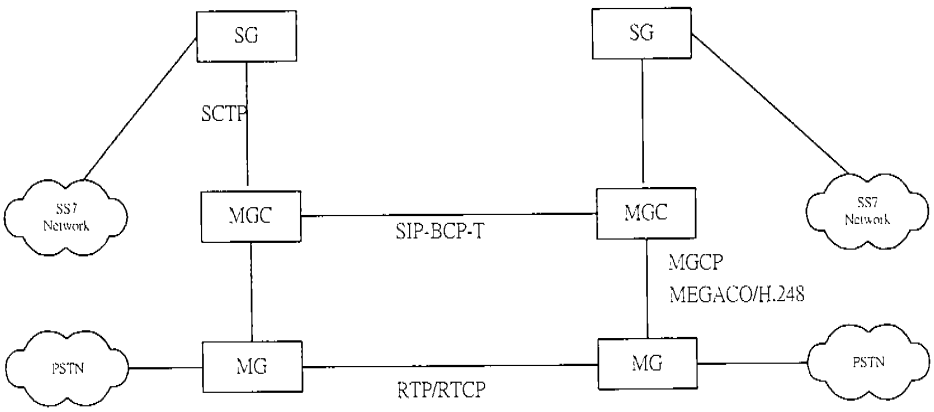
4 SS7 信號於 IP 網路之應用

VoIP 與 PSTN 兩個相異系統現趨向整合，如何將 PSTN 中具備先進電話服務所必備之 SS7 信號整合到 VoIP 系統內，以便於電話系統整合到 VoIP 後，仍能提供各式各樣先進電話功能。

MGCP（Media Gateway Control Protocol；媒體閘道控制協定）於 1999 年發表為 RFC2750，1998 年 LUCENT 公司發表類似用途之標準，名為媒體設備控制協定（Media Device Control Protocol：MDCP），之後制定 MGCP 與 MDCP 的相關廠商協議將兩者合一制定單一標準，名為 MEGACO（MEdia GAteway COntrol），在 IUT-T 中制定為 H.248，在 MEGACO 的架構下，新一代的包含 SS7 信號之網際網路電信系統架構已見雛形，如下圖所示。

- SG（Signaling Gateway）：為信令閘道，負責與 PSTN SS7 信號系統溝通，取得通話相關信息，並提供給 MGC 作通話控制用。如原電信網路之 SS7 信號網路將 SS7 信號傳到近端之 SG，SG 將之轉換成以 IP 網路為基礎之簡易控制傳輸協定（SCTP，Simple Control Transmission Protocol）傳送到 MGC，然後傳到遠方 MGC 後，遠端 MGC 再用 SCTP 將 IP Base 之 SS7 信號傳到遠端 SG，遠端 SG 再還原為 SS7 信號傳到遠端 SS7 網路。
- MG（Media Gateway）：為媒體閘道，只負責電話建立好後，語音資料在 IP 網路之傳輸，如 PSTN 語音送往 MG 後轉為語音封包，送往遠端的 MG 後，再由遠端 MG 轉成語音送往目的 PSTN 網路。為求符合 QoS（Quality of Service）的要求，MG 之間需要執行 RTP（Real-Time Transport Protocol）及 RTCP（RTP Control Protocol）等相關協定。

- MGC（Media Gateway Control）：為媒體閘道控制器，負責執行通話控制功能，負責在 VoIP 系統中建立通話。當收到由 SG 傳來之 SS7 信號，然後利用 H.248 來控制 MG 傳送聲音信號。



5 Natural Access

Natural Access 包括 CT Access 及支援 AG 硬體的服務（ADI service），CT Access 為不相依於硬體的電話功能函數標準程式介面的發展環境，ADI 服務是於 MMS AG 卡板或 QX 卡板發展電話應用程式的函數程式館。

CT Access 結構：CT Access 是電話技術應用之發展環境，提供如下之內容函數：

- 提供電話技術功能的標準組合，並被組織成合邏輯的服務：CT Access 服務提供下列功能，建立與維護網路連接，制止呼叫狀態，錄放聲音訊息，產生及檢知 DTMF 及音頻信號，也可提供控制板間之 MVIP（Multi-Vendor Integration Prptocol）交換區塊之服務。
- 提供不相依於硬體之服務標準應用程式介面（API，Applivation Programming Interface）：CT Acces 提供標準應用程式介面，只要符合其標準應用程式介面之 Third party 設計之服務也能供 CT Acces 平台使用，且 CT Acces 設計成可於不同之硬體上執行相同之服務，即應用程式只需接受具備相同程式介面之服務函數，並不需知道是那一種硬體作為服務之工具。於 CT Acces 發展環境，不必為不同之硬體撰寫不同之程式，而是關注在使用同一應用程式介面之服務。
- 提供接受與修改服務參數之標準介面：所有相同功能之服務都有標準之參數組合，這些服務參數之預設值滿足大多之配置，但於特殊之配置，參數能被修改以開始或關掉某些特性。CT Acces 提供函數標準組合來恢復或修改服務參數。
- 提供捕捉錯誤之特性：CT Acces 提供一種機構，可紀錄由應用程式定義的錯誤處理
- 提供能配置所需資源之結構：當 CT Acces 啟始化，應用程式

定義應用所須之服務，只有這些服務所須之資源被配置到 CT Access。

- 提供多工程式模型，以配合許多應用之需要：CT Acces 的結構允許多工及單工模型以配何所有之應用程式需求。
- 提供如下多工處理函數：
  - 可由同一個 CT Access 脈絡來完成多樣的 CT Acces 應用程序工作。（參考 context sharing）
  - CT Access 應用程式（與各別電話呼叫脈絡聯合）可將 CT Access 脈絡之控制權傳給其它的 CT Access 應用程式（掛斷脈絡）

CT Access 包含下列元件：CT Access 核心函數（提供載入與配置服務的方法）、CT Access 服務及服務管理（服務提供相關聯的電話技術函數群體，服務管理提供服務的標準介面）、CT Acces 伺服（ctdaemon）（致能共享服務資源之應用）。

CT Access 核心函數：當用 CT Access 建立應用時，使用 CT Access 核心函數設定操作環境，使用服務的應用程式介面來使用電話技術函數的權力。CT Access 提供標準化的介面來完成工作，例如：

- 啟死始化 CT Access 環境
- 建立及消滅 context 及事件 queue
- 開啟及關掉服務們
- 控制參數、事件，及追蹤

CT Access 分為伺服及程式館方式：程式館方式（預設）：每個應用程式連結它擁有的 CT Access 共享程式館，應用程式建立與獨特管理它擁有的程度脈絡，資源不允許在應用程式間共享。每個應用程式管理個自分離的 CT Access 程式環境。伺服方式：CT Acces 伺服（ctdaemon）代表顧客應用程式建立及管理程序脈絡，多樣的應用程式能共享及掛上共通程序資源。顧客應用

程式透過 ctdaemon 來共享 CT Access 程序資源，顧客應用程式持續透過服務應用程式介面（API，Applivation Programming Interface）來完成工作，ctdaemon 對應用程式好像隱形一般，每個應用承式就如直接與它擁有的程序資源交流。

服務：一個服務就是一邏輯上相關聯的電話技術函數群體，一個服務也許配置於多種的硬體卡板，不論那種硬體提供此函數，所有具備同功能之服務具有一標準的 API。如此允許硬體與應用互相獨立。每個服務定義了由 CT Access 管理的參數組，稱之為標準參數。同樣的服務配置在不同的硬體平台也許新增參數來處理這特殊卡板之獨特函數，稱之為延展參數。CT Access 提供進入及修改此二種參數的能力。CT Access 提供下面的標準服務：

- 交換服務（SWI，Switching）：提供於電話卡板控告 MVIP 交換區塊之函數。
  - 服務管理：SWIMGR
  - 服務 ID：0x4
  - 硬體需求：任何使用 MVIP-90 或 MVIP-95 驅動之卡板。
- 聲音訊息服務（VCE，Voice Message）：提供錄放及編輯聲音訊息的一組函數服務，如使用 ADI 服務與硬體介面。
  - 服務管理：VCEMGR
  - 服務 ID：0x3
  - 硬體需求：任何提供錄放服務之卡板
- AG 裝置服務（ADI，AG Device）：提供裝置等級的電話函數，如錄放、呼叫程序、音頻檢知、音頻產生、DTMF 收集、能源檢知、FSK 資料傳送及接受、和計時。
  - 服務管理：ADIMGR，QDIMGR（單獨可用）
  - 服務 ID：0x1C
  - 硬體需求：AG 系列及 QX 系列卡板
- Natural 呼叫控制服務（NCC，Natural Call Control）：提供

標準呼叫控制服務，也提供電話傳輸（Call transfer）的呼叫控制（call control）特性。

- 服務管理：ADIMGR，QDIMGR（單獨可用）
- 服務 ID：0x1
- 硬體需求：AG 系列及 QX 系列卡板

- 參數管理服務（PRM，Parameter Management）：提供操作由應用定義的參數功能。
  - 服務管理：PRM
  - 服務 ID：0xE
  - 硬體需求：與硬體無關
- 點對點交換服務（PPX，Point to Point Switching）：提供以電話匯流排連接板子間之交換函數。
  - 服務管理：PPX
  - 服務 ID：0x17
  - 硬體需求：任何使用 MVIP-90 或 MVIP-95 驅動之卡板。
- 數位通信波道監視服務（DTM，Digital Trunk Monitor）：提供 T1 及 E1 幹線告警監視函數，及採集效率的統計值。
  - 服務管理：ADIMGR
  - 服務 ID：0xXC
  - 硬體需求：AG 系列及 QX 系列卡板

服務管理：服務管理以軟體形別圍繞一或多個服務封裝，如此使服務能無縫的在 CT Access 內整合，服務管理在 NT 環境下以動態連結程式館（DDL，Dynamic link libraris）方式應用。

CT Access 伺服（ctdaemon）：使客戶應用程式能於伺服模式執行，代表應用程式建立和管理程序脈絡（processing context），使應用程式們共享及脫離伺服資源。Ctdaemon 也允許開發者修改系統預設的整體參數，設置整體追蹤標記，以爪哇 applet 執行遠端追蹤，可選擇將追蹤訊息紀錄到一檔案。當執行伺服模式，

AG.CFG 檔必須包括期望用於任何和所有應用程式的所有服務/服務管理對，若沒依照上述要求，於呼叫 ctaOpenServices 時會出現 CTAERR_NOT_FOUND 的錯誤。

操作原則：非同步程式模式及組織應用程式適當方法的 CT Access 操作基本原則。非同步程式模式：CT Access 使得非同步程式模式可得到同步處理的好處。應用程式使用服務的 API 函數，當執行呼叫服務，大多 API 函數立刻回應，若得到 SUCCESS 表示函數已初始化，當 CT Access 正處理這個命令時，應用程式可能然後執行其他函數。CT Access 將命令傳給服務，服務就傳送命令給電話卡板。電話卡板執行所要求之函數，送回事件給服務來指出執行的狀態（如函數開始，函數完成）。服務送事件給 CT Access，如此提供可用給應用程式。事件：所有 CT Access 函數傳回狀態或碼，於非同步函數，傳回碼指出是否函數已初始化。當執行函數，將產生事件來指出抹某些情況的發生或狀態改變。事件以 CTA_EVENT C 資料結構表示，包含如下欄位：

- id：為 CT Access 事件的事件碼，所有 CT Access 事件碼以 XXXEVN_加在前面，XXX 為服務的三字母縮寫（如 VCEEVN_PLAY_DONE）
- ctahd：CT Access 脈絡處理（產生事件，如由 ctaCreateContext，ctaCreateContextEx，ctaAttachContext.傳回）。
- timestamp：時間印記，以 milliseconds 為單位，當事件發生，AG 卡板之事件解析度是 10 milliseconds。
- userid：為提供給 ctaCreateContext 或 ctaCreateContextEx 的使用者 CT Access 脈絡值。
- value：用於和單 32-bit 資料通訊的事件特定值。
- size：由緩衝器指定的區域尺寸（以位元組為單位），這欄位也指出與事件關聯的資料緩衝器是否須用 ctaFreeBuffer 釋放。如

果這緩衝器是 NULL，這欄位也許用來控制一事件特定值。

- buffer：如緩衝器傳回事件，這欄位指向事件特定資料。當事件的尺寸欄位含有緩衝器的實際尺寸，這欄位包含一應用程式緩衝器位址。
- objhd：服務物件處理（如 swihd，vcehd，callhd）

CT Access Context（脈絡）：CT Access Context 於單一 CT Access Contex 組織服務及伴隨的資源，一個 context 通常表示一個應用的要求，例如控制一個單一的電話呼叫。有些 context 並不與電話相關聯，例如聲音轉換的應用並不需要電話線。開啟服務要求（Opening Service Instance）：開啟服務脈絡使服務函數連上 CT Access Context，建立服務要求（Service Instance）。一個服務要求定義一個服務（如一函數的邏輯群體）及伴隨的資源（如 MVIP 卡板，流（stream），時槽（timeslot）），一個 CT Access Context 是服務要求的界限集合，它以狀態（state）封裝函數與資源群體，建立服務相依，及處理參數管理。於一 CT Access Context，一個服務可能只開啟一次，例如於 AG 資源卡板所有 24 通道都能使用 ADI 服務函數，直每個脈絡需要 24 脈絡及 ADI 開始服務。每一種形別的服務都有定義參數，每個服務要求有由 CT Access Context 維護的自備參數被複本，如此允許每一呼教有其自有的特性。Context Sharing and Hand Off（脈絡共享與放手）：當於伺服模式，多工應用能共享共同 CT Access Context 資源。所有共享一特殊脈絡之應用能使用和它關聯的服務要求。例如於一 IVR 系統，當他人經由連接放聲音檔時，一個應用能從或往 PSTN 建立連接。當應用一與應用二使用服務要求（SWI，ADI，VCE）於分離的 context（其中 SWI 為 context 1，ADI，VCE 為 context 2）上執行，應用一與應用二並不會為共享的服務開啟分離的脈絡。於伺服模式，也能使應用互相放手脈絡，應用能以所給的服務要求執行工作，然後放手資源控制給其它應用。例如一應用能執行來話應答之電

話程序後，安排電話程序給其它應用，是因有放手脈絡規則的建立，這第一個應用可以返回等待來話。

事件佇列（Event Queue）：事件佇列是事件與服務間的通訊路徑，一服務產生事件來指出某種情況或狀態改變，應用程式由事件佇列取回事件。當事件佇列建立，可定義服務管理連接事件佇列，只有於 CT Access Context 開啟且提供特殊服務管理的服務允許連接事件佇列。（如果於 CT Access 的 CTA.CFG 檔之預設服務管理值為 NULL，當於伺服模式，CT Access 於建立事件佇列時將忽略定義服務管理）。當建立 CT Access Context 時定義事件佇列，所有由脈絡服務產生之事件送往與脈絡關聯的事件佇列。全部的應用程式可建立一個或多個事件佇列，依所使用的程式模式來決定，一些模式使用一個事件佇列處理所有 CT Access Context，如處理多電話呼叫使用同一事件佇列，其它模式每個呼叫使用一個事件佇列。

命令程序（Command Processing）：命令程序以下列順序進行。

- 應用程式求助於具有 VCE 函數的 CT Access Context 處理。
- 當執行這命令時，VCE 服務要求接受它的 CT Access Context 參數。
- 因為 VCE 服務要求硬體資源，且提供錄放函數之 ADI 服務在同一脈絡內，所以 VCE 服務送命令給 ADI 服務要求。
- 當執行這命令時，ADI 服務接受它的 Context 參數。
- ADI 服務送命令給 AG 卡板驅動程式。

事件程序（Event Processing）：CT Access 事件程序以下列順序進行。

- 應用程式呼叫 ctaWaitEvent 來等待需由 CT Access 服務處理的非同部事件。
- ADI 服務接到由 AG 卡板驅動程式所送的卡板事件。

- ADI 服務處理卡板事件，然後送出 ADI 事件給 VCE 服務要求。
- VCE 服務處理 ADI 事件，然後送出 VCE 事件給關聯於 CT Access Context 的事件佇列。
- 註：應用程式必須以及時的方法呼叫 ctaWaitEvent，以確保處理對時間挑剔的事件，這對 CT Access 處理內部事件很重要。

應用程式發展：程式模式：能再進入程式館可於程式模式滿足大多的應用需求，CT Access 是能再進入程式館的集合，程式模式支持單線（Single-thread）及多線（Multi-Thread）操作系統，Win NT 即為多線之操作系統。CT Access 及事件佇列可建立不同的操作模式。如下：

- 每個應用程式使用一個脈絡及一個事件佇列：為最簡單的應用發展，當使用多個應用程式時，每個應用程式有它自己的事件佇列。這種配置主要用於單線操作系統的應用，適合用於具少數 CT Access context 之系統。使用這種配置，應用程式能夠建立同步封套，允許執行循序無國籍線。當於伺服模式，分離的應用可以共享同一脈絡，每個分離的應用有自己的事件佇列。
- 每線一佇列多脈絡：於多線操作系統執行，多事件佇列及多 CT Access 脈絡配置用於應用程式上，每線有它自己的線及一個脈絡，應用程式可以如配置成一個事件佇列，一個脈絡般的以同步函數發展。因為每線控制一事件佇列，它允許一線處於等待服務傳送事件的休息狀態。於共享脈絡環境配置多事件佇列及多 CT Access 脈絡，當於伺服模式，每個應用程式分離的線能關聯多個共享脈絡。(於伺服模式，可執行多線的應用程式)。
- 每個應用程式有一個佇列及多個脈絡：一個佇列及多個 CT Access 脈絡的配置是更有效的利用系統資源，應用計數及檔案

系統資源請求於發展複雜性被最小化。於此種配置，佇列接受所有 CT Access 脈絡傳來的事件，這樣規定一接近的狀態機械模式。當於伺服模式，幾個應用程式可共享相同脈絡資源。

CT Access 應用程式的建立考慮下列幾個步驟：

- 啟始化 CT Access：
  - 註冊服務：有兩種方式，一為求助於 ctaInitialize 和定義服務及服務管理名子，只有經由呼叫 ctaInitialize 定義的服務才能被應用程式使用。二為求助於 ctaInitialize，傳 NULL 給 svcname，傳使用者定義的 CT Access 配置檔名（通常為 CTA.CFG）。當呼叫 ctaInitialize 後，CT Access 知道可供應用程式使用的服務有那些，及服務參數定義及所有可使用服務的相關整體預設值。
  - 設定參數管理策略
  - 致能或禁能追蹤
  - 當內部的等待物件表被改變時，致能或禁能等待物件通知事件
  - 設定是否應用程式於程式模式或伺機模式執行，或是否於執行時經由 CTA.CFG 檔的設定來決定模式。
  - 註：設計於伺服模式執行的應用程式必須於呼叫任何其他 CT Access 函數前先呼叫 ctaInitilize。
- 建立事件佇列：下一步即為呼叫 ***ctaCreatQueue*** 來建立一或多個事件佇列，定義那一個服務管理可關連到每一個佇列（如果與預先定義於 CTA,CFG 檔的不同），當連接或裝訂一服務管理給一佇列，使得此一服務管理可用於此佇列。當服務被開啟後，服務管理將連接佇列，並由服務管理產生事件。由執行 ***ctaCreatQueue*** 函數後返回的事件佇列處理（***ctaqueuehd***）定址事件佇列，當呼叫 ***ctaWaitEvent*** 將提供由佇列取回事件的處理。***CtaCreatQueue*** 於單一程序可能被求助多次，每個求助傳

回獨一無二的佇列處理。建立事件佇列後，CT Access 那個服務可使用此事件佇列（由服務管理所設定），現在可於事件佇列建立脈絡。

- 建立 CT Access 脈絡：應用程式呼叫 ***ctaCreateContext***（或 ***ctaCreateContextEx***）來建立 CT Access context，我們提供由執行 ***ctaCreatQueue*** 函數後返回的佇列處理（***ctaqueuehd***），所有在 CT Access context 上服務的事件將由設定的事件佇列接收。執行 ***ctaCreateContext*** 後返回一 CT Access 脈絡處理（***ctahd***），當應用程式求助於 CT Access 時提供脈絡處理，傳回應用程式的事件也與脈絡相結合。當開啟脈絡時設定選項的使用者設定值（***userid***），這選項的使用者設定值是設計來使非同步程式易於使用，從不被 CT Access 翻譯，當事件被投遞並能被用（取代 ***ctahd***）於有相互關聯的 CT Access 脈絡和應用程式資料結構，使用者設定值被於應用程式資料結構內回音給應用程式。當建立脈絡時也可提供一 CT Access 脈絡名子，當配備共享 CT Access 脈絡及執行追蹤時指派脈絡名子是有用的。CT Access 於脈絡基礎上管理服務參數，CT Access 脈絡將維護所有由派遣者管理及呼叫 ***ctaInitialize*** 定義的服務參數的備份，修改尚未被 ***ctaOpenServices*** 開啟的服務參數就如呼叫 ***ctaInitializ*** 註冊服務，這樣給於應用的便利性，當開啟及關閉服務不會影響脈絡參數值。
- 開啟服務（Opening Services）：應用程式呼叫 ***ctaOpenServices*** 開啟脈絡的服務，傳送脈絡處理及服務描述符號，服務描述符號定義服務名稱、服務管理者及服務特定引數（arguments）（如 MVIP 位址）。於一 CT Access 脈絡開啟服務將建立一服務要求，服務要求定義服務（如一邏輯上的函數群體）及相關的資源（如 MVIP 卡板、流（stream）、時槽（timeslot））。Context 是服務要求的界限集合，，它以狀態（state）封裝函數與資源

群體，建立服務相依（），及處理參數管理。CT Access 脈絡處理（***ctahd***）參照服務，應用程式並不直接定址服務，而以脈絡處理呼叫服務函數，如 **vceOpenFile**（***ctahd...***）的應用程式介面命令提供一定要包括 VCE 服務的 ctahd 要求。呼叫 ***ctaOpenServices*** 是非同步的並能馬上回應，當所有服務已被開啟，一個 CTAEVN_OPEN_SERVICES_DONE 事件將回應給應用程式。CT Access 於共享不足資源的需求基礎上提供開啟與關閉服務，如 Natural 認可/DSP 子板有 16 聲音辨識埠，當需要時這聲音辨識服務可能被開始或關閉，所以這 16 埠也許被用來處理由 AG-T1 卡板的 24 通道電話呼叫。

- 開啟 ADI 服務：當開啟 ADI 服務，必須定義 CT Access 脈絡、特定卡板、時槽及模式。對 AG-24/30/48/60 卡板，也必須定義 MVIP 流。CTA_MVIP_ADDR 參數結構包括下列欄位：卡板、匯流排、流、時槽及模式。匯流排沒使用，除了 AG-24/30/48/60 卡板外之所有卡板，流應設為 0。卡板參數設定所希望使用之板號，agmon 的配置檔含有卡板指令來鑑定系統的每一個卡板。流、時槽及模式欄位用來計算那一個流與時槽被分派到這服務，當模式欄位強制規定有多少時槽被分派，時槽欄位定義所分派時槽的基底時槽。模式的值可有如下數種：
  - ADI_VOICE_INPUT：只接收帶內資訊，這資訊是由 DSP 於所給的時槽所接收。
  - ADI_VOICE_OUTPUT：只傳送帶內資訊，這資訊是由 DSP 於所給的時槽所傳送。
  - ADI_VOICE_DUPLEX：於所給的時槽傳送及接收帶內資訊。
  - ADI_FULL_DUPLEX：包含 ADI_SIGNAL_DUPLEX 及 ADI_VOICE_DUPLEX，這埠傳送及接收帶內媒體及帶外信令。

- 最常用的模式為 ADI_FULL_DUPLEX 及 ADI_VOICE_DUPLEX，ADI_ VOICE _DUPLEX 模式通常配合 NOCC 協定使用，且允許媒體（聲音、傳真）接收及發送。ADI_FULL_DUPLEX 模式通常當於這個埠執行往路協定時使用，這模式允許聲音（帶內）及網路信令傳送及接收。Ctatest 驗證程式能驗證 NCC 及 ADI 服務安裝與操作恰當。

- 關閉服務，消滅脈絡與事件佇列：CT Access 關閉並聯的 CT Access 啟始化函數，***ctaCloseServices*** 關閉於一脈絡內一或多個服務，***ctaCloseServices*** 是非同布的並馬上回應，必須等到完成事件（CTAEVN_CLOSE_SERVICES_DONE）已回到事件佇列後才可關閉所有服務。當多個服務正被關閉且一個以上的服務關閉失敗，CTA_EVENT 結構（ctaWaitEvent）的值欄位將包含第一個關閉失敗服務的原因。***CtaDestroyContext*** 消滅 CT Access 脈絡，所有目前開啟的服務被關閉，這函數是非同布的並馬上回應，必須等到完成事件（CTAEVN_DESTROY_CONTEXT_DONE）已回到事件佇列後才可消滅脈絡及釋放資源。***CtaDestroyQueue*** 消滅事件佇列與和這事件佇列相關聯的所有 CT Access 脈絡，這是同步函數所以應用程式將保留區塊直到所有清除，關閉行動完成，如果服務與脈絡已先關閉，這函數將很快回應。當使用供享脈絡，脈絡實際並沒消滅直到最後使用此脈絡之應用程式呼叫 ***CtaDestroyContext*** 時才真正消滅。

於事件佇列接收事件：一事件可包含資料緩衝器或數量，如果它包含緩衝器，記憶體必須被配置來儲存緩衝器的內容，通常使用下列的方法之一來管理記憶體：

- 應用程式預先配置記憶體，***ctaWaitEvent*** 以資料裝填記憶體，於此狀況應用程式負責配置與釋放記憶體，如 ***adiCollectDigits***

即為此用途。

- 應用程式不預先配置記憶體，記憶體由 CT Access 或 CT Access 的一個服務代為配置記憶體，將 CTA_INTERNAL_BUFFER 位元設入尺寸欄位，於此狀況應用程式負責釋放記憶體，但並不負責配置記憶體。
- 事件程序推荐如下：
- 呼叫 ***ctaWaitrEvent***。
- 檢查 CTA_INTERNAL_BUFFER 位元為"1"，表示成功回應。
- 如果 CTA_INTERNAL_BUFFER 位元為"1"，於事件程序前先將之清除（如此尺寸欄位只定義緩衝器尺寸）。
- 當事件程序完成，如果 CTA_INTERNAL_BUFFER 被初始設為"1"，呼叫 ***ctaFreeBuffer***。

使用等待物件：應用程式使用非童步程式模式可以等待多源事件時不必管制任何特定來源，CT Access 為所有能產生事件的服務內部管理等待物件目錄，CT Access 揭發兩機構來整合應用程式等待物件，CT Access 內部管理等待物件不需額外的執行線。第一機構將控制權給予 CT Access 來等待物件信號及等待應用程式，第二機構允許應用程式等待這些信號且當 CT Access 等待物件被以信號通知時呼叫 CT Access。例如一應用程式可能正等待 Socket 網路事件的到來，或使用者經由鍵盤或滑鼠所產生的輸入及由 CT Access 來的事件，於第一機構 CT Access 允許應用程式以 CT ACcess 註冊網路 Socket、鍵盤及滑鼠物件，如此當任何一個這些事件起來都會呼叫使用者應用程式。第二機構由 CT Access 提供，允許應用程式接受 CT Access 內部等待物件，如此應用程式能就如 CT Access 內部物件一般的使用操作系統特定等待函數來等待網路 Socket、鍵盤及滑鼠物件。CT Access 的等待物件是操作系統特定且當操作系統需求等待函數時被定義成相同的基本形別。於 Win NT，等待物件是與定義於 nmstypes.h 的 MUX_HANDLE 形別

相同，它是如 HANDLE 般相同形別且於 **WaitForSingleObject** 或 **WaitForMultipleObjects** 通用。

等待物件函數：

- **ctaQueryWaitObjects**：如果想得到 CT Access 內部管理等待物件的操作系統特定陣列，執行此函數。
- **ctaRegisterWaitObject**：如果想以 CT Access 註冊等待物件，執行此函數。
- **ctaUnregisterWaitObjects**：如果想解除先前註冊的等待物件，執行此函數。

由 CT Access 管理的等待物件及事件：**ctaRegisterWaitObject** 允許應用程式增加等待物件到它內部管理的等待物件目錄，這函數需要應用程式提供等待物件且不論何時等待物件被操作系統被以信號通知這函數被呼叫，應用程式必須也定義應用程式事件的優先權：

- CTA_PRIORITY_HIGH：CT Access 給予應用程式的等待物件的優先權優於 CT Access 內部等待物件。
- CTA_PRIORITY_NORMAL：CT Access 內部等待物件的優先權優於給予應用程式的等待物件。

以 CT Access 註冊應用程式等待物件後，應用程式呼叫 **ctaWaitEvent**，無論何時當應用程式的等待物件被被以信號通知，CT Access 鎖住佇列並呼叫應用程式所定義的呼叫返回函數來處理事件，這使用者呼叫返回函數可被除了 **ctaWaitEvent** 或 **ctaDestroyQueue** 外的其他任何 CT AccessAPI 函數所呼叫。應用程式也會呼叫 **ctaUnregisterWaitObject** 來未註冊由 CT Access 註冊的應用程式等待物件，當未註冊後所有於此應用程式等待物件上的事件將被 CT Access 忽略。由應用程式管理的等待物件及事件：CT Access 也提供由應用程式管理的等待物件的機構，於啟始化時，應用程式也可定義 CTA_NOTIFY_UPDATE_WAITOBJS 旗

標給 **ctaInitialize**，無論何時 CT Access 改變它的內部等待物件目錄，這旗標會經由事件告訴 CT Access 通知此應用程式。不論何時由 CT Access 管理的等待物件目錄改變時，CTAEVN_UPDATE_WAITOBJS 事件將傳回應用程式。應用程式呼叫 **ctaQueryWaitObjects** 來得到正由 CT Access 內部使用的等待物件目錄，除了 CT Access 等待物件被 **ctaQueryWaitObjects** 所送回，還有應用程式接著於其等待物件上呼叫相依於物件函數的操作系統。無論何時一個 CT Access 等待物件被以信號通知，應用程式應該檢知該信號並以零（0）休息時間（timeout）呼叫 **ctaWaitEvent**，CT Access 處理事件並傳回控制給應用程式，應用程式必須確定當一 CT Access 等待物件被以信號通知後，須能及時地呼叫 **ctaWaitEvent**。

管理參數：修改相關聯的參數能改變 CT Access 服務的相關特性，於 C 語言結構中參數被組織在一起叫作參數結構(parameter structures)，所有的參數結構都含有足夠大部份配置的預設值，參數可被修改來致能或禁能特性且可為特定配置修改服務。CT Access 知道如何為任何服務儲存和取回已被命名的參數及結構值，提供相同函數的所有服務有相同的標準參數，提供相同函數的有服務若配置於不同硬體也許有獨特的延伸參數。除了 CT Access 命名參數，大部份的服務 API 函數允許應用程式傳送一資料結構來覆來廢棄所有參數。

- 參數範圍 CT Access 儲存服務參數兩份備份（整體預設參數及 CT Access 脈絡預設參數），當 CT Access 經由 **ctaInitialize** 被啟始化，服務描述符號及值被建立及儲存，為整體預設參數。當服務於一 CT Access 脈絡被開啟，整體預設參數被複製進入這脈絡、然後 CT Access 脈絡預設參數也可能被修改來定義脈絡特性。當有需要你也可以整體預設參數值來回復 CT Access 脈絡參數。

- 整體預設參數：當於 **ctaInitialize** 定義，CT Access 使用本地記憶體及系統共享記憶體來儲存整體預設參數，如果預設參數在使用系統共享記憶體程序中被共享，那麼 ctdamon 必須執行，於此情況參數值能被配置檔所修改。以包含改變預設值的檔名啟動 Ctdamon，Ctdamon 為參數建立共享記憶體並且修改預設值，接著如整體預設般的使用共享記憶體啟動應用程式。當於伺服模式，參數永遠儲存在共享記憶體。如使用應用程式本地記憶體，於應用程式經由 ctdamon 來改變參數值是無效的，必須使用 CT Access 參數通路函數（如 **ctaSetParmByName**）來修改值。
- 參數函數：如果一個有效的 CT Access 脈絡處理（**ctahd**）被傳遞給函數，這脈落參數的備份被檢知或修改，如果脈絡傳遞 NULL_CTAHD，則整體預設參數被檢知或修改。參數函數如下：
  - **ctaSetParmByName**：用來修改所給參數名的單一欄位。
  - **ctaGetParmByName**：用來檢知所給參數名的單一欄位。
  - **ctaGetParmInfo**：用來檢知參數欄位定義。
  - **ctaGetParm**：用來為一已給的參數結構傳回參數值。
  - **ctaRefreshParm**:用來重置所有於 CT Access 脈絡的脈絡參數值為整體預設值。
  - **ctaGetParmID**：用來檢知所給參數名的參數 ID。
  - **ctaLoadParameterFile**：用來載入及修改預設參數值。
  - CT Access 參數函數允許由定義參數 ID(如 **ctaGetParmID**）或定義參數名（如 **ctaGetParmByName**）來得到參數訊息，這兩種選項各有其優缺點：
  - 由定義參數 ID 方式：優點為檢知程序較快，缺點為記憶 ID 比記憶名稱困難。
  - 由定義參數名方式：優點為記憶名稱比記憶 ID 容易，缺點

為檢知程序較慢。

- Ctaparm 公用程式用來顯示系統預設參數，ctdaemon 允許修改系統整體預設參數且設定整體追蹤符號。

- 參數管理服務 PRM（The Parameter Management Service）：參數管理服務是 CT Access 的一種服務，能使應用程式發展者以操作其它 CT Access 服務參數的方法來操縱應用程式定義的參數，應用程式參數被定義為 ASCII 本文檔，也能使用 pfencode 公用程式將之轉換成二進位參數檔，這本文或二進位（被編碼）參數檔被儲存在被 CTA_DPATH 環境變數定義的目錄表列內的目錄裏，於服務目錄中定義 PRM 服務所載入的參數傳遞給 **ctaInnitialize**。

處理 CT Access 錯誤：所有的 CT Access 函數傳回狀態碼，一 SUCCESS（0）傳回碼表示函數完成（於同步函數）或函數已啟始化（於非同步函數）。如果傳回碼不是 SUCCESS（0），表示是一錯誤碼且函數執行失敗，於非同步函數傳回非零值且未完成啟始化，沒有隨後而來的是件被產生。如一非同步函數於啟始化後才執行失敗，CT Access 將投遞一 DONE 事件給應用程式並且事件值欄位包含一錯誤碼。錯誤碼由每個服務所定義，所有提供相同函數的服務提供一致的錯誤碼，ctaGetText 用來取得錯誤碼的本文表示。CT Access 系統錯誤碼被定義在 ctaerr.h 表頭檔且以 CTAERR_為字首，服務錯誤碼被定義在相關的表頭檔（如於 VCE（Voice Message）服務錯誤碼定義在 vcedef.h）。

- 設定錯誤處理：**ctaSEtErrorHandler** 是用來定義一如果錯誤發生時之行動，當執行任何 CT Access 函數遭遇錯誤，CT Access 在傳回應用程式前會先執行錯誤處理行動，只有當 CT Access 被求助時才會答應執行錯誤處理行動，如果一錯誤是經由事件傳回時則不會執行錯誤處理行動。於預設時並沒有定義錯誤處理行動，可定義如下的錯誤處理行動。

- 列印錯誤訊息到標準輸出設備且反回應用程式。
- 列印錯誤訊息並跳出 CT Access。
- 呼叫一使用者定義的錯誤處理。

- 錯誤追蹤：於呼叫 **ctaInitialize** 時追蹤被致能或禁能，服務及應用程式都能記錄追蹤訊息。呼叫 **ctaSetTRaceLevel** 來致能服務追蹤，於呼叫 **ctaSetTRaceLevel**，設定所要追蹤（如事件、命令等）的旗標，依據所設定的旗標，服務將紀錄內部追蹤訊息到 ctdaemon。**CtaLogTrace** 被應用程式用來記錄應用程式追蹤訊息，一追蹤艱難的狀況及格式隨著追蹤值被傳送。如果追蹤被致能，ctdaemon 一定要執行，執行 ctdaemon 可以定義傳送追蹤訊息給檔案、給交談式的控制台或給遠端的顯示器。
- 錯誤的反向追蹤：CT Access 提供一追蹤選項，允許於錯誤發生的脈絡上接收追蹤訊息的歷史紀錄。如何想致能錯誤反向追蹤，修改傳給 **ctaInitialize** 的 CTA_INIT_PARMS 結構內的追蹤旗標欄位。如果想回顧導致錯誤發生的命令及事件資料，致能反向追蹤，如果想觀察不論有無錯誤的所有 CT Access 脈絡追蹤訊息，不要致能反向追蹤。如果想將一特定脈絡所有最近紀錄的追蹤訊息寫入 CT Access 守護神（daemon），呼叫 **ctaLogTrace**，設定所要分析的脈絡，並且設定追縱艱難的話狀態（***tracesevrity***）引數到 CTA_TRACE_SEVERITY_ERROR。

相容性驗證：由於 CT Access 及被啟始化的 CT Access 服務是動態連結的，如果正使用一相容的程式館，應用程式需要一個方法來確定，當任何包括需要應用程式重組譯的修改發表將增加相容的水準。如下函數：

- **ctaGetVersion**：用來檢查安裝的 CT Access 版本。
- **ctaGetServiceVersion**：用來檢查安裝的服務版本。

CtaGetVersion 及 **ctaGetServiceVersion** 傳回版本、建立日期及 CT Access 及 CT Access 服務板本的相容水準。Ctavers

公用程式顯示 CT Access 及 CT Access 服務板本，這版本訊息可用於報告給 Natural MicroSystem 公司，所使用的是那個版本及建立日期。

使用 CT Access 伺服模式：伺服模式致能於 CT Access 環境多客戶應用程式共享相同電話技術資源，當於伺服模式操作，應用程式可共享脈絡（及服務物件），存取相同開啟服務要求，及從一應用程式放手脈絡控制將控制轉給下一個。CT Access 伺服（ctdaemon）代表客戶應用程式建立脈絡及服務物件，於這些脈絡可用的資源上應用程式使用脈絡處理來執行特定命令，CT Access 伺服（ctdaemon）以獨特物件描述符號對應這些應用程式特定脈絡處理，如此允許應用程式們於同時間使用相同的脈絡（或服務物件）。當於伺服模式操作，應用程式透過 CT Access API 履行工作，CT Access 伺服（ctdaemon）的行動，實際上對客戶應用程式而言是看不不到的，當應用程式希望共享已存在的脈絡，將得到脈絡的物件描述符號且連接上相同脈絡。

工作於 CT Access 伺服模式須：

- 查證於 CTA.CFG 檔內的設定。
- 執行 ctdaemon 來致能伺服模式函數。
- 於伺服模式開始應用程式。

查證 CT Access 配置檔：執行伺服模式，為了使系統包含適當的訊息，須編輯 CT Access 配置檔（CTA.CFG），確定被編輯檔出現在 application 區域的如下陳述須為：CTAmode=1（預設時 CTAmode=0）這個陳述告訴應用程式除非於經由 **ctaInitialize** 明確定義它的操作模式，否則將於伺服模式執行。確定被編輯檔出現在 server 區域的如下陳述須為：TraceMode=1，這個陳述致能伺服的完全追蹤。StartCtaSErver=1，這個陳述致能伺服模式函數。

開始 CT Access 伺服（ctdaemon）：使用下列一種方式來啟動 CT Access 伺服（ctdaemon）。

- 於命令 prompt 下求助於 ctdaemon-i（Windows NT）：這方式允許 ctdaemon 與控制台完全交談。
- 求助於 ctdaemon-I 來安裝 ctdaemon 成為一 Windows NT 服務，然後從控制台服務 applet 開始 NMS CT 守護神服務。
- 為了控制台與 NMS ctdaemon Windows NT 服務交談，當正在執行 NMS CT 守護神服務時可於命令 prompt 下求助於 ctdaemon-c。

於伺服模式啟始 CT Access 應用程式：CT Access 應用程式開始時必須啟始化，使用下列的方式之一來於伺服模式啟始 CT Access 應用程式。

- 求助於 **ctaInitialize** 並定義 NULL 為 ***initparms*** 為引數，當定義 NULL 為 ***initparms*** 為引數時，CT Access 啟始化時會依照編輯於 CTA.CFG 檔案內的特定訊息。
- 求助於 **ctaInitialize** 並傳送 CTA_MODE_SERVER 到 CTA_INIT_PARMS 結構的 ctaflags 欄位，這樣將定義應用程式於伺服模式執行。註：應用程式於伺服模式執行，必須於呼叫任何其他 CT Access 函數前先執行 **ctaInitialize**。

使用共享脈絡：ctdaemon 的行為就如供 CT Access 脈絡使用的伺服且能使多個客戶應用程式使用相同的程序資源，例如兩個客戶應用程式共享兩分離的 CT Access 脈絡，SWI 服務於脈絡 X 被開啟，ADI 及 VCE 服務於脈絡 Y 被開啟，兩個客戶應用程式都能使用於 X、Y 脈絡上開啟的服務，但每個使用者使用不同的 CT Access 脈絡處理來參照這些脈絡，ctdaemon 以共同的 CT Access 脈絡來關聯這些客戶 ***ctahd***s。要建立及共享 CT Access 脈絡，客戶應用程式必須建立 CT Access 脈絡並連接 CT Access 脈絡。

建立共享脈絡：應用程式以 **ctaCreateContextEx** 來建立共享脈絡，**ctaCreateContextEx** 需要和 **ctaCreateContext** 一樣相同的引數及新增設定伺服共享模式的旗標，伺服共享模式決定客戶

程序如何連接共同物件能使用脈絡上開啟的服務要求。當客戶應用程式以 **ctaDestroyContext** 消滅一共享脈絡，這應用程式放棄這脈絡資源的控制，無論如何等到每一使用這脈絡資源的應用程式消滅這脈絡後，CT Access 才會消滅這脈絡。應用程式也可以 **ctaCreateContext** 來建立共享脈絡，無論如何以 **ctaCreateContext** 來建立共享脈絡必須總是建立於共同存取模式下。

- 設定服務共享模式：當兩個客戶應用程式共享一 CT Access 脈絡，它們兩個都能使用於這脈絡上開啟的服務要求，無論如何，相依於共享模式，脈絡被建立，客戶程序可能或可能不能馬上使用函數，及從開啟的服務接收事件。CT Access 提供兩服務共享模式：
  - Common access（共同存取）（預設）：於共同存取模式，被一個客戶應用程式於脈絡上開啟的服務要求能立刻被其他連接到此脈絡之客戶應用程式所使用，應用程式不需要使用 **ctaOPenSErvices** 來開啟早已在脈絡上作用的服務要求，除此之外，任何應用程式以 **ctaCloseService** 於脈絡上關閉一服務，將會關閉共享此脈絡之所有應用程式的服務。例如兩應用程式共享於共同存取模式建立的 CT Access 脈絡，應用程式一建立這脈絡且於此脈絡上開啟 ADI 及 VCE 服務，當應用程式二連接這脈絡，它馬上可存取此脈絡上開啟的 ADI 及 VCE 服務要求。
  - Declared Access（公告存取）：於公告存取模式，當需要時每個應用程式開啟及關閉它自有的服務要求，當應用程式連接此脈絡，沒有服務可用直到這應用程式開啟它們，甚至共享此脈絡的其它應用程式已有開啟的服務要求也無法供此應用程序使用，除此之外，此應用程式關閉一特定服務要求，將只會關閉針對此應用程式的服務，共享此脈絡已有開啟的服務要求的其它應用程式仍能繼續使用它。例

如兩應用程式共享於公告存取模式建立的 CT Access 脈絡，應用程式一建立這脈絡且於此脈絡上開啟 ADI 及 VCE 服務，當應用程式二連接上這脈絡，它無法使用著些服務直到它於脈絡上明確的開啟它們。

比較共同存取及公告存取模式：下表描述此二模式相同相異之處。

共同存取模式	公告存取模式
共享服務單一要求	
事件被廣播給所有客戶，由客戶固守共享脈絡的處理	
於伺服模式只有最後一個客戶求助於 ctaDestroyContext 時 CT Access 脈絡才被消滅	
任何應用程式以 **ctaCloseService** 於脈絡上關閉一服務，將會關閉共享此脈絡之所有應用程式的服務	此應用程式關閉一特定服務要求，將只會關閉針對此應用程式的服務，共享此脈絡已有開啟的服務要求的其它應用程式仍能繼續使用它
被一個客戶應用程式於脈絡上開啟的服務要求能立刻被其他連接到此脈絡之客戶應用程式所使用	當應用程式連接此脈絡，沒有服務可用直到這應用程式開啟它們
從所有繼承非隱藏服務而來的事件是看得見的	只有從被明確開啟服務而來的事件是看得見的

連接共享脈絡：當一 CT Access 應用程式建立一脈絡，CT Access 為這脈絡傳回一脈絡處理（***ctahd***），脈絡處理是由應用程式所設定的，並不能傳送給其它希望使用此脈絡之應用程式使用，當 CT Access 執行於伺服模式，應用程式使用脈絡物件描述

符號傳送有關 CT Access 訊息給其它應用程式，這物件描述符號包含所有獨一無二識別一 CT Access 脈絡所需訊息，應用程式可於建立脈絡的同時設定描述符號。或由服務來產生描述符號，當應應用程式得到特定脈絡之描述符號後，它可將描述符號傳給其他應用程式（由應用程式決定傳送及方法），其他應用程式然後可用此描述符號來連接脈絡。註：脈絡名稱是物件描述符號的捷徑，於呼叫 **ctaCreateContext**、**ctaCreateContextEx** 及 **ctaAttachContext** 時，脈絡名稱可用作描述符號。

得到服務物件描述符號：如下程序描述如何得到服務物件描述符號。

- 建立脈絡，記錄由呼叫 **ctaCreateContextEx** 傳回的脈絡處理。
- 於呼叫 **ctaCreateContextEx** 使用脈絡名稱，或求助於 **ctaGetObjDescriptor** 來設定由 **ctaCreateContextEx** 傳回的 **ctahd** 方式來取回脈絡物件描述符號。
- 以下列三種方法之一傳送描述符號給其他應用程式：
  - **ctaQueueEvent** 到一眾所周知被命名的脈絡。
  - 應用程式間私人的 IPC 通道。
  - 當它們共享已存在之脈絡時，儲存脈絡名稱到被其他應用程式用作物件描述符號的檔案上。

連接 CT Access 脈絡：以 **ctaAttachContext** 連接已存在的 CT Access 脈絡，當求助於 **ctaAttachContext**，設定由 **ctaCreateQueue** 傳回的 ***ctagueuehd***，及一個 CT Access 物件描述符號，**ctaAttachContext** 傳回由應用程式定義的 **ctahd**。下列程序為一應用程式建立脈絡，第二應用程式連接這 CT Access 脈絡：

- 第一應用程式呼叫 **ctaCreateContextEx**（或 **ctaCreateContext**）建立脈絡，然後應用程式求助於 **ctaGetObjDescriptor**（配合 ***ctahd***）取回由脈絡定義的物件描述符號。如果應用程式呼叫 **ctaCreateContextEx** 建立脈絡時同時設立描述符號，就不必呼

叫 **ctaGetObjDescriptor** 獲得描述符號。

- 第一應用程式傳送物件描述符號給第二應用程式（參考 **ctaQueueEvent**）。
- 第二應用程式求助 **ctaAttachContext** 及提供的物件描述符號及 CT Access 事件佇列處理來連接上脈絡，**ctaAttachContext** 一由應用程式定義的 ***ctahd***。建立脈絡的應用程式可求助 **ctaDestropContext** 來放棄脈絡控制，如此做，應用程式將放手脈絡的控制將之交給其他連接到此脈絡的應用程式，直到所有應用程式都放棄脈絡控制，脈絡才被消滅。如果應用程式呼叫 **ctaCreateContextEx** 建立脈絡時同時設立脈絡名稱，脈絡名稱可被其它應用程式用作描述符號來連接到這建立的脈絡。

於共享脈絡上開啟服務：應用程式以 ctaOpenService 在共享脈絡上開啟服務，任何連接上一共享脈絡的客戶應用程式能於此共享脈絡上開啟服務（並不止於建立它的應用程式），無論如何，是否共享脈絡的應用程式都能依賴設定於 **ctaCreateContextEx** 的服務共享模式來立刻使用服務要求。隨之而來於公告模式連接共享脈絡的應用程式呼叫 **ctaOpenService** 是不需傳送服務特定訊息到這呼叫，例如當脈絡建立時，ADI 服務需要卡板、流、及時槽的訊息，隨之而來連接脈絡的應用程式並不需要傳送訊息，如果已有訊息在場將被忽略。註：共享脈絡模式只有當呼叫 **ctaCreateContextEx** 來建立脈絡時能設定，接著共此脈絡的客戶程序不能改變這共享模式。

為共享脈絡處理事件：共享一 CT Access 脈絡，每一客戶應用程式必須建立至少一個事件佇列，當連接到一存在的脈絡，應用程式指定事件佇列處理（***ctaqueuehd***）。任何連接到脈絡的應用程式立刻收到此脈絡上所有可用服務的事件（不論應用程式是否使用服務上的函數）。例如有應用程式一與二，共享含有 VCE、NFX、ADI 三種服務的脈絡，應用程式一需要三者的服務，應用

程式二只需要 NFX 的服務，但當它們連接上脈絡時都會收到可提供 VCE、NFX、ADI 三種服務的事件。註：當應用程式連接建立於公告模式的脈絡時，應用程式只接到明確開啟的服務的事件。

指定接收那些事件：**ctaSetEventSources** 指定提供應用程式接收所需事件的服務，當應用程式連接建立於共同存取模式的脈絡時，能求助 **ctaSetEventSources** 來限制所接收事件的形別（type）。如上例的情況改為共同存取模式，最初應用程式二收到可提供 VCE、NFX、ADI 三種服務的事件，但由求助於 **ctaSetEventSources**（配合適當的 ***ctahd***）及指定 NFX 服務，應用程式將有效的遮蔽除 NFX 服務外所有的服務。於共同存取模式，在求助 **ctaAttachContext** 後到呼叫 **ctaSetEventSources** 前的期間會收到不需要的服務的事件，你必須編碼你的事件處理以處理不想要的事件。

管理共享服務參數：當一個應用程式為共享服務要求調整參數值，它改變使用這服務的應用程式的這些參數，因此，於共享脈絡上共服務要求使用的參數本質上是總體性質的。當客戶應用程式連接到共享脈絡，它們可使用 **ctaGetParmInfo** 來取回於脈絡上指定給服務要求的參數。

監視伺服狀態：當應用程式收到 CTAERR_SVR_COMM 錯誤，這指出伺服失敗並回應，於伺服模式期間任何發行命令都能傳回 CTAERR_SVR_COMM。應用程式於不活動的期間監視伺服連續狀態可求助於 **ctaWaitEvent**，這樣允許應用程式於某些特定的線（Threads）以休眠狀態來監視伺服的狀態。

發展環境：用於發展應用程式的頭檔（header files）表列如下：

- ctadef.h：CT Access 描述符號與參數應用程式介面
- ctaerr.h：CT Access 錯誤定義
- nmstypes.h：提昇便於攜帶的基本形別定義

- vcedef.h：聲音訊息服務應用程式介面
- swidef.h：交換服務應用程式介面
- nccdef.h：Natural 呼叫控制服務應用程式介面
- adidef.h：AG 裝置服務應用程式介面
- ppxdef.h：點對點交換服務應用程式介面
- dtmdef.h：數位通信波道監視服務應用程式介面

程式館檔案：

- cta.lib：應用程式連結程式模式或伺服模式的 CT Access 程式館。
- swimgr.lib：交換服務程式館。
- vcemgr.lib：聲音訊息服務程式館。
- ppxmgr.lib：點對點交換服務程式館。
- adiapi.lib：AG 裝置服務程式館。
- adidtm.lib：數位通信波道監視服務程式館。

如何於 Windows NT 環境下配置環境變數及如何使用 CT Access 編譯應用程式：這軟體是安裝在 nms 目錄下

- 編譯：當編譯一應用程式，告訴編譯程式 CT Access 頭檔放在那個目錄下，預設頭檔放在\nms\include 目錄下，定義 WIN32 於編譯命令線上，如-Ic:\nms\include –DWIN32，你可選擇增加 \nms\include 到你的包含環境變數
- 應用程式連結：當於 Windows NT 環境連結，指定 CT Access 程式館（如 cta.lib、vcemgr.lib 等）所存放的路徑，可明確的於命令線上指定，或選擇增加\nms\lib 到程式館環境變數。當執行執行 CT Access 應用程式，指定服務管理動態管理連結程式館（DLL）所在的路徑，經由控制台系統對話盒增加\nms\bin 到路徑環境變數。註：CT Access 已經使用 Microsoft's Visual C++6.0 來編譯與連結，LINK50COMPAT 旗標已經用來確定與由 Microsoft's Visual C++5.0 建立的應用程式的相容。

- 範例：編譯與連結範例參照示範程式目錄，NMS 提供 nmake 的編譯檔。
- 二進位碼的公用程式：二進位碼的公用程式被儲存於\nms\bin\ 目錄，二進位碼的示範程式是在它們各自的目錄。
- 配置檔：\nms\ctaccess\cfg 目錄包含一配合 ctavers 公用程式及 ctdaemon 使用的樣版配置檔（cta.cfg）。
- 提示/聲音檔：\nms\ctaccess\prompts 目錄包含被示範程式使用的提示表（prompt tables）及聲音檔，當執行 CT Access 應用程式，聲音檔可在 CTA_DPATH 環境變數指定的路徑找到。

函數摘要：建立應用程式的相關函數：

函數	非同步/同步	說明
ctaInitialize	同步	為 CT Aaccess 應用程式建立可用服務目錄
ctaCreateQueue	同步	建立一 CT Access 事件佇列並傳回佇列處理(*ctaqueuehandle*).
ctaCreateContext	同步	製造一 CT Access 脈絡並傳回脈絡處理(*ctahd*).
ctaCreateContextEx	同步	製造一脈絡並傳 *ctahd*, 及安排脈絡共享模式(僅用於伺服模式).
ctaOpenServices	非同步	於特定脈絡上開啟一或多個服務(*ctahd*).
ctaAttachContext	同步	致能應用程式使用其他應用程

函數	非同步/同步	說明
		式所開啟的脈絡 (僅用於伺服模式).
ctaGetObjDescriptor	同步	傳回與特定 CT Access 脈絡或服務物件關聯的物件描述符號 .

控制事件佇列程序的相關函數

函數	非同步/同步	說明
ctaQueueEvent	同步	使成為排隊一事件到事件佇列
ctaWaitEvent	同步	由特定佇列取回一事件
ctaSetEventSources	同步	指定事件佇列可以收到事件的服務 (僅用於伺服模式).
ctaGetEventSources	同步	取回應用程式可以取回事件的服務目錄(僅用於伺服模式).
ctaAllocBuffer	同步	配置經由**ctaQueueEvent**資料傳輸的記憶體
ctaFreeBuffer	同步	釋放內部配置的緩衝器或經由**ctaAllocBuffer**配置的緩衝器

關閉服務、消滅脈絡及佇列，釋放資源的相關函數

函數	非同步/同步	說明
ctaCloseServices	非同步	於一脈絡上關閉一或多個服務
ctaDestroyContext	非同步	消滅脈絡及關閉所有現在開啟的服務.
ctaDestroyQueue	同步	消滅所有在這事件佇列建立的脈絡，關閉所有現在開啟的服務及消滅事件佇列

修改參數的相關函數

函數	非同步/同步	說明
ctaGetParmByName	同步	由所給的參數名稱取回單一欄位
ctaSetParmByName	同步	由所給的參數名稱修改單一欄位
ctaGetParmID	同步	由所給的參數名稱取回參數分類 ID
ctaGetParmInfo	同步	由所給的參數結構傳回參數值
ctaGetParms	同步	由所給的參數結構傳回參數值
ctaLoadParameterFile	同步	載入及修改參數預設值
ctaRefreshParms	同步	重置一 CTA 脈絡上所有脈絡參數值為總體遇設值

使用等待物件的相關函數

函數	非同步/同步	說明
ctaQueryWaitObjects	同步	得到由 CT Access 內部管理的操作系統指定等待物件陣列，當應用程式管理等待物件時使用此函數
ctaRegisterWaitObject	同步	以 CT Access 註冊一等待物件當 CT Access 管理等待物件時使用此函數
ctaUnregisterWaitObject	同步	反註冊先前註冊的等待物件。 當 CT Access 管理等待物件時使用此函數

處理錯誤的相關函數

函數	非同步/同步	說明
ctaGetText	同步	傳回一錯誤的本文陳述，原因，事件或命令碼
ctaSetErrorHandler	同步	定義當一錯誤從一 CT Access 服務傳回時的採行行動.

驗證相容的相關函數

函數	非同步/同步	說明
ctaGetVersion	同步	取回所安裝的 CT Access 版本
ctaGetServiceVersion	同步	取回初置的服務版本.

追蹤的相關函數

函數	非同步/同步	說明
ctaLogTrace	同步	允許記錄錯誤、警告、或情報的訊息到 CT Aaccess 追蹤紀錄
ctaSetTraceLevel	同步	為一特定 CTA 脈絡上的一服務設定追蹤符號

實行混合工作的相關函數

函數	非同步/同步	說明
ctaFindFile	同步	尋找一使用這特定預設延長及環境變數
ctaFormatEvent	同步	為列印診斷用，格式化一事件進入一字串
ctaGetQueueHandle	同步	得到特定 CTA 脈絡的佇列處理

CT Access 的公用程式：

- ctaparm：用於顯示儲存於共享記憶體由 ctdaemon 所選的參數定義與值。它讀取儲存於共享記憶體的參數然後顯示於螢幕上，指定於命令線上取回的參數。如果 ctdaemon 已執行，ctaparm 顯示這些儲存於共享記憶體的參數，若尚未執行 ctdaemon，ctaparm 顯示於編譯時的預設參數。如果 ctaparm 顯示 CTAERR_ALREADY_INITIALIZED 錯誤，它可能使用比 ctdaemon 不同的配置檔，這被 ctaparm 使用的檔於【ctasys】區間內表列服務，並非由 ctdaemon 所載入。Ctaparm 與 ctdaemon 預設載入\nms\ctaccess\cfg\cta.cfg。
  - 用法：ctaparm【選項】：選項如下：
    - –e：顯示所有參數結構的名稱
    - –s 結構名稱：顯示所有在結構目錄內的參數
    - –f 欄位名稱：顯示所有與表列欄位名稱相配的參數
    - -g：由現有的值產產生一參數檔
    - –1：配置檔：指定定義服務啟始用的配置檔
  - 例如：執行 ctaparm 可於提示輸入 ctaparm [-e –s structlist –f fieldlist –g]，顯示這參數結構的名稱使用-e 選項，顯示於螢幕的參數結構名稱然後配合-f 選項來顯示所給參數結構的所有參數。由-s 及-f 選項所產生的名單能以+（plus）來定界，如：
    - ctaparm –s adi.play+adi.record
    - ctaparm –f gain+dtmfabort
- pfencode：為用於 PRM 服務的參數檔。用來轉譯參數管理本文參數檔為二進位參數檔。Pfencode 轉譯參數檔用來勸阻終端使用者的修改。本文及二進位(被編譯)參數檔都能有.pf 的延伸，如果沒有指定不同的檔名給被編譯輸出檔，這原始的本文參數檔將改名為具.tpf 延伸的檔名。不論本文或被編譯的二進位參

數檔都必須包含 prm 字首及.pf 延伸才能被 PRM 服務所載入。儲存原始的本文參數檔，因為 pfencode 無法反編譯一編譯的參數檔來再生這原始本文參數檔。

- 用法：pfencode【選項】 檔名：選項如下：
  - –o 檔名：指定輸出（被編譯）檔名，這預設的檔名為這被指定的本文參數檔加上.pf 的延伸，如果輸出(被編譯)檔名與所指定的本文參數檔名相牴觸，這原來的本文參數檔的延伸被改為.tpf。
- 例如：執行 pfencode prmMyFile.pf，pfencode 將更名原始本文參數檔為 prmMyFile.tpf，及產生一被編譯的二進位參數檔 prmMyFile.pf。執行 pfencode –o prmData.pf prmMyFile.pf，將產生一含有指定於 prmMyFile.pf 檔的參數訊息的被編譯的二進位參數檔 prmData.pf。

- ctavers：用來查證 CT Access 安裝及顯示 CT Access 及它的成員的版本訊息。
  - 用法：ctavers【選項】，選項如下：
    - -s：報告服務及正被 ctraemon 使用的服務管理的版本，如果沒有 ctraemon，ctavers 將於程式模式執行並報告包含在預設 CT Access 配置檔內的服務及服務管理版本。Ctavers 配合指定於 CT Access 配置檔內的服務及服務管理來呼叫 **ctaInitialize**，然後它取回並顯示所有當 CT Access 被啟始化被註冊的服務及服務管理版本訊息，也顯示 CT Access 派遣員（cta.dll）的版本訊息。於命令線或於 CT Access 資料夾的代表此功能的圖像呼叫 ctavers.exe，確定 CTA_DPATH 包含可以找到 cta.cfg 檔的目錄，預設的檔案是放置在\nms\ctaccess\cfg 目錄下。
    - –x xxx,xxxmgr：報告被指定的服務（xxx）及服務管理（xxxmgr）版本。

- –l cfgfile：報告包含在 CT Access 配置檔內的服務及服務管理版本。

- ctdaemon：CT Access 的守護神。用於致能客戶應用程式於伺服模式執行。Ctadaemon 代表應用程式建立及管理程序脈絡，允許應用程式共享及放手伺服資源。允許應用程式發展者修改系統總體預設參數，配置總體追蹤標誌，經由 java appelet 實行遠端追蹤，可選擇記錄追蹤訊息到一檔案。Ctdaemon 讀取指定於命令線的配置檔或於 CTA_DPATH 發現的 cta.cfg（於安裝時指定的）。Cat.cfg 檔的【ctasys】區間指定被 ctdaemon 用來要傳給呼叫來 **ctaInitialize**（啟始化）的服務及服務管理。當於 CT Access 執行 **ctaInitialize** 完成註冊服務，這編譯時的參數值於共享記憶體內被取代並且能被守護神（daemon）修改。於啟始化服務後，ctdaemon 讀取配置檔【ctapar】區間並將新的參數值寫入共享記憶體。任何使用系統總體預設參數的程序將從共享記憶體取回預設參數值來取代由程序資料區域取回編譯時程度總體預設值。Ctdaemon 的配置檔必須指定所有用於期望使用系統總體預設值的應用程式的 CT Access 服務。例如，cta.cfg 檔必須修改（如下所示）來允許 Natural Text 應用程式使用系統總體預設參數：Service = tts , adimgr。應用程式必須配置傳給 Initialize 的 CTA_INIT_PARMS 結構的參數旗標（parmflags）欄位為 CTA_PARM_MGMT_SHARED 來指定它們希望使用由 ctdaemon 管理的系統總體預設參數，當此位元被選定但 ctdaemon 未執行，將傳回 CTAERR_SHAREMEM_ACCESS。如果應用程式希望總體預設參數儲存於共享系統記憶體但又不希望沒執行 ctdaemon 時會傳回錯誤，配置參數旗標（parmflags）到 CTA_OARM_MGMT_SHARED_IF_PRESENT，當這個配置被選擇並且 ctdaemon 未執行，這預設參數管理就地到這程序。

當 ctdaemon 於視窗命令開始執行，它能用來取回及修改總體預設參數值。例如：輸入 s parmname = value 或 g parmname 來修改（調整）或取回（穫得）現在的預設值；於交談命令提示輸入?來顯示可用的命令；使用 ctaparm 公佣程式產生能用於配置檔【ctapar】區間內的參數表列。Ctdaemon 也為追蹤控制一共享記憶體片段。從應用程式致能追蹤，必須於 ctdaemon 正在執行且傳給 ctaInitialize 的 CTA_INIT_PARMS 結構的參數旗標（parmflags）欄位必須設為 CTA_TRACE_ENABLE。於 daemon 命令線輸入 T，開始顯示追蹤訊息。只會顯示追蹤標誌與被致能的指定訊息形別相符合的追蹤訊息。由 daemon 命令線使用 m 追蹤標誌命令，daemon 能配置總體追蹤標誌，這樣為已經由 **ctaInitialize** 致能追蹤的所有程序配置追蹤標誌。於追蹤有更多的選擇，可於應用程式本身使得 **ctaSetTraceLevel**，來於一指定的程序內的一指定的 CT Access 脈絡配置追蹤標誌。追蹤標誌可以被指定為一序列字串間以+（plus）連接起來（沒有空隙），每個符合一追蹤標誌值的字串能於 **ctaSetTraceLevel** 內被指定。

- 用法：ctdaemon【選項】，選項如下：
  - –f filename：指定包含服務及服務管理敘述的 ASCII 檔並且不理會預設參數，通常為 cta.cfg 檔。
  - –I：當 WindowsNT 服務呼吸 NMS CT Daemon 時安裝 ctdaemon。
  - –U：反安裝 NMS ctdaemon 的 Windows NT 服務。
  - –c：允許控制台與 NMS ctdaemon 的 Windows NT 服務交談。於 NMS ctdaemon 服務正執行的期間於任何命令提示求助於 ctdaemon -c。
  - –i：當執行與控制台交談的應用程式執行 ctdaemon 時使用此選項，如果沒有選項被選擇，這是 ctdaemon 的預

設選項。

◆ -?/-h：求助用。

交談命令提示輸入 D 可看見為配置追蹤標誌的識別明稱如下表：

字串	追蹤標誌值
drvcmd	CTA_TRACEMASK_DRIVER_COMMANDS
drvevt	CTA_TRACEMASK_DRIVER_EVENTS
dispcmd	CTA_TRACEMASK_DISP_COMMANDS
dispevt	CTA_TRACEMASK_DISP_EVENTS
apicmd	CTA_TRACEMASK_API_COMMANDS
apievt	CTA_TRACEMASK_API_EVENTS
apierr	CTA_TRACEMASK_API_ERRORS
svcerr	CTA_TRACEMASK_SVC_ERRORS
dbgbit0	CTA_TRACEMASK_DEBUG_BIT0
dbgbit1	CTA_TRACEMASK_DEBUG_BIT1
dbgbit2	CTA_TRACEMASK_DEBUG_BIT2
dbgbit3	CTA_TRACEMASK_DEBUG_BIT3
allcmd	CTA_TRACEMASK_ALL_COMMANDS
allevt	CTA_TRACEMASK_ALL_EVENTS
all	CTA_TRACEMASK_ALL
none	0

有三種方法可以觀看由 ctdaemon 使可用的追蹤訊息，為控制台、遠端公用程式及記錄檔。除了於本地控制台追蹤訊息的標準顯示，也可使用 Java 1.1-capable web 瀏覽器於遠端位置觀看。

Ctdaemon 也可將追蹤訊息寫入一檔案，於 cta.cfg 內的 TraceFile 關鍵字指定追蹤檔的名稱，每次 ctdaemon 開始執行，所追蹤訊息將附加於記錄檔後。當需要時可由使用者決定刪除或儲存這個檔案。

錯誤摘要：

十六進位碼	十進位碼	錯誤名稱	敘述
0x1	1	CTAERR_RESERVED	
0x2	2	CTAERR_FATAL	問題：於 CT Access 發生內部錯誤。解答：與 NMS 開發者連絡。
0x3	3	CTAERR_BOARD_ERROR	於卡板上發生不預期的錯誤，在多數的情況也有一 CTAEVN_BOARD_ERROR 事件發生，那事件將包含卡板的錯誤碼。
0x4	4	CTAERR_INVALID_CTAQUEUEHD	一無效的 CT Access 佇列處理被當作引數傳給函數，或這佇列被其他線消滅。
0x5	5	CTAERR_INVALID_CTAHD	一無效的 CT Access 脈絡處理被當作引數傳給函數，或這脈絡被其他線消滅。
0x6	6	CTAERR_OUT_OF_MEMORY	問題：無法配置記憶體給佇列、驅動程式或 CTA 脈絡、錄或放緩衝器、或暫存用。解答：當錯誤發生在一

十六進位碼	十進位碼	錯誤名稱	敘述
			DONE 事件，可能是卡板上記憶體不足。
0x7	7	CTAERR_BAD_ARGUMENT	問題：一函數引數有一無效值或一所需的指標引數為 NULL。解答：檢查所有引數為有效形別及範圍
0x8	8	CTAERR_OUT_OF_RESOURCES	問題：於內部表不能找到自由的空間。解答：驗證當不使用時關掉處理。
0x9	9	CTAERR_NOT_IMPLEMENTED	這函數沒有配置。
0xA	10	CTAERR_NOT_FOUND	指定的參數並不存在，服務或服務管理沒被啟始化，沒有連接佇列，或沒於 CT Access 脈絡上開啟，於伺服模式應用程式要求一並未於 cta.cfg 指定的服務。
0xB	11	CTAERR_BAD_SIZE	問題：一尺寸引數太小以致無法接收一資料結構，或一錄或放的緩衝器為指定的編碼碼框尺寸不是倍數。解答：檢查所有引數為有效尺寸。
0xC	12	CTAERR_INVALID_ST	於現有的敘述這函數是無效

十六進位碼	十進位碼	錯誤名稱	敘述
		ATE	的。
0xD	13	CTAERR_FUNCTION_NOT_AVAIL	所要求的操作在現有的協定上是不可用的，或所需的.dsp 或.tcp 檔沒有下傳的卡板，或所要求的函數需要一被保留給 ADI 服務使用的服務。
0xE	14	CTAERR_FUNCTION_NOT_ACTIVE	企圖停止或修改沒在執行的函數。
0xF	15	CTAERR_FUNCTION_ACTIVE	企圖開始一已開始的非同步函數，或企圖於收集數位的期間接收到一數位或使數位佇列充溢。
0x10	16	CTAERR_SHAREMEM_ACCESS	問題：存取共享記憶體失敗。解答：檢查 ctdaemon 是否正在執行。
0x11	17	CTAERR_INCOMPATIBLE_REVISION	問題：一服務或服務管理不相容於已使用的 CT Access 修正版，解答：確定所有的修正版是相容的或與 NMS 開發者連絡。
0x12	18	CTAERR_RESOURCE_CONFLICT	問題：一所需資源已被使用。解答：釋放這資源再試。
0x13	19	CTAERR_INVALID_SE	企圖於其它要求懸而未決期

十六進位碼	十進位碼	錯誤名稱	敘述
		QUENCE	間開啟或關閉服務，或企圖停止一已停止的的函數，或在未收到前一個呼叫控制命令的回應前已再呼叫控制函數，或一錄或放緩衝器當不期望的被使服從。
0x14	20	CTAERR_DRIVER_OPEN_FAILED	裝置開啟失敗是因為驅動程式沒載入或者開啟已超過最大數目。
0x15	21	CTAERR_DRIVER_VERSION	這驅動程式並沒有提供所需的函數。
0x16	22	CTAERR_DRIVER_RECEIVE_FAILED	當從一裝置取回事件時發生錯誤。
0x17	23	CTAERR_DRIVER_SEND_FAILED	當傳送一訊息時由驅動程式傳回錯誤，於 AG 卡板，當 AG 卡板被重置時會發生此錯誤。
0x18	24	CTAERR_DRIVER_ERROR	問題：一裝置的驅動程式傳回錯誤碼。解答：檢查被載入及執行的裝置的驅動程式。
0x19	25	CTAERR_TOO_MANY_OPEN_FILES	問題：於現有的程序沒有更多的檔案處理可供使用。解答：關閉一些檔案或增加檔

十六進位碼	十進位碼	錯誤名稱	敘述
			案處理的極限。
0x1A	26	CTAERR_INVALID_BOARD	問題：被指定的卡板號碼沒被配置成功。解答：重配置這卡板。
0x1B	27	CTAERR_OUTPUT_ACTIVE	問題：因流與時槽於其他脈絡上早已開啟造成服務開啟失敗，或因有其它者使活動輸出函數致使輸出函數（如play）失敗。解答：檢查其它的流與時槽或停止用執行的函數。
0x1C	28	CTAERR_CREATE_MUTEX_FAILED	建立線鎖失敗
0x1D	29	CTAERR_LOCK_TIMEOUT	一線鎖時間超過。
0x1E	30	CTAERR_ALREADY_INITIALIZED	CTAccess 程式館或 CT Access 服務已被啟始化
0x1F	31	CTAERR_NOT_INITIALIZED	問題：CT Access 尚未啟始化。解答：呼叫 ctaInitialize。
0x20	32	CTAERR_INVALID_HANDLE	一無效的 CT Access 處理被當作引數傳給函數。
0x21	33	CTAERR_FILE_NOT_FOUND	問題：所指定的檔案並不存在。解答：建立這檔案或指定一不同的檔案名稱。

十六進位碼	十進位碼	錯誤名稱	敘述
0x22	34	CTAERR_PATH_NOT_FOUND	問題：這目錄若干的檔名是無效的。解答：建立這目錄或指定一不同路徑。
0x23	35	CTAERR_FILE_ACCESS_DENIED	問題：程式想要開啟唯讀檔來寫入，或以不充分的准許來開啟一檔案，或這檔案稱是一目錄。解答：改變檔案屬性或准許，或開啟為唯讀檔，或指定為不同的檔案名稱。
0x24	36	CTAERR_FILE_EXISTS	問題：企圖開啟一已存在的檔案。解答：除掉改名這已存在的檔，或指定一不同檔案名稱。
0x25	37	CTAERR_FILE_OPEN_FAILED	問題：因為系統錯誤使檔案開啟失敗。解答：驗證檔案存在。
0x26	38	CTAERR_FILE_CREATE_FAILED	問題：因為系統錯誤使檔案建立失敗。解答：檢驗這路徑是有效的及這檔案是不存在的。
0x27	39	CTAERR_FILE_READ_FAILED	問題：這檔案未開啟或被鎖住，或所期望數量的資料無法被讀取。解答：確定檔案

十六進位碼	十進位碼	錯誤名稱	敘述
			為所期望的形別，驗證一不正確的處理不用來關閉其它的檔案。
0x28	40	CTAERR_FILE_WRITE_FAILED	問題：這檔案未開啟或被鎖住，或所期望數量的資料無法被寫入。解答：驗證一不正確的處理不用來關閉其它的檔案。
0x29	41	CTAERR_DISK_FULL	問題：於硬碟沒有足夠的空間完成寫入的操作，沒有資料被寫入。解答：於硬碟上增加空間。
0x2A	42	CTAERR_CREATE_EVENT_FAILED	問題：一指定事件處理的操作源傳回錯誤。解答：與 NMS 開發者連絡。
0x2B	43	CTAERR_EVENT_TIMEOUT	
0x2C	44	CTAERR_OS_INTERNAL_ERROR	問題：一操作系統指定錯誤發生。解答：與 NMS 開發者連絡。
0x2D	45	CTAERR_INTERNAL_ERROR	問題：CT Access 發生內部錯誤。解答：與 NMS 開發者連絡。
0x2E	46	CTAERR_DUPLICATE	一早已註冊的 CTA 等待物

十六進位碼	十進位碼	錯誤名稱	敘述
		_CTAWAITOBJ	件被當作引數傳給 ctaRegisterWaitObject。
0x2F	47	CTAERR_NO_LICENSE	所指定的服務沒有有效的許可證，或所有可用的許可證都已在使用。
0x30	48	CTAERR_LICENSE_EXPIRED	這服務的許可期滿。
0x34	52	CTAERR_WRONG_COMPAT_LEVEL	問題：傳給 ctaInitialize 的相容性標準與現有的 CT Access 標準不同。解答：重編譯這應用程式
0x40	64	CTAERR_LOAD_LIB	
0x41	65	CTAERR_SVR_COMM	於客戶提出要求的要求時間內服務並未回應（預設為三秒）。
0x42	66	CTAERR_BAD_NAME	由 ctaCreateContext 或 ctaCreateContextEx 所指定的脈絡名稱早已於 CT Access 伺服使用。
0x43	67	CTAERR_ENUM_END	以指定的索引尋找到目錄的終點並未發現元件。
0x44	68	CTAERR_EVNT_BUF	
0x45	69	CTAERR_INVALID_LI	當使用 ctaInitialize 設置

十六進位碼	十進位碼	錯誤名稱	敘述
		B	ctaflags 參數，指定不由所連結的 CT Access 程式館所提供的伺服模式。
0x10000001	268435457	CTAERR_INCOMPATIBLE_PARMS	問題：檢知到兩個完全相同服務的標準參數被兩不同服務管理配置所產生的茅盾。解答：於應用程式並不需要實行兩個相同的服務。
0x10000002	268435458	CTAERR_DUPLICATE_EXTPARMS	問題：由不同服務管理所配置的同名的兩個服務有一相同（複製）的參數名稱。解答：與 NMS 開發者連絡。
0x10000003	268435459	CTAERR_WAIT_PENDING	一操作系統指定等待函數傳回一錯誤。
0x10000004	268435460	CTAERR_WAIT_FAILED	於同一 CTA 佇列處理早已呼叫過 ctaWaitEvent。
0x10000005	268435461	CTAERR_INVALID_ADDRESS	問題：一無效的命令或事件位址被 CT Access 派遣者所檢知。解答：確定所有需要的服務已被啟始化及開啟。
0x10000006	268435462	CTAERR_SERVICE_NOT_AVAILABLE	問題：企圖於一 CTA 脈絡尚未開啟的服務呼叫服務 API 函數。解答：於呼叫服務 API 前開啟服務。

十六進位碼	十進位碼	錯誤名稱	敘述
0x10000007	268435463	CTAERR_SERVICE_ERROR	
0x10000008	268435464	CTAERR_SERVICE_IN_USE	企圖以 ctaOpenServices 開啟已開啟的服務。
0x10000009	268435465	CTAERR_ALREADY_DEFINED	問題：服務及服務管理已經被定義。解答：確定沒有於 ctaInitialize 及 ctaCreateQueue 內指定冗餘的服務或服務管理
0x1000000A	268435466	CTAERR_TRACE_NOT_ENABLED	問題：於執行 ctaInitialize 時並未設置 CTA_TRACE_ENABLE。解答：確定 CTA_TRACE_ENABLE 被設置。

CT Access 事件簡介：CT Access 事件是定義在包含檔 ctadef.h 內，事件以 CTAEVN 為字首，事件表列如下表：

事件名稱	十六進位碼	十進位碼	敘述
CTAEVN_CLOSE_SERVICES_DONE	0x10002103	268443907	於 CT Access 脈絡上指定的服務已被關閉。
CTAEVN_DESTROY_CONTEXT_DO	0x10002101	268443905	CTA 脈絡已被消滅。

事件名稱	十六進位碼	十進位碼	敘述
NE			
CTAEVN_NULL_EVENT	0x10002100	268443904	ctaWaitEvent 傳回一錯誤
CTAEVN_OPEN_SERVICES_DONE	0x10002102	268443906	於 CT Access 脈絡上指定的服務已被開啟。參考 ctaOpenServices.
CTAEVN_UPDATE_WAITOBJS	0x10002005	268443653	被 CT Access 管理的等待物件的內部陣列已修改。參考 ctaQueryWaitObjects.
CTAEVN_WAIT_TMEOUT	0x10002004	268443652	ctaWaitEvent 時間超過

Ctdaemon 輸出：Ctdaemon 允許應用程式於伺服模式執行，也允許修改系統總體預設參數，設置總體追蹤標誌，及可選責擇記錄追蹤訊息到一檔案。Ctdaemon 輸出有追蹤時間印記（Trace timestamp）、一般追蹤訊息（Generic trace information）及指定形別追蹤訊息（Type-specific trace information）。

- Trace timestamp（追蹤時間印記）輸出：時間印記以如下形式顯示：Wed Dec 24 21：40：01 1997。時間印記指出事%見被記錄的時間。
- Generic trace information（一般追蹤訊息）輸出：一般追蹤訊息以如下形式顯示：|pid=ce tid=ba ctahd=80000001 （CTATEST）

uid=0 tag=4005 sev=0。其內容描述如下表：

欄位	敘述
pid	被記錄追蹤訊息程序的識別，如 ce
tid	被記錄追蹤訊息程序內線的識別，如 ba，於無線操作系統是無效的
ctahd	被記錄追蹤訊息 CTA 脈絡的識別，如 80000001(傳給 **ctaCreateContext 的**脈絡 ID 是一任意字串)
text in parentheses	被記錄追蹤訊息脈絡的名稱，如 CTATEST
uid	當建立被記錄追蹤訊息 CTA 脈絡時被指定的使用者，如 0。
tag	追蹤形別標籤的數值，如 4005。這值決定附屬於追蹤訊息的情報將如何被格式化。
sev	嚴重的錯誤。參考 ctadef.h 內的 CTA_TRACE_SEVERITY. **0 = CTA_TRACE_SEVERITY_INFO** **1 = CTA_TRACE_SEVERITY_WARNING** **2 = CTA_TRACE_SEVERITY_ERROR**

Type-specific trace information（指定形別追蹤訊息）輸出：追蹤資料的剩餘被用來指定被追蹤物件的形別，能被追蹤物件的形別為：Commands（命令）、Events（事件）、Errors（錯誤）、Strings（字串）、DWORDs 及 32-bit integers（INT32）。

- Commands information（命令情報）：命令情報顯示如後：|↓
  ADICMD_START_PROTOCOL（=13001）src=2000 dst=1 s1=60

d1=12f8c8 s2=e0 d2=0 s3=78002907 d3=0，各欄位敘述如下表：

欄位	敘述
↓	於 Windows NT，向下箭頭被顯示，在其他操作系統則不顯示。
Name	送給派遣員的命令名稱，如 **ADICMD_START_PROTOCOL.**
Number in parentheses	命令的數值，如=13001
src	來源服務識別，如 2000 = CTA_APP_SVC_ID. 服務識別值被定義於它們各自的服務頭檔(例如，adidef.h 或 ctadef.h)。
dst	目的服務識別，於本例，目的 1 對應 ADI_SVCID，服務識別值被定義於它們各自的服務頭檔(例如，adidef.h 或 ctadef.h)。
s1	與這命令相關聯的第一個緩衝器尺寸，如 60。
d1	與這命令相關聯的第一個緩衝器地址，如 12f8c8.
s2 and d2	與這命令相關聯的第二個緩衝器尺寸與地址，如 e0 or 0.
s3 and d3	與這命令相關聯的第三個緩衝器尺寸與地址，如 78002907 or 0.

- Events information（事件情報）：事件情報也能從 ctdaemon 取回，可由派遣員或應用程式產生的事件情報，事件情報顯示如後：| ↓ CTAEVN_DISP_OPEN_SERVICE_DONE（10002111）Fonished(val=1001)src=1 dst=1000 time=df8ee5ce uid=0 size=7c

buf=421a90。各欄位敘述如下表：

欄位	敘述
↑	於 Windows NT，向上箭頭被顯示，在其他操作系統則不顯示。
Name	派遣事件的名稱，如 CTAEVN_DISP_OPEN_SERVICE_DONE。
Number in parentheses	事件 ID 值，如 10002111。
String	關聯於這事件的本文原因碼，如 Finished。
val	派遣事件的值，如 1001。
src	使事件送給派遣員的服務的來源識別，如 1 = ADI_SVCID。
dst	事件被送的目的服務的來源識別，如 1000 = CTA_SYS_SVCID。
time	關聯於事件的派遣員的時間印記，如 df8ee5ce，比較其他事件的時間印記來決定事件產生的順序。
uid	使事件產生的 CTA 脈絡使用者識別 (0)(**ctaCreateContext**)。
size	關聯於這事件的事件指定緩衝器尺寸，如 7c。
buf	關聯於這事件的事件指定緩衝器地址，如 421a90。

- Errors information（錯誤情報）：錯誤情報也能從 ctdaemon 取回，錯誤情報顯示如後：| ERR： CTAERR_INVALID_STATE （=c） d1=0 d2=0 txt= fn=adiStartProtocol ln=0 file=，各欄位敘

述如下表：

欄位	敘述
Name	錯誤名稱，如 CTAERR_INVALID_STATE。
Code in parentheses	錯誤碼的值，如=c。
d1 and d2	關聯於這錯誤的錯誤指定緩衝器
fn	產生這錯誤的函數名稱，如 **adiStartProtocol.**
ln	產生錯誤的函數在來源檔內的線數，如果 ln=0，那麼產生錯誤的函數並未指定一線數。
file	產生錯誤的函數在來源檔名稱，如果 file=空白，那麼產生錯誤的函數並未指定在這檔案內。

- Strings（字串）：如果 **ctaLogTrace** 明確地被使用者以追蹤標籤（tracetag）值（CTA_TRACETAG_STRINF）呼叫，然後將出現如後字串：| debug_string。
- DWORD：如果 **ctaLogTrace** 明確地被使用者以追蹤標籤（tracetag）值（CTA_TRACETAG_DWORD）呼叫，然後將出現如後字串：| DWORD： 10（0 x a）。這第一個值是 DEORD 的十進位值，於括號內的是 DEORD 的十六進位值。
- 32-bit integers（INT32）：如果 **ctaLogTrace** 明確地被使用者以追蹤標籤（tracetag）值（CTA_TRACETAG_INT32）呼叫，然後將出現如後字串：| INT32： 10（0xa），這第一個值是 32 位元整數的十進位值，於括號內的是 32 位元整數的十六進位值。

cta.cfg 檔案：cta.cfg 是 CT Access 的配置檔，它被 **ctdaemon、ctaInitialize** 及 **ctaLoadParameterFile** 使用。CT Access 的配置檔包含章節頭（section headers），對應的章節本體（section-specific body）及命令（command），如下表：

檔案章節	敘述
章節頭	定義在中括號內於本文檔的欄 1 開始，例如：[ctasys] 指定 CT Access 系統配置檔頭。
章節指定本體	由說明實體所定義，例如 API。這章節的範圍開始於章節頭並結束在另一章節頭，或檔案的終點。
命令	以#開頭，任何在#之後的本文都視為命令。

- Access 系統頭是被 ctdaemon 及 ctaInitialize 所翻譯，CT Access 系統頭的範圍開始於【ctasys】，結束於下一章節頭前。
- 應用程式章節：【application】開始應用程式於程式模式或伺服模式執行的宣告，也包含是否於伺服模式當發現於伺服早已使用，自動修改所指派的脈絡名稱（如此項不致能，當於伺服模式發現重複的脈絡名稱會傳回錯誤）。
- 伺服章節：開始如何執行 ctdaemon 的宣告，包括設置參數管理、追蹤標籤、遠端追蹤，這參數旗標（ParmFlags）與追蹤模式（TraceMode）的設置僅影響伺服模式，所有使用伺服模式的應用程式看見相同的參數。
- ctapar 章節：【ctapar】開始不顧 CT Access 預設參數的宣告，這將取代預設值，這章節由 ctdaemon 及 ctaLoadParameteFile 所翻譯。

參數管理服務（PRM，Parameter Management）：這 CT Access 參數管理服務允許應用程式開發者如使用其他 CT Access 服務參數的相同方法使用應用程式定義參數，當使用者指定參數經由 PRM 服務啟始化，應用程式可取回及/或修改所定義的參數，可於

共享記憶體內、處理過的總體記憶體、或一 CTA 脈絡主要成份。參數被定義於一 ASCII 本文檔內並由 PRM 服務所載入，PRM 服務能載入 ASCII 定義檔或這 ASCII 檔的二進位編譯版本，如果不鼓勵由終端使用者修改參數，請編譯為二進位版本。於 CT Access 下定義應用程式指定參數步驟如下：

- 使用任何 ASCII 本文編輯器（如 Microsoft notepad）來建立一 ASCII 本文參數檔（prmxxx.pf，prm 是字首，xxx 用來識別這檔案，.pf 是檔案的延伸）。
- 使用 pfencode 轉譯公用程式（選項）編譯這 ASCII 檔為一二進位參數檔。
- 確定所產生的 pf 檔所在的目錄在 CTA_DPATH 環境變數所指定的目錄表列裡面。
- 於應用程式裏，於傳給 ctaInitialize 的表列，除了所有其他想要的 CT Access 服務外也指定這 PRM 服務。
- 使用所提供的 CT Access 參數函數來存取參數（ctaSetParmByName，ctaGetParmByName，ctaGetParmInfo，ctaGetParm，ctaRefreshParm，ctaGetParmID，ctaLoadParameterFile）。

  CT Access 參數回顧：一參數是由下表的元素所組成：

元件	範例
ASCII 名稱	ADI.COLLECT.intertimeout
資料形別	DWORD
單位	ms
值	7000

- 參數名稱：CT Access 參數被依如下的語法命名：SvcName.category.fieldName 或 SvcName.category.subStructure.fieldName。如下表敘述：

名稱	敘述
SvcName	參數所屬的服務
Category	允許多樣的，邏輯上同族的 **fieldNames** 被成群體
SubStructure	允許多樣的，邏輯上同族的 **fieldNames** 被成一 Category 下的堆疊
FieldName	這實際的參數名稱

例如，這 ADI 服務包含多樣化的 Category，每個 Category 包含多樣化的 FieldName 舉例如下：

- ADI.CALLPROG.busycount
- ADI.CALLPROG.leakagetime
- ADI.CALLPROG.maxbusy
- ADI.COLLECT.firsttimeout
- ADI.COLLECT.intertimeout

參數形別：一 CT Access 參數可被定義為數量型別或字串，例如：ADI.RECORD.AGCattacktime 參數是一 DWORD。

參數單位：一 CT Access 參數也有一相關的單位，如 dB 或 dBm，單位的初步用途是為顯示使用。

參數值：一 CT Access 參數有一相關的值，可以設置及取回，一參數於啟始時有一預設值。

使用 CT Access 以名稱存取參數：CT Access 應用程式能呼叫 ctaGetParmByName 來取回任何參數的值。

應用程式參數：參數管理服務提供一動態機構來宣告及應用程式指定參數。

6 Natural Call Control（NCC）

CT Access 是提供電話功能標準程式介面之發展環境，在 CT Access 內部又將諸多電話功能細分為群組（Groups）或稱為服務（Services），NCC 即為其中的一個群組，NCC 即是 Natural Microsystem 公司針對 CAS（Channel Associated Signaling）協定所發展之服務，故在 CT Access 稱之為 NCC services。值得一提的，CT Access 與硬體之間是互相獨立的，NCC service 與 CT Access 之間也是獨立的，為了這個目地，將 NCC 與硬體 Trunk 間之實際通信，委由另外之軟體實體來負責，稱之為 TCPs（Trunk Control Programs），TCPs 針對不同之協定發展對應之 TCP，如 NCC 所包含的十數個協定每一個都有對應之 TCPs （如下表所示），Natural Microsystem 公司先前使用 ADI services 作為 CAS 協定方面之服務，現可遷移至功能更強大之 NCC services，且不必更換硬體及 CT Access，這種容易擴展之發展平台及環境，正是未來發展之趨勢。

協定	TCP 檔名
Analog Loop Start	Lps0.tcp
Australian P2	Ap20.tcp
Ground Start	Gds0.tcp
European Digital CAS	Euc0.tcp
Feature Group D	Fgd0.tcp
MF-Socotel protocol	Mfs0.tcp
MFC- R2 protocol	Mfc0.tcp
Off-Premises Station	Ops0.tcp
Operator Work Station	Sta0.tcp
Pulsed E & M	Eam0.tcp

協定	TCP 檔名
Signaling System 5	Ss50.tcp
System R1.5	R150.tcp,r151.tcp
Digital and Analog Wink Start	wnkx（x=0,1）

NCC 針對 AG 系列的硬體，提供了下列的 AG（Alliance Generation） CAS 協定群組，條例如下：

- Analog Loop Start protocol（LPS）：可分成兩種狀態描述如下：

狀態一：由交換機端向電話端要求通話

狀態	交換機輸出狀態	迴路狀態	被叫電話機輸入狀態
閒置		無迴路電流	
振鈴	提供振鈴電壓	振鈴電壓	(電話機響鈴)
當第一次振鈴結束，交換機即將主叫號碼送到回路上，電話機在未接電話前即可檢知來話號碼。			
接電話		回路電流	"Off-hook"
若沒接電話或電話使用完掛電話，將出現如下之狀態			
清除		迴路電流截止或信號音清除	
閒置		無迴路電流	

狀態二：由電話端向交換機端要求通話

狀態	主叫電話機輸出狀態	迴路狀態	交換機輸入狀態
閒置		無迴路電流	
拿起電話筒	"Off-hook"	回路電流	
撥號		撥號音信號	
交換機通知被叫端		振鈴音信號	
如果被叫端忙線，則迴路出現忙線音			
交談		聲音信號	(交談中)
清除		迴路電流截止或信號音清除	
閒置		無迴路電流	

- Australian P2 protocol（AP2）：使用 CCITT MFC-R2 協定（Recommendation Q.421）之線上信號組合（A、B、C、D Code），其中 C、D 碼不使用，C 碼設為"0"，D 碼設為"1"。下例為一典型的打電話之線上信號狀態：

狀態	A_fB_f 輸出狀態	方	A_bD_b 輸入狀態
閒置	10	←→	10
佔線	00	→	01
佔線確認	00	←	11
振鈴	00	←	11
應答-交談狀態	00	←	01
順向清除（當被叫端未接電話）	10	→	11
閒置	10	←	10
應答-交談狀態	00	←	01
公告脈衝	00	←	11 or 00
應答-交談狀態	00	←	01
被叫端先掛斷電話: 反向清除	00	←	11
順向清除	10	→	11
閒置	10	←→	10
主叫端先掛斷電話: 順向清除	10	→	01
釋放佔線	10	←	11
閒置	10	←→	10

- Digital Ground Start protocol（GDS）：雖然 E1 CAS 碼框提供雙向 A、B、C、D 位元，但只有 A、B 兩位元用於 GDS 線上信號，於順向為 A_fB_f 位元，於反向為 A_bB_b 位元，順向通道之 A_fB_f 指示交換輸出設備之狀況並影響主叫端電話線上狀態，反向通道之 A_bB_b 指示被叫端（輸入設備）線上狀況。

其餘 C 和 D 位元通常為固定，但於網路間卻是可變動的。

所有狀態的改變都是由主叫端決定，下二表為 DGS 信號 FX 和 SA 的兩種狀況。

交換端打電話到終端設備之狀況如：

狀態	交換機輸出狀態	迴路狀態	被叫電話機輸入狀態
閒置	11 (FX) 01 (SA)	←→	01 (FX) 00 (SA)
佔線	01 (FX) 11 (SA)	→	01 (FX) 00 (SA)
振鈴開啟	00 (FX) 10 (SA)	→	01 (FX) 00 (SA)
振鈴關閉	01 (FX) 11 (SA)	→	01 (FX) 00 (SA)
應答-交談狀態	01 (FX) 11 (SA)	←	11 (FX) 10 (SA)
被叫端先斷線	01 (FX) 11 (SA)	←	01 (FX) 00 (SA)
主叫斷線	11 (FX) 01 (SA)	→	11 or 01 (FX) 10 or 00 (SA)
閒置	11 (FX) 01 (SA)	←→	01 (FX) 00 (SA)

終端設備為主叫端之情況如下：

狀態	主叫電話機輸出狀態	迴路狀態	交換機輸入狀態
閒置	01 (FX) 00 (SA)	←→	11 (FX) 01 (SA)
佔線	00 (FX) 01 (SA)	→	11 (FX) 01 (SA)
佔線確認	00 (FX) 01 (SA)	←	01 (FX) 11 (SA)
掛斷電話	11 (FX) 10 (SA)	→	01 (FX) 11 (SA)
傳送過程	11 (FX) 10 (SA)	←dial tone	01 (FX) 11 (SA)
應答-交談狀態	11 (FX) 10 (SA)	←voice	01 (FX) 11 (SA)
被叫端先斷線	11 (FX) 10 (SA)	←	11 (FX) 01 (SA)
主叫斷線	01 (FX) 00 (SA)	→	11 or 01 (FX) 01 or 11 (SA)
閒置	01 (FX) 00 (SA)	←→	11 (FX) 01 (SA)

- Feature Group D protocol（FGD）：FGD 信號源於 Digital and Analog Wink Start 協定，控制 E1 CAS 碼框的 A、B 位元，FGD 協定影響順向通道之 A_fB_f 指示交換輸出設備之狀況並影響主叫端電話線上狀態，及反向通道之 A_bB_b 指示被叫端（輸入設備）線上狀況。其餘 C 和 D 位元通常為固定，但於網路間卻是可變動的。

  下例為一典型的打電話之線上信號狀態：

狀態	A_fB_f 輸出狀態	方向	A_bD_b 輸入狀態
閒置	00	\xdf	00
佔線	11		00
佔線確認	11	\xdf	00-11-00 (wink)
註冊第一欄信號: 數位輸出	MF tones		00
確認第一數位序		\xdf	00-11-00 (wink)
註冊第二欄信號: 數位輸出	MF tones		00
順向清除及閒置	00		00
應答-交談狀態	11	\xdf	11
被叫端先掛斷電話: 反向清除	11	\xdf	00
主叫端先掛斷電話: 順向清除	00		00 or 11
閒置	00	\xdf	00

- MF-Socotel protocol（MFS）：雖然 E1 CAS 碼框提供雙向 A、B、C、D 四位元，但 MFS 線上信號一般只需 A 位元即夠使用，於順向為 A_f 位元，於反向為 A_b 位元，順向通道之 A_f 指示交換輸出設備之狀況並影響主叫端電話線上狀態，反向通道之 A_b 指示被叫端（輸入設備）線上狀況。

Bb 位元只有在運送公告脈衝到輸出設備時使用，Bf 則不使用設為"1"， 其餘 C 和 D 位元於雙向設為"0"與"1"。

下例為一典型的打電話之線上信號狀態：

狀態	A_fB_f 輸出狀態	方向	A_bD_b 輸入狀態
閒置	11	←→	11
佔線	01	→	11
佔線確認	01	←	01
振鈴	01	←	01
應答-交談狀態	01	←	11
順向清除	11	→	01
閒置	11	←	11
應答-交談狀態	01	←	11
公告脈衝	01	←	10
應答-交談狀態	01	←	11
被叫端先掛斷電話：反向清除	01	←	01
順向清除	11	→	01
閒置	11	←	11

狀態	A_fB_f 輸出狀態	方向	A_bD_b 輸入狀態
主叫端先掛斷電話: 順向清除	11	→	11
釋放佔線	11	←	01
閒置	11	←→	11

- Multi-Frequency Compelled R2 protocol（MFC）：雖然 E1 CAS 碼框提供雙向 A、B、C、D 四位元，但 R2 線上信號一般只需 A、B 位元即夠使用，於順向為 A_fB_f 位元，於反向為 A_bB_b 位元，順向通道之 A_fB_f 指示交換輸出設備之狀況並影響主叫端電話線上狀態，反向通道之 A_bB_b 指示被叫端（輸入設備）線上狀況。

  其餘 C 和 D 位元通常為固定，但於網路間卻是可變動的。

  下例為一典型的電話之線上信號狀態：

狀態	A_fB_f 輸出狀態	方向	A_bD_b 輸入狀態
閒置	10	←→	10
佔線	00	→	10
佔線確認	00	←	11
振鈴	00	←	11
應答-交談狀態	00	←	01
順向清除	10	→	11
閒置	10	←	10
應答-交談狀態	00	←	01
公告脈衝	00	←	11 or 00

狀態	A_fB_f 輸出狀態	方向	A_bD_b 輸入狀態
應答-交談狀態	00	←	01
被叫端先掛斷電話: 反向清除	00	←	11
順向清除	10	→	11
閒置	10	←→	10
主叫端先掛斷電話: 順向清除	10	→	01
釋放佔線	10	←	11
閒置	10	←→	10

- Off-Premises Station protocol（OPS）：雖然 E1 CAS 碼框提供雙向 A、B、C、D 四位元，但 OPS 線上信號一般只需 A、B 位元即夠使用，於順向為 A_fB_f 位元，於反向為 A_bB_b 位元，順向通道之 A_fB_f 指示交換輸出設備之狀況並影響主叫端電話線上狀態，反向通道之 A_bB_b 指示被叫端（輸入設備）線上狀況。

其餘 C 和 D 位元通常為固定，但於網路間卻是可變動的。

所有狀態的改變都是由主叫端決定，下二表為 OPS 信號 FX 和 SA 的兩種狀況。

交換端打電話到終端設備之狀況如：

狀態	交換機輸出狀態	迴路狀態	被叫電話機輸入狀態
閒置	01 (FX) 11 (SA)	←→	01 (FX) 00 (SA)
振鈴開啟	00 (FX) 10 (SA)	→	01 (FX) 00 (SA)

狀態	交換機輸出狀態	迴路狀態	被叫電話機輸入狀態
振鈴關閉	01 (FX) 11 (SA)	→	01 (FX) 00 (SA)
應答-交談狀態	01 (FX) 11 (SA)	←	11 (FX) 10 (SA)
清除和閒置	01 (FX) 11 (SA)	←	01 (FX) 00 (SA)

終端設備為主叫端之情況如下：

狀態	主叫電話機輸出狀態	迴路狀態	交換機輸入狀態
閒置	01 (FX) 00 (SA)	←→	01 (FX) 11 (SA)
佔線	11 (FX) 10 (SA)	→	01 (FX) 11 (SA)
佔線確認	11 (FX) 10 (SA)	← Dial tone	01 (FX) 11 (SA)
叫程序音	11 (FX) 10 (SA)	← Ring tone	01 (FX) 11 (SA)
應答-交談狀態	11 (FX) 10 (SA)	← Voice	01 (FX) 11 (SA)
清除和閒置	01 (FX) 00 (SA)	→	01 (FX) 11 (SA)

- Operator Work Station protocol（STA）：所有狀態的改變都是由主叫端決定，下表為 STA 信號由 PBX 主叫終端設備之狀態。

狀態	輸出 PBX	迴路	輸入終端
閒置		無迴路電壓	
振鈴	提供振鈴電壓	振鈴電壓	(電話振鈴)
應答-交談狀態		← 迴路電流，聲音	"Off-hook"
PBX 先清除 (選項)		迴路電流截斷，或清除音頻	
清除		← 迴路電流截斷	"On-hook"
閒置		無迴路電流	

下表為 STA 信號由終端設備 PBX 主叫 PBX 之狀態。

狀態	輸出終端	迴路	輸入 PBX
閒置		無迴路電流	
佔線	"Off-hook"	迴路電流	
佔線確認		← 撥號音	
叫程序音		← 振鈴音	
應答-交談狀態		← 聲音	

狀態	輸出終端	迴路	輸入 PBX
PBX 先清除 (選項)		← 迴路電流截斷，或清除音頻	
清除	"On-hook"	→ 迴路電流截斷	
閒置		無迴路電流	

- Pulsed E & M protocol（EAM）：EAM 協定雙向只使用 A 位元，脈衝分為長、短兩種，依電話現在狀態之上下文而代表不同意義，下例為一典型的打電話之線上信號狀態：

狀態	輸出	方向	輸入
閒置	閒置碼	←→	閒置碼
佔線	短脈衝	→	閒置碼
佔線確認	閒置碼	←	短脈衝(通常不需要)
振鈴	閒置碼	←	閒置碼
應答-交談狀態	閒置碼	←	短脈衝
順向清除	長脈衝	→	閒置碼
閒置	閒置碼	←	長脈衝
應答-交談狀態	閒置碼	←→	閒置碼
公告脈衝	閒置碼	←	短脈衝
應答-交談狀態	閒置碼	←→	閒置碼

狀態	輸出	方向	輸入
被叫端先掛斷電話: 反向清除	閒置碼	←	長脈衝
順向清除	長脈衝	→	閒置碼
釋放佔線(有些國家)	閒置碼	←	長脈衝
閒置	閒置碼	←→	閒置碼
主叫端先掛斷電話: 順向清除	長脈衝	→	閒置碼
釋放佔線	閒置碼	←	長脈衝
閒置	閒置碼	←→	閒置碼

- Signaling System 5 protocol（SS5）：SS5 協定線上信號使用 2400Hz 及 2600Hz 兩種頻率，以單頻或雙頻方式收發信號。這兩個頻率與註冊信號音頻相差有一段距離，所以不易誤檢，由於不含信號位元，能被用於類此通道之通話建立，目前也用於數位 T1 及 E1 通道之電話連線建立上。

下例為一典型的電話之線上信號狀態：

狀態	順向調頻音	方向	反向調頻音
閒置	無		無
佔線	f_1 (2400 Hz)	→	
持續傳送		←	f_2 (2600 Hz)
此時輸出端以 MF 音傳送位址信息給輸入端，若輸入端接受建立電話，將會回傳振鈴音。當雙方建立此電話後，開始通話			
振鈴		←	振鈴音
應答		←	f_1 (2400 Hz)
應答確認	f_1 (2400 Hz)	→	

狀態	順向調頻音	方向	反向調頻音
若被叫端拒絕接聽，其音頻會改變如下			
短促忙線音		←	f_2 (2600 Hz)
短促忙線確認	f_1 (2400 Hz)	→	
閒置	無		無
當預到輸入端無法送出短促忙線音時，可由當地的網路代替傳送			
由於輸出輸入掛斷電話之順序不同，會傳送順向清除信號或反向清除信號，當反向清除信號被輸出端確認後，會送出順向清除信號，當確認後會送出釋放佔線信號，然後進入閒置狀態			
輸入端先掛電話：反向清除		←	f_2 (2600 Hz)
反向清除確認	f_1 (2400 Hz)	→	
暫停：兩連續信號在同方向	100 ms minimum		
順向清除	$f_1 + f_2$ (2400+2600 Hz)	→	
釋放佔線		←	$f_1 + f_2$ (2400+2600 Hz)
閒置	無		無
輸出端先掛電話：順向清除	$f_1 + f_2$ (2400+2600 Hz)	→	
釋放佔線		←	$f_1 + f_2$

狀態	順向調頻音	方向	反向調頻音
			(2400+2600 Hz)
閒置	無		無

- System R1.5 protocol（R15）：R15 協定使用 A、B 兩個雙向位元來控制通話狀態，電話線同時呈現信號位元與音頻的聯合信號，通常用 MF 音作為註冊信號，當輸出裝備送出一 MF 音後（一位址位元），輸入裝備會反送一 MF 音以要求下一個位址位元（MF 音），所有 MF 音依時間不會互相強迫。因為有些國家（如俄國）沒有 MF 調頻音之檢知設備，使用 R15 協定之輸出入裝備一般以 MF 脈衝為預設註冊信號，若裝備改進，可再改為使用 MF 調頻音為註冊信號。
- Digital and Analog Wink Start protocol（WNK）：Digital and Analog Wink Start 協定具對稱性，由網路端及用戶端同步完成，WNK 協定同樣影響 A、B bit（順向為 A_fB_f，反向為 A_bB_b），順向通道之 A_fB_f 指示交換輸出設備之狀況並影響主叫端電話線上狀態，反向通道之 A_bB_b 指示被叫端（輸入設備）線上狀況。

在臺灣閒置碼 AB=00，下例為一典型的打電話之線上信號狀態：

狀態	A_fB_f 輸出狀態	方向	A_bD_b 輸入狀態
閒置	00	←→	00
佔線	11	→	00
佔線確認	11	←	00-11-00 (wink)
註冊信號: 數位輸出	DTMF or MF	→	00

狀態	A_fB_f輸出狀態	方向	A_bD_b輸入狀態
註冊信號: 數位輸出	00 pulse on	→	00
	11 pulse off	→	00
順向清除及閒置	00	→	00
應答-交談狀態	11	←	11
被叫端先掛斷電話: 反向清除	11	←	00
主叫端先掛斷電話: 順向清除	00	→	00 or 11
閒置	00	←→	00

7 CompactPCI

CompactPCI 為工業標準 HA 平台，可靠度（Reliability）包含堅固的機械環境，易於管理的環境熱量及 NEBS 保證的平台。可維護性（Maintainability）包含容易存取及快速替換。高可利用性（Availability）即為可靠度加上可維護性加上功能冗餘及優良的排列。

目前的 HA 特性為改良的可維護性，如前面插拔卡板，可變通的前後板輸出入位置，可熱插拔之卡板及具管理平台能力。改良的可靠度包含具有 NEBS 能力之包裝和改良的熱量環境。具 HA 電源及熱量管理之低階叢集平台。

高可利用性之下一步發展為熱插拔（Hot Swap）之持續改善，可熱抽換之軟體，具多運算及冗餘系統插槽之 N+1 冗餘功能及可交換之架購。

8 IP Phone

就如一般使用的電話一樣，IP Phone 為 VoIP 服務系統中使用者直接接觸的設備，也應具有一般電話現在應具備之功能，如指定轉接、通話轉移、通話保留、三方通話、、、等等，也可以說將電信系統之電話相關功能移植到網路上之翻版。

為了共通性，TIA（Telecommunications Industry Association）與 EIA（Electronic Industries Alliance）合作訂定了 IP Phone 的規格標準 IS-811，其最基本規格如下：

- 顯示面板：至少能顯示 2 x 16 之英文字母與數字及 Unicode Pages U+0000->U+00ff 中所定義的符號。
- 按鍵：至少提供*#0123456789 十二個按鍵，及按鍵之 DTMF 音。
- 音質：需支援 TIA/EIA/IS-810 中對音質校能之要求，如 Codecf 方面須支援 ITU-T G.711 之基本要求，Handset 需符合 TIA/EIA/IS-810A 第五部份規範，Headset 需符合 TIA/EIA/IS-810A 第六部份規範，Hands-Free 需符合 TIA/EIA/IS-810A 第七部份規範。
- 乙太網路：須符合 IEEE 802.3 標準之規範。
- 網際網路協定：至少須符合 IPv4（Internet Protocol,version 4）、ICMP（Internet Control Message Protocol）、UDP（User Datagram Protocol）、ARP（Address Resolution Protocol）、DHCP（Dynamic Host Configuration Protocol）、RTP（Real-Time Protocol）。
- 電信特性須具備撥出接通電話、來話位址別名顯示、來話名稱顯示、來話位址別名顯示限制等基本功能，來話直接轉接（來話詢問後轉接）、設定轉接、無條件轉接、無人應答時轉接、忙線中轉接等轉接功能，通話中保留（駐留或

接回）之保留功能，留言等候指示燈之提示功能，三方通話。

- 訊號控制協定：且至少支援 H.323 協定（配合 TCP 傳輸協定）、MEGACO 協定（配合 UDP 傳輸協定）、SIP 協定（配合 UDP 傳輸協定）中的一種。

9 IP PBX（網際網路用戶交換機）

電話、用戶交換機（PBX）、電腦、網路對現代企業是不可或缺之設備，隨著資訊管理的日益重要，及於網際網路傳送聲音技術之發展，要求將上述設備整合之需求日增，於是發展出網際網路用戶交換機（IP PBX）。

網際網路用戶交換機有許多優點，如整合所有電子應用於一體，可將語音（Audio）、視訊（Video）、資料（Data）等相關媒體完全整合，並可透過圖形化之人機介面作管理及相關應用。另因網際網路電話通信協定的訂定，更強化了維護的便利，後勢看好，所以目前各相關通訊網路大廠如 Cisco、Lucent、3Com、Siemens、Ericsson 等等，莫不卯足全力發展 IP PBX。

目前 IP PBX 之發展有兩種方向，第一種是純網路電話使用之用戶交換機， IP PBX 系統只使用 IP 網路，所有話機皆透過區域網路與交換機連接，此種 IP PBX 需另透過 GETWAY 連接外線，另一種為混合式，即可同時接傳統電話機與網路交換機，並可直接接外線。前者容量較大，擴充容易、彈性大適用於大企業或有擴張性的企業，後著容量小，擴充較難且彈性小，適用於不具擴張性小企業。以趨勢而言將朝向純網路電話使用之用戶交換機。

本次用戶研討會，內容包羅萬象，不過對『中華電信 GSM900、DCS1800 行動電話傳真數據分配處理系統』之研發架設及後續計畫之建案，可以有以下效益價值：

1. 使用 Natural Access 平台，於 AG 系列硬體上發展 CEPT，CAS 介面之相關運用軟體。
2. 使用 Natural Access 平台，於 TX 系列硬體上發展 SS7 介面之相關運用軟體。
3. 了解通訊與網路之發展趨勢，及 VoIP 通訊監察是未來必將面臨之課題。

十月二十八日　由台灣出發，抵達印尼巴里島。

十月二十九日　參加『Natural MicroSystem 亞洲工程師研討會議』。

十月三十日　參加『Natural MicroSystem 亞洲工程師研討會議』。

十月三十一日　參加『Natural MicroSystem 亞洲工程師研討會議』。

十一月一日　參加『Natural MicroSystem 亞洲工程師研討會議』。

十一月二日　參加『Natural MicroSystem 亞洲工程師研討會議』。

十一月三日　參加『Natural MicroSystem 亞洲工程師研討會議』。

十一月四日　由印尼巴里島回台灣。

因時間短絀，且人生地不熟，大部時間皆在參加研討會，也有機會參觀NMS協力廠商以NMS產品為基礎所發展的商用產品展示，如 InfoTalk 公司的 ASR（Automatic Speech recognition）展示，iSoftel 公司的 Renaissance（有關通訊話務的帳單管理結算系統），Brooktrout Sofrware 公司的 IVR(Interactive Voice Response)，CTI 及話務管理發展平台，Wilco 公司的 PCX-System（提供智慧型電話技術解決方案）及 FORCE 公司的 CopactPCI 平台等，與許多於國內都沒機會接觸的國外領導廠商，作第一類接觸，實受益良多。

隨著科技的發展，造成競爭日益激烈，一則迫使公司的國際化以降低成本，再則也迫使新產品的研發一次比一次更需要龐大的人力物力及堅實的工業基礎，也造成了大則恆大的世界潮流。本院雖為國內軍事工業之重鎮，但卻無法與世界性大廠在基礎科技上一爭長短，如本次參展的國際大廠產品若改由本院研發，結果必然是性能比不上國際大廠產品或是成本過高。所幸隨著科技的日益發展，分工也越為精細，縱使世界性大廠也無法通吃，以本院的人力物力若朝向系統整合商發展，具有一定的優勢，所以學習及應用領導大廠的最先進產品為立足系統整合這塊領域的不二法門，未來.朝向民營或財團法人似乎是不可避免的趨勢，也請有關單位能早日指出本院未來的發展方向，讓同仁能早日朝此方向努力。