21世纪计算机语音通信开发技术丛书 5

本书配套光盘内容包括：
与本书配套的电子书

计算机语音通信核心技术内幕
CT Media
程序设计参考手册
（第一卷）

21世纪计算机语音通信开发技术丛书编委会 编写

北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

21 世纪计算机语音通信开发技术丛书 5

计算机语音通信核心技术内幕

CT Media

程序设计参考手册

（第一卷）

21 世纪计算机语音通信开发技术丛书编委会 编写

www.php.com.cn

内 容 简 介

计算机语音通信技术（CT）是新世纪中最热门的技术。Dialogic 公司是开放式 CT 技术的全球先导者。Dialogic 产品广泛应用于语音、传真、数据、语音识别、声音合成、互联网电话和呼叫中心管理等一系列商业领域。使用 Dialogic 公司提供的模块化和无阻塞的部件与技术服务，开发商可以迅速灵活地设计出融合语音与数据网络的商业通信方案，以满足不断增长的社会需求。为开发和设计新的商业通信方案，维护、管理和拓展已建立的通信系统，抓住商机发展通信事业，都要求更多地了解和精晓通信和网络通信中的语音技术。为通信领域开发商、技术支持和维护人员以及技术用户，提供系统、完善和最新的技术资料，我们组织了本套丛书，共 7 本，本书是其中之一。

Dialogic 公司研发的 CT Media for Windows NT 是第一个开放的软件平台，用来设计标准的电信服务以支持各种开发商的信息、交互式语音响应系统、传真、自动呼叫转发和其他应用。CT Media 是一种 Client/Server 方式的资源管理软件，它使得按照 ECTF S.100 和 TAPITM 标准设计的多个应用程序可以共享公共的电脑语音(CT)服务器和现存的技术。另外，CT Media 还为 SCBus 和 H.100 技术硬件提供了一个开放的接口，允许将新技术加到服务器上，而不需要改变现存的应用程序。

本书与第二卷构成一套完整的手册。该书全面地介绍了如何利用 CT Media API 函数对 CT Media 的管理、应用程序配置文件、会议、连接、容器、CT Media 函数、组、KVset 与其它数据类型、系统呼叫路由器、会话、符号等进行操作。本书提供了相关 API 函数的详细说明、完成事件、可能返回的错误、所使用的数据结构和参数类型以及相应的示例程序，从而使 CT Media 用户能够方便地编写应用程序。

本书内容新、实用性、可操作性和指导性强，层次清晰，内容详尽，不但是从事 CT Media 系统管理员和应用程序开发人员的重要开发工具书，而且直接面向对象和 Client/Server 编程技术，同时也是高等院校相关专业师生教学、自学参考书和国内科研院所各图书馆重要馆藏图书。

本书光盘内容包括与本书配套的电子书。

系　列　书：	21 世纪计算机语音通信开发技术丛书（5）
书　　名：	计算机语音通信核心技术内幕——CT Media 程序设计参考手册（第一卷）
文本著作者：	21 世纪计算机语音通信开发技术丛书编委会　编写
CD 制 作 者：	希望多媒体创作中心
CD 测 试 者：	希望多媒体测试部
责 任 编 辑：	郭淑珍
出版、发行者：	北京希望电子出版社
地　　址：	北京海淀路 82 号　100080
	网址：www.bhp.com.cn　E-mail: lwm@hope.com.cn
	电话：010-62562329,62541992,62637101,62637102,62633308,62633309
	（发行和技术支持）
	010-62613322-215（门市）　010-62531267（编辑部）
经　　销：	各地新华书店、软件连锁店
排　　版：	希望图书输出中心
CD 生 产 者：	北京中新联光盘有限责任公司
文本印刷者：	北京广益印刷厂
规 格 / 开 本：	787×1092 毫米　1/16 开本　19.375 印张　429 千字
版 次 / 印 次：	2000 年 8 月第 1 版　2000 年 8 月第 1 次印刷
印　　数：	0001～3000 册
本 版 号：	ISBN7-900049-01-0/TP・01
定　　价：	50.00 元(1CD，含配套书)

21世纪计算机语音通信开发技术丛书

编 委 会 名 单

序

计算机语音通信技术（CT）是新世纪中最热门的技术。Dialogic 公司是开放式 CT 技术的全球先导者，是一个提供开放的、高品质的、基于标准的电信和计算机语音集成部件的国际供应商，在该领域占有全球 64%的市场份额。Dialogic 产品广泛应用于语音、传真、数据、语音识别、声音合成、互联网电话和呼叫中心管理等一系列商业领域。使用 Dialogic 公司提供的模块化和无阻塞的部件与技术服务，开发商可以迅速灵活地设计出融合语音与数据网络的商业通信方案，以迎合不断增长的社会需求。目前，基于 Dialogic 产品的系统已开始进入国内不少部门，被用于管理电话、传真和由计算机通过有线和无线网络来应答的多媒体呼叫系统。

随着通信市场的不断增长，通信技术以及领先的 Dialogic 技术已深入到与每一位社会人息息相关的程度。开发和设计新的商业通信方案，维护、管理和拓展已建立的通信系统，抓住商机发展通信事业，都要求更多地了解和精晓通信和网络通信中的语音技术。为通信领域开发商、技术支持和维护人员以及技术用户，提供系统、完善和最新的技术资料，无疑已迫在眉睫。

Dialogic 公司的 CT 成为通信行业最具特点的技术领域。为满足国内相关领域技术用户、系统管理、网络维护、应用编程和开发人员的要求，我社组织了本丛书——**21 世纪计算机语音通信开发技术丛书**。本丛书由以下 7 种图书组成。

1．**计算机语音通信核心技术内幕——CT Media 程序设计参考手册（第一卷）**。本书分一、二两卷，主要针对 CT Media 系统管理员和应用程序开发人员在 Windows NT 环境下开发应用程序而编写的，是一本 API 函数参考手册，支持面向对象和 Client/Server 编程技术。

Dialogic 公司研发的 CT Media for Windows NT 是第一个开放的软件平台，用来设计标准的电信服务以支持各种开发商的信息、交互式语音响应系统、传真、自动呼叫转发和其他应用。CT Media 是一种 Client/Server 方式的资源管理软件，它使得按照 ECTF S.100 和 TAPITM 标准设计的多个应用程序可以共享公共的电脑语音(CT)服务器和现存的技术。另外，CT Media 还为 SCBus 和 H.100 技术硬件提供了一个开放的接口，允许将新技术加到服务器上，而不需要改变现存的应用程序。

本书全面地介绍了如何利用 CT Media API 函数对 CT Media 的管理、应用程序配置文件、会议、连接、容器、CT Media 函数、组、KVset 与其它数据类型、系统呼叫路由器、会话、符号等进行操作。提供了相关 API 函数的详细说明、完成事件、可能返回的错误、所使用的数据结构和参数类型以及相应的示例程序，从而使 CT Media 用户能够方便地编写应用程序。本书是一本开发工具书。

2．**计算机语音通信核心技术内幕——CT Media 程序设计参考手册（第二卷）**。本书与第一卷构成一套完整的手册。它全面介绍了数据通信领域的八个应用程序编程接口，它们是：自动语音识别（ASR：Automatic Speech Recognition）、呼叫频道资源（CCR：Call Channel Resource）、传真接收器资源（Facsimile Receiver Resource）、传真发送器（Facsimile Sender）、播放器（Player）、记录器（Recorder）、信号检测器（SD：Signal Detector）、信号发生器（SG：Signal Generator）。

本书详细地介绍了上述各类编程接口的属性和参数、常数和符号、完成事件和主动提供的事件、函数引用和出错码，并且给出了相应的示例程序代码。

3．**计算机语音通信核心技术内幕——CT Media 应用程序开发指南**。Dialogic Computer Telephony Server（简称 CT Server 或 Server）和相关的 CT Media 是一个客户/服务端开发环境，用于开发计算机电话应用程序和核心信号处理技术。本书为计算机电话应用程序开发人员介绍了如何使用和掌握 CT Media。

全书分 11 章，讨论了 CT Media 基本知识、编写一个 CT 应用程序、应用程序前的准备、将应用程

序连接到 CT Server、在应用程序中获得呼叫、在程序中录制消息、检测应用程序中的信号、运行应用程序、CT Sim、演示 WinVote 应用程序以及异步编程等。全书条理清晰，讲解与示例相结合，有助于读者掌握 CT Media 的使用。

4．计算机语音通信核心技术内幕——CT Media 存储器扩展技术指南。计算机电话（Computer Telephony：CT）和相关的计算机电话服务软件开发包使得在客户机/服务器环境中开发计算机电话应用程序成为可能。本书针对 CT Media 存储扩展技术讨论了 CT Media 基础、CT Media 存储管理模型、建立开发环境、编写组织策略、编写存储策略、故障诊断、组织策略接口、存储策略接口和 KVset 符号等内容。

5．计算机语音通信核心技术内幕——语音通信技术开发指南。介绍了 CT Access 环境下的语音通信服务的技术开发。

全书由 9 章及 6 个附录组成。第一章是计算机电话访问 CT Access 环境的概述；第二章概述了 CT Access 语音消息服务；第三章介绍了使用语音消息服务对语音文件进行播放和录制；第四章阐述了状态信息的获取；第五章概述语音文件的编辑；第六章介绍了提示生成器的使用；第七章是语音消息函数的概要；第八章提供了按字母顺序排列的函数参考；第九章提供了 CT Access 语音消息服务的演示程序和工具。在附录中还介绍了术语表；错误,事件和条件码；CT Access 函数的参数；VOX 文件格式；提示生成器和编码信息，并给出了技术支持的联系方法。

本书介绍了网络通信工程方面的最新技术，内容新颖、精炼、实用性强，既是网络集成工程师、语音工程师、网络通信工程人员对 CT Access 环境下的语音通信服务技术开发的必备技术参考书，也是高校相关专业师生教学、自学参考书和科研机构、科技图书馆的馆藏图书。

6．计算机语音通信核心技术内幕——CT Connect：C 程序设计指南。本书是 CT Connect（CTC）编程参考手册。CT Connect 是 Dialogic CT 分部研制开发的一套用于呼叫控制的软件。CT Connect 的服务器软件部分是基于 Windows NT 或 SCO UNIX 的，它通过与交换机的连接完成复杂的呼叫控制及监视功能。CT-Connect 的服务器软件部分完成与多种交换机的 CTI(Computer Telephony Integration) Link 的通信。它可将不同交换机的 CTI Link 的不同的协议及消息映射为同样的基于 CSTA 的消息，并管理服务器及相应应用程序间电话服务请求及状态消息的交换。通过 CT Connect，OEM 厂家、应用程序开发商及集成商很容易就可以在他们的应用程序中完成完善的电话路由及监视功能。

本书主要面向需要编写 CTC 应用程序的程序员。有两种编写 CTC 应用程序的编程界面：C 语言格式应用程序接口和 JAVA 语言的应用程序接口。本书是专门针对 C 语言格式编程接口进行描述的。

全书共分为两大部分，第一部分三章，具体对 CTC 应用程序接口例行程序的机制、数据结构和多线程设计进行了描述，给出了每个可调用例行程序参数和用法的具体讲解，并针对返回的错误信息进行了说明。第二部分附录，针对不同厂商生产的交换机描述了特定类型交换机的例行程序。首先对通用例行程序进行了概括说明，然后分别针对 CSTA 交换机、朗讯 DEFINITY 交换机和 Nortel Meridian 交换机进行了指定例行程序的详细说明。

全书层次清晰，内容详尽，简洁明了，适合于有一定编写电话应用程序经验的程序员在编写 CTC 应用程序时参考。

7．计算机语音通信核心技术内幕——GDK 3.2 程序设计。本书讨论 GammaLink Developer's Kit(GDK)的编程。GDK 3.2 是 Dialogic 最新软件开发包之一。Dialogic 软件开发包为应用系统的开发提供了完整的语音处理开发环境，它的运行稳定性以及丰富的功能最适合计算机语音处理系统。规模的伸缩性使得同一应用程序既适用于小系统，也适用于大系统。完整的软件开发包包括函数库、驱动程序和固件，以及附加的应用程序，如固件下载程序和安装程序等。软件开发包为不同种类的 Dialogic 产品提供了完整的集成开发环境，包括语音处理、传真、文本语音转换、语音识别、多方会议、交换以及不同的电话网络接口。由于这些功能均使用相同的程序设计风格，以及共享许多相同的函数调用，因此，能使应用系统天衣无缝地集成以上功能。所有 Dialogic 板卡均使用同一个下载程序。Dialogic 软件开发包的可靠性在世

界各国都得到证实，且得到越来越多用户的支持。

本书详细介绍了 GammaLink 传真系统的结构、编程模型、相关应用编程接口（API）函数调用及传真应用编程方面的指令，适用于在 Mcrosoft Windows NT 平台上进行传真软件和语音软件的应用程序开发。本书共分 8 章，分别介绍了传真技术和 GammaLink 传真系统的结构，描述配置命令和队记录编程，编程模型、PEB 与 SCbus 下的 API 编程，以及传真状态文件等内容。

本丛书集中讨论了 Dialogic 公司 CT 核心技术，反映了 90 年代末、21 世纪初 CT 技术的最新结果，内容定位与国内外技术和产品市场同步，技术内涵高、指导性强，特别是从事语音、传真、数据、语音识别、声音合成、互联网电话和呼叫中心管理的广大开发与编程人员、技术支持和管理与维护人员重要技术参考书，同时也是高等院校相关专业师生教学、自学参考书和国内科研院所各图书馆重要馆藏图书。

藉本丛书出版之际，特别感谢 Dialogic 公司副总裁恰克·霍斯先生，Dialogic 公司通信产品部资深经理霍华德·巴勃博士，本丛书就是在他们的大力帮助和协调下才得以完成。感谢 Dialogic 公司产品经理约翰·拉道博士、Dialogic 公司 CT 部资深经理蒂恩·却伦波博士、MIT CT 实验室主任马克·威尔斯博士、MIT CT 实验室资深研究员托蒂·鲍勃博士，以及资深记者阿道夫·吉尔先生，由于他们的技术指导和全力参与，本丛书才得以及时完稿。还要感谢黄太成、龙启铭、陆卫民、张中民、李国华、柴文强等，是他们夜以继日的辛勤劳动，使本丛书及时面市。真诚感谢参与本丛书编写的全体专家和技术人员，以及编辑、美工设计人员和录排人员、光盘制作人员等，是他们的加班、加点、忘我的工作，才使本丛书如期付梓出版。

因出版时间紧迫，书中错误在所难免，敬请读者谅解，并请拨冗指正，以期再版时修订。

21 世纪计算机语音通信开发技术丛书编委会

2000 年 6 月

目　录

第一章　管理

1.1　导言

本部分 API 规定的函数允许系统管理员与应用程序开发人员建立自定义的管理应用程序。

本部分 API 提供如下的函数功能性:

- 配置文件的管理
- CT Media 服务管理
- 有关服务供应商的信息
- 类属管理命令的处理
- 统计数据的收集

1.2　函数表

注释　有关全部函数的完整资料, 参阅 1.5 节。

1.2.1　配置文件管理函数

这些函数用于在 CT 服务器的配置数据库中建立和管理配置文件。

函数名称	说明
CTadm_AddProfile()	添加一个配置文件
CTadm_ExportProfile()	把一个配置文件导出到一个文件
CTadm_GetProfileInfo()	获取有关配置文件的信息
CTadm_ImportProfile()	从一个文件导入到一个配置文件中
CTadm_RemoveProfile()	删除一个配置文件
CTadm_UpdateProfile()	更新一个配置文件

1.2.2　CT Media 服务管理函数

这些函数用于在 CT 服务器的配置文件数据库中建立和管理 CT Media 服务项目。

函数名称	说明
CTadm_AddService()	添加一条 CT Media 服务项目到数据库中
CTadm_GetServiceInfo()	获取有关 CT Media 服务的信息
CTadm_GetServiceState()	获取一个或多个 CT Media 服务状态的信息
CTadm_RemoveService()	从数据库中删除一条 CT Media 服务项目
CTadm_StartService()	启动一个 CT Media 服务
CTadm_StopService()	停止一个 CT Media 服务
CTadm_UpdateService()	更新一个 CT Media 服务

1.2.3 服务供应商信息函数

服务供应商信息函数提供当前运行于 CT 服务器上的服务供应商的信息。

函数名	说明
CTadm_GetSPInfo()	获取服务供应商信息

1.2.4 统计函数

函数名	说明
CTadm_GetCounterInfo()	获取统计计数器的名字
CTadm_GetcounterValues()	获取统计计数器的值

1.2.5 类属管理函数

类属管理函数提供了访问 CT 服务器上其它服务程序的一种类属的机制。

函数名	说明
CTadm_DoAdmin()	执行一条类属的管理命令
CTadm_SetTargetServerLevel()	改变 CT 服务器的目标级
CTadm_GetServerLevel()	返回 CT 服务器当前的目标级

1.3 定 义

1.3.1 错误

下表列出的所有错误均属于 CT 类型错误。

注释 本部分 API 中可通过函数返回的其它错误可参阅 10.3.4 节。

错误代码名	说明
CT_errorBadSession	传送了一个错误的会话 ID
CT_errorBadKVSet	传送了一个错误的 KVSet
CT_errorCantOpenFile	配置文件无法打开
CT_errorComms	客户机/服务器通信发生错误
CT_errorExists	配置文件名已存在
CT_errorExportError	配置文件无法存储
CT_errorFileNotFount	找不到配置文件
CT_errorNoMem	内存分配发生错误
CT_errorNotFound	找不到合适的数据库入口
CT_errorTooManyProfs	配置文件数达到极限（99，999，999）

1.3.2 服务程序登录开关

下表所列开关用于创建或编辑 CT Media 服务程序，它们包含在可执行服务程序的命令行中，或在 DLL 服务程序中充当参数。

开关	说明
-F，/F	将该服务程序的所有失败或致命的错误输出到 *<CTMSLogDir>\fileName* 中
-f，/f	将该服务程序的所有失败或致命的错误输出到 *cout* 中。本在资源开发者调试服务程序时是非常有用的。选项 注释 为了查看输出结果，用户必须 · 转到 Settings|Control Panel|Services (1) 选择 "CTMS Server" (2) 在 "Log On As" 区域中，选择 "Allow Service toInteract with Desktop" · 在控制窗口中运行该服务程序

注释 <CTMSLogDir>是 CMTS 日志文件路径，该路径由 NT 注册表中适当的键值指明，如果找不到键值，<CTMSLogDir>缺省为.\。

1.3.3 函数返回状态定义

标准的 CTstatus 值适用于本部份 API 函数中。参阅 10.3.3 节。

1.4 管理 API 示例程序

下列程序及其描述文件存放于\CTServer\client\sdk\ex_code\Ref 目录中。

directory.

```
/*****************************************************************************
 *
 *      Copyright (c) 1998 Dialogic Corp.
 *      All Rights Reserved
 *
 *      THIS CODE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR
 *      IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE
 *      IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR
 *      A PARTICULAR PURPOSE.
 *
 *      THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF Dialogic Corp.
 *      The copyright notice above does not evidence any actual or
 *      intended publication of such source code.
 *
 *****************************************************************************/

/*****************************************************************************
 *      MODULE:  ct_admin.c
 *     VERSION:  1.1
 *
 * DESCRIPTION:  Administration API demonstration program
 *               This program demonstrates how to use the
 *               CT Media Administration API functions
 ************/*****************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <ct_apps.h>

/* Required for Sleep function   */
#define WIN32_LEAN_AND_MEAN
#include <windows.h>

/* Function Prototypes */
void     Initialize();
void     Shutdown();

void     CreateSession( CTses_ct *hpSession, char *profileName );
void     DestroySession( CTses_ct hSession );
```

```
void     CreateKVSet( CTkvs_ct *phKVS );
void     PutString( CTkvs_ct hKVS, CTsymbol key, char* string );
void     PutSymbol( CTkvs_ct hKVS, CTsymbol key, CTsymbol symbol );
CTuint   GetSymbolArray( CTkvs_ct hKVS, CTsymbol key, CTsymbol **array );
CTuint   GetStringArray( CTkvs_ct hKVS, CTsymbol key, char ***array );
CTuint   GetKVSetArray( CTkvs_ct hKVS, CTsymbol key, CTkvs_ct **array );
void     DestroyKVSet( CTkvs_ct hKVS );

void     GetCounterInfo(CTses_ct hSession,
                        CTsymbol asymManagers[],
                        CTuint   *puiNumManagers);

void LoadTable( CTses_ct hSession );
void UnloadTable( CTses_ct hSession);

void     GetSymbolName( CTses_ct hSession ,
                        CTsymbol symbol,
                        char* pBuff,
                        const int iBuffLength ) ;

void     GetCounterValues( CTses_ct  hSession,
                           CTsymbol  asymManagers[],
                           CTuint  numManagers);

#define MY_SERVICE_NAME          "example_service"
#define MY_PROFILE_NAME          "example_profile"
#define MY_PROFILE_CLASS_NAME    "example_class"

/*+=========================================================================
*         Name: main
*  Description: Executes all Administration APIs. This test requires a
*               running server in order to work properly.
*-=========================================================================*/
int main()
{
   CTkvs_ct    kvsEvent;
   CTstatus    status;
   CTtranInfo  tranInfo;
   CTerror     error;

   CTses_ct    adminSession;
   CTkvs_ct    profile;
   CTkvs_ct    dataKVSet;
   CTint       serverLevel=-1;
   CTsymbol    *stateArray;
   char        **nameArray;
   CTkvs_ct    *serviceProfileArray;
   CTkvs_ct    serviceProfile;
   CTuint      count;
   CTuint      i,j;

   /* declarations for Statistics functions  */
   CTkvs_ct       hCounterInfo, hManager, hCounterValues;
   CTsymbol       symManager, symCounter;
   CTvaltyp       type;
   CTuint         numCounters, numProcesses;
   CTsymbol       asymManagers[1024];
   CTuint         iNumManagers;
   const CTsymbol *symCounters;
   char           acManagerName[1024];
   char           acCounterName[1024];
   const char     **aszProcessNames;
   const CTkvs_ct *akvsManagerValues;
   const CTuint   *auiCounterValues;

   /* Initialize   */
   Initialize();

   /* Initialize some variables       */
   CreateKVSet(&profile);
   CreateKVSet(&kvsEvent);
   CreateKVSet(&dataKVSet);
   CreateKVSet(&serviceProfile);
   CTtran_Initialize(tranInfo,kvsEvent);

   /* Create an administration session. The empty profileName indicates
   /* that an administration session is required.
    */
   CreateSession( &adminSession,"" );
```

```
/*-------------------------------------------------------------
   Change the server level. This example forces the server to level 1.
 -------------------------------------------------------------*/
status = CTadm_SetTargetServerLevel(adminSession,
                                    1,      /* Level of Server */
                                    Admin_DLGC_NormalShutdown,
                                    &tranInfo,
                                    CT_modeSync);
assert( status == CT_statusOK );

/*-------------------------------------------------------------
   Wait for the server level to become 1  - demonstrates
   CTadm_GetServerLevel()
-------------------------------------------------------------*/
while ( serverLevel != 1 ) {
   status = CTadm_GetServerLevel(   adminSession,
                                    &serverLevel,
                                    &tranInfo,
                                    CT_modeSync);
   assert( status == CT_statusOK);
   printf( "Server Level is now %i\n", serverLevel );

   #ifdef WIN32
      Sleep(1000);
   #endif
}
printf("\n");

/*-------------------------------------------------------------
   Get state of the Core Service: dataKVSet contains the info
 -------------------------------------------------------------*/
status = CTadm_GetServiceState(  adminSession,
                                 "DLGC_CoreService",
                                 dataKVSet,
                                 &tranInfo,
                                 CT_modeSync);
assert( status == CT_statusOK);

/* Get array of states from the returned KVSet */
GetSymbolArray(   dataKVSet,
                  Admin_DLGC_ServiceState,
                  &stateArray);

/* indicate results  */
if ( stateArray[0] == Admin_DLGC_ServiceStateIdle ) {
   printf( "The Core Service is idle\n" );
} else {
   printf( "The Core Service is not idle\n" );
}
printf("\n");


/*-------------------------------------------------------------
   Get general info on all services
-------------------------------------------------------------*/
status = CTadm_GetServiceInfo(adminSession,
                              "",
                              dataKVSet,
                              &tranInfo,
                              CT_modeSync);
assert( status == CT_statusOK);

/* Get array of service names    */
count = GetStringArray( dataKVSet,
                        Admin_DLGC_ServiceName,
                        &nameArray );

/* Get array of profiles     */
GetKVSetArray( dataKVSet,
               Admin_DLGC_ProfileData,
               &serviceProfileArray);

/* Copy service profile number 0 and save for later   */
status = CTkvs_Copy( serviceProfile, serviceProfileArray[0], &error );
assert( status == CT_statusOK);

printf( "The current Services are:\n" );
for ( i=0; i<count; i++ ) {
   printf( "%s\n", nameArray[i] );
}
printf( "\n" );


/*-------------------------------------------------------------
   Stop the Core Service
-------------------------------------------------------------*/
status = CTadm_StopService(   adminSession,
                              "DLGC_CoreService",
                              Admin_DLGC_NormalShutdown,
                              &tranInfo,
                              CT_modeSync);
assert( status == CT_statusOK);
```

```
/*-------------------------------------------------------------
   Start the Core Service
-------------------------------------------------------------*/
status = CTadm_StartService(  adminSession,
                              "DLGC_CoreService",
                              &tranInfo,
                              CT_modeSync);
assert( status == CT_statusOK);


/*-------------------------------------------------------------
   Add a service - use the profile saved earlier and change the name
-------------------------------------------------------------*/
status = CTadm_AddService( adminSession,
                           MY_SERVICE_NAME,
                           serviceProfile,
                           &tranInfo,
                           CT_modeSync);
assert( status == CT_statusOK);

/*-------------------------------------------------------------
/* Update the service - use the profile saved earlier. This won't,   */
/* in fact, change the profile because 'serviceProfile' has not      */
/* changed.                                                          */
-------------------------------------------------------------*/
status = CTadm_UpdateService( adminSession,
                              MY_SERVICE_NAME,
                              serviceProfile,
                              &tranInfo,
                              CT_modeSync);
assert( status == CT_statusOK);


/*-------------------------------------------------------------
   Remove the service
-------------------------------------------------------------*/
status = CTadm_RemoveService( adminSession,
                              MY_SERVICE_NAME,
                              &tranInfo,
                              CT_modeSync);
assert( status == CT_statusOK);


/*-------------------------------------------------------------
   Add a profile
-------------------------------------------------------------*/
status = CTadm_AddProfile(adminSession,
                          MY_PROFILE_NAME,
                          MY_PROFILE_CLASS_NAME,
                          profile,
                          &tranInfo,
                          CT_modeSync);
assert( status == CT_statusOK);


/* -------------------------------------------------------------
   Update a profile - again the profile is not really updated since
   'profile' is unchanged.
-------------------------------------------------------------*/
status = CTadm_UpdateProfile( adminSession,
                              MY_PROFILE_NAME,
                              MY_PROFILE_CLASS_NAME,
                              profile,
                              &tranInfo,
                              CT_modeSync);
assert( status == CT_statusOK);

/*-------------------------------------------------------------
   Remove a profile
-------------------------------------------------------------*/
status = CTadm_RemoveProfile( adminSession,
                              MY_PROFILE_NAME,
                              MY_PROFILE_CLASS_NAME,
                              &tranInfo,
                              CT_modeSync);
assert( status == CT_statusOK);

/*-------------------------------------------------------------
   Get general info on all application profiles
-------------------------------------------------------------*/
```

```
status = CTadm_GetProfileInfo(adminSession,
                              "",
                              "Application",
                              CT_boolFalse,
                              dataKVSet,
                              &tranInfo,
                              CT_modeSync);
assert( status == CT_statusOK);

/* Get array of profile names from returned KVSet       */
count = GetStringArray(dataKVSet,
                       Admin_DLGC_ProfileName,
                       &nameArray);

printf( "The current Application Profiles are:\n");
for ( i = 0; i < count; i++) {
   printf( "%s\n", nameArray[i] );
}
printf("\n");

/*-----------------------------------------------------------------
   Get list of all Service Providers
-----------------------------------------------------------------*/
status = CTadm_GetSPInfo(adminSession,
                         "",
                         dataKVSet,
                         &tranInfo,
                         CT_modeSync);
assert( status == CT_statusOK);

/* Get array of profile names from returned KVSet           */
count = GetStringArray( dataKVSet,
                        Admin_DLGC_SPName,
                        &nameArray);

printf("The current Service Providers are:\n");
for ( i = 0; i < count; i++ ) {
   printf("%s\n", nameArray[i]);
}
printf("\n");


/*-----------------------------------------------------------------
   Export a profile to a file - the WinVoteProfile must exist in order
   for this call to work.
-----------------------------------------------------------------*/
status = CTadm_ExportProfile(adminSession,
                             "profile.ief",
                             "WinVoteProfile",
                             "Application",
                             &tranInfo,
                             CT_modeSync);
assert( status == CT_statusOK);

/*-----------------------------------------------------------------
   Import a profile from a file - the export profile must have worked
   or this call will fail.
-----------------------------------------------------------------*/
status = CTadm_ImportProfile(adminSession,
                             "profile.ief",
                             "WinVoteProfile2",
                             "Application",
                             &tranInfo,
                             CT_modeSync);
assert( status == CT_statusOK);

/* Delete the imported profile         */
status = CTadm_RemoveProfile(adminSession,
                             "WinVoteProfile2",
                             "Application",
                             &tranInfo,
                             CT_modeSync);
assert( status == CT_statusOK);


/*-----------------------------------------------------------------
   An example of calling DoAdmin - it will fail because the arguments
   are nonsensical
-----------------------------------------------------------------*/
status = CTadm_DoAdmin( adminSession,
                        "A_Service_Provider",
                        Any_ECTF_Null,
```

```
                        CT_kvsNull,
                        dataKVSet,
                        &tranInfo,
                        CT_modeSync);
assert( status != CT_statusOK); /* Failure is expected

/*-------------------------------------------------------------
   Change the server level back to 5.
-------------------------------------------------------------*/
status = CTadm_SetTargetServerLevel(   adminSession,
                                       5,
                                       Admin_DLGC_EmergencyShutdown,
                                       &tranInfo,
                                       CT_modeSync);
assert( status == CT_statusOK );


/* Wait for the server level to become 5       */
while ( serverLevel != 5 ) {
   status = CTadm_GetServerLevel(   adminSession,
                                    &serverLevel,
                                    &tranInfo,
                                    CT_modeSync);
   assert( status == CT_statusOK);
   printf( "Server Level is now %i\n", serverLevel );

   #ifdef WIN32
      Sleep(1000);
   #endif
}
printf("\n");

/* ********* Statistics examples          ******** */

/* Load symbol table for translation purposes      */
LoadTable( adminSession );

/* Get the counter information for                 */
/* use in the GetCounterValues function            */

 /* We create the KVSets needed and put in the necessary information*/
status = CTkvs_Create( &hCounterInfo, &error );
assert( status == CT_statusOK);

// Get counter managers
status = CTadm_GetCounterInfo(   adminSession,
                                 hCounterInfo,
                                 &tranInfo,
                                 CT_modeSync);
assert( status == CT_statusOK);

/*-------------------------------------------------------------------------
   Display the counter manager symbols to stdout
 -------------------------------------------------------------------------*/
printf("COUNTER MANAGERS\n");
symManager = KVS_ECTF_FirstKey;
iNumManagers = 0;
while ( CTkvs_GetNextKey( hCounterInfo, &symManager, &type, &error )
        != CT_statusWarning ) {  /* while not at end of KVSet   */
   /* translate and print out Manager name   */
   GetSymboolName( adminSession,
                   symManager,
                acManagerName,
                sizeof(acManagerName));
printf("     %s\n", acManagerName);

/* Extract the Manager information  */
status = CTkvs_GetKVSet( hCounterInfo, symManager, &hManager, &error);
assert( status == CT_statusOK);

/* Extract the Counter array from the Manager KVSet   */
status = CTkvs_GetSymbolArray(hManager,
                              Admin_DLGC_StatisticsCounters,
                              &symCounters,
                              &numCounters,
                              &error);
assert( status == CT_statusOK);
/* translate and print each Symbol from the array  */
for(i=0;i<numCounters;i++) {
   GetSymbolName( adminSession,
                  symCounters[i],
                  acCounterName,
                  sizeof(acCounterName));
   printf("          %s\n", acCounterName);
}
```