WebChip 系列资料

WebChipServer™

用 户 手 册

P & S 数 据 通 信 公 司

二 零 零 一 年 七 月

目　录

第一部分
WebChipServer™ 用户安装手册
（适用于 Win98\WinNt\Win2000）

1.1 前言

欢迎你使用 WebChipServer™ 评估软件，我们希望它能给你带来一些全新的感受。本手册将介绍什么是 WebChipServer™ 以及如何安装 WebChipServer™。

WebChipServer™ 是一个网关软件。用此软件，客户端可以通过任何浏览器监视、管理、控制基于 WebChip™ 系列产品做的任何一台智能设备。

安装 WebChipServer™ 包括:
安装前的准备，即
提示你在安装 WebChipServer™ 之前所应做的事情和准备的材料;
具体安装 WebChipServer™ 的细节部分。

1.2 安装前的准备

- 一台计算机
  一CPU 400MHz 及以上
  一内存 128M 及以上
  一硬盘 100M 自由空间
  一CDROM 驱动器
  一已安装上 Windows 98 或 WindowsNT4.0 或 Windows2000
- PSEA00A Evaluation kit

1.3 安装 WebChipServer™

PSEA00A Evaluation kit 附带的光盘的内容
WebChipServer™ 目录下的是 P&S DataCom 公司的 WebChipServer™ 安装软件。

1.3.1 安装 JRE

WebChipServer™ 是由纯 java 编写的软件，它是运行在 java 虚拟机之上的。在安装 WebChipServer™ 之前请先安装 JRE, 版本要求在 1.3 及以上。安装 JRE 步骤如下:

1. 从 http://java.sun.com/j2se/1.3/jre/download-windows.html 下载 JRE1.3 及以上的 International version;

2. 运行下载的 JRE1.3 执行程序即可安装 JRE1.3;

3. 注册表键值的修改。

- 若你所用的 Windows 操作系统是中文操作系统，则应修改系统注册表的以下键值

应把 HKEY_LOCAL_MACHINE\Software\JavaSoft\Java 插件 修改为:
HKEY_LOCAL_MACHINE\Software\JavaSoft\Java Plug-in
应把 HKEY_LOCAL_MACHINE\Software\JavaSoft\Java 运行时环境 修改为:
HKEY_LOCAL_MACHINE\Software\JavaSoft\Java Runtime Environment

用以下步骤可修改上面提到的注册信息。在中文 Windows 开始菜单中选择运行，键入 regedit 并按回车，在注册表编辑器应用中点击 HKEY_LOCAL_MACHINE、Software、JavaSoft、用鼠标右键点击 Java 插件用重命名把它修改成 Java Plug-in，用同样的方法把 Java 运行时环境修改为 Java Runtime Environment。

- 若你用的是英文版的操作系统则不用修改这些注册表。

安装完 JRE 后就可安装并运行 WebChipServer™ 了。

1.3.2 安装 WebChipServer™

可按以下步骤安装 WebChipServer™:
在光盘的 WebChipServer™ 目录下直接运行 setup.exe 文件;
运行 setup.exe 后，界面显示所图 1 所示。

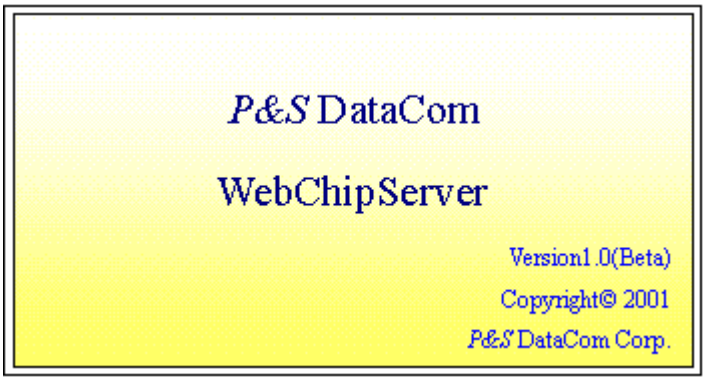

图 1

过数秒后显示画面如图 2 所示。

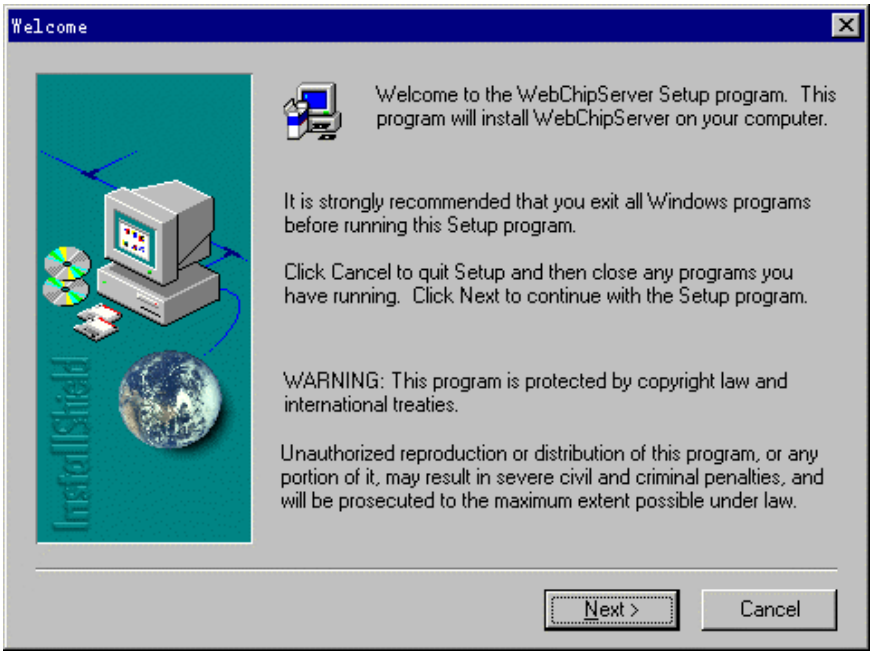

图 2

若要继续安装请点击 Next 按钮，若要退出请点击 Cancel 按钮。按下 Next 按钮后显示如图 3 所示的界面。

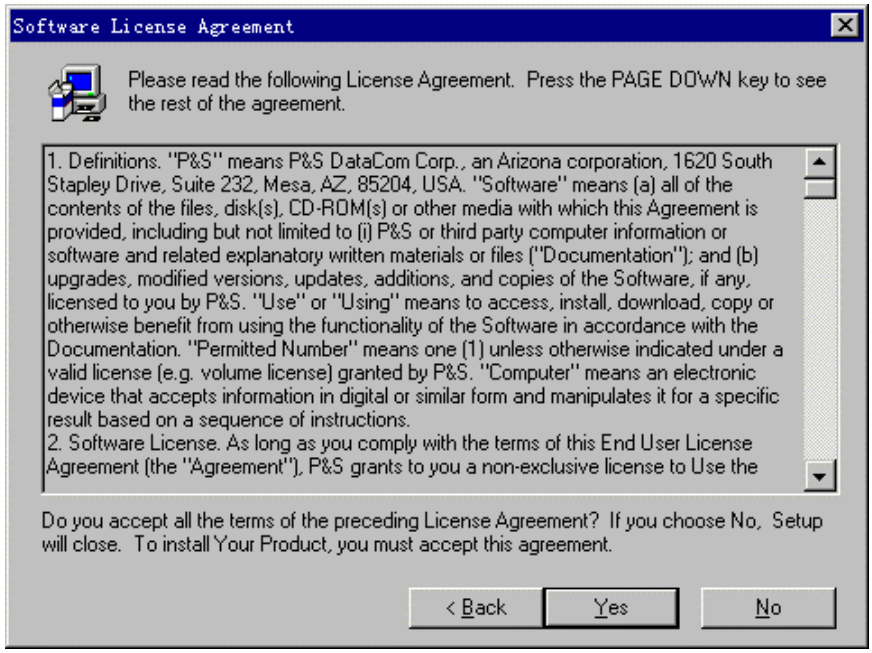

图 3

若接受许可时请按 Yes 按钮，若不接受许可按 No 退出安装。按 Yes 按钮后出现的界面如图 4 所示。

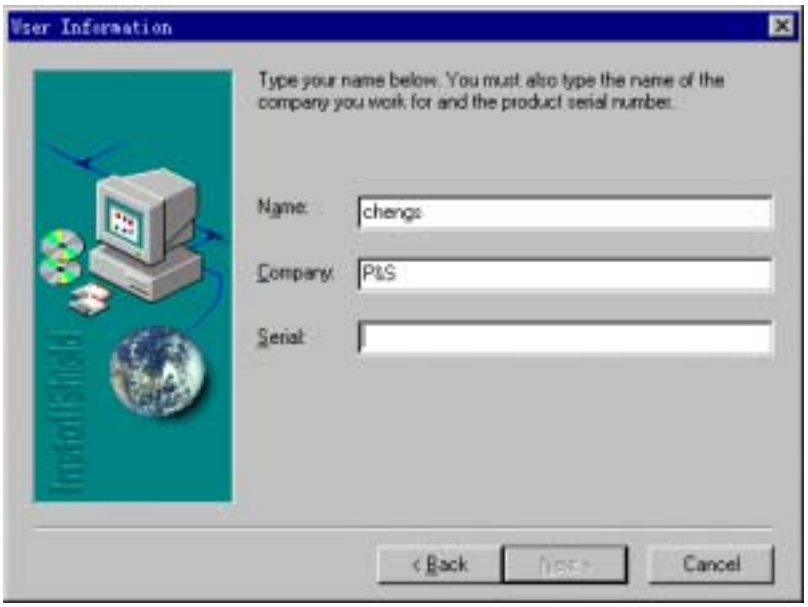

图 4

输入正确的序列号后按 Next 按钮，若退出请按 Cancel 按钮。点击 Next 按钮后界面如图 5 所示。

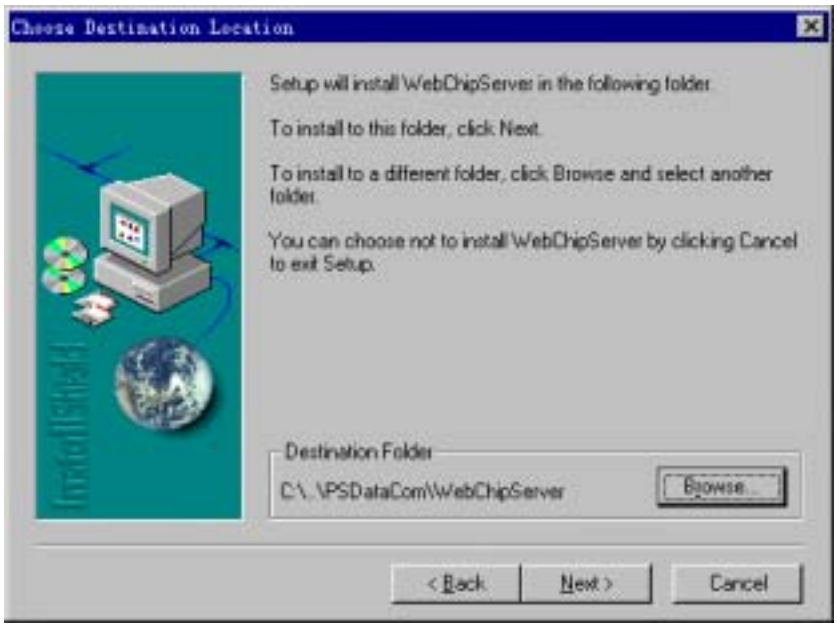

图 5

若要改变目录请按 Browse 按钮进行目录定位或修改，确定安装目录后按 Next 按钮继续。点击 Next 按钮后界面如图 6 所示。

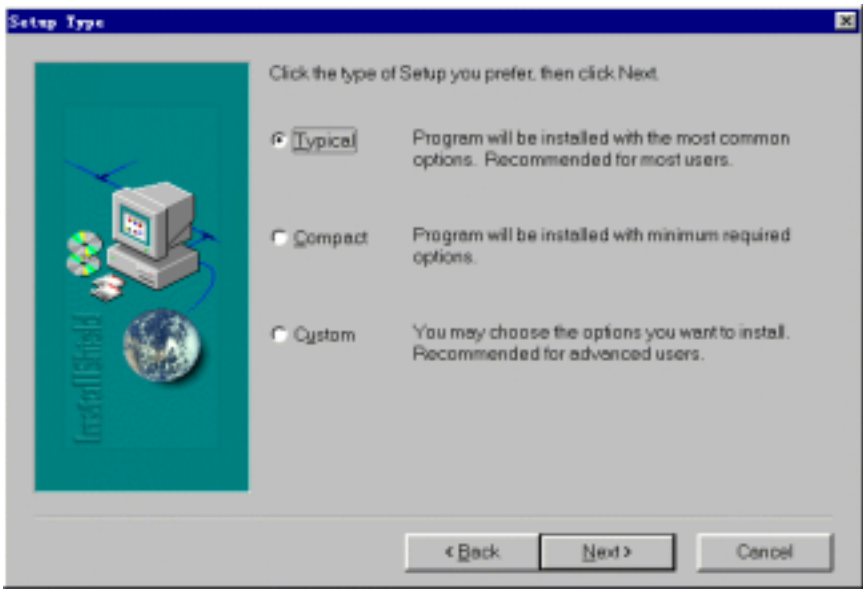

图 6

该界面有三种安装选择，Typical 和 Compact 方式安装的结果是一样的，Custom 方式可选择安装程序组件。一般选择 Typical 安装方式，点击 Next 按钮后出现如图 7 所示的界面。

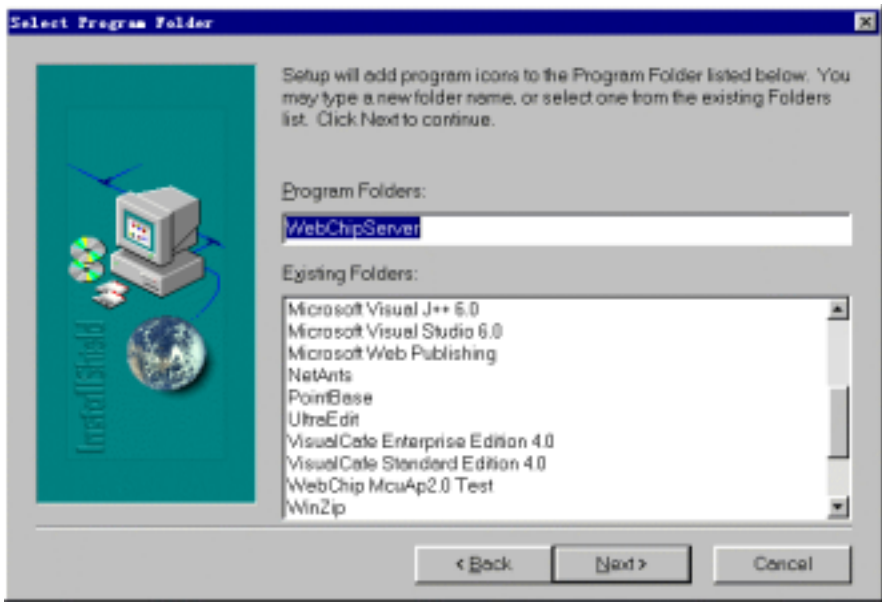

图 7

在 Program Folders 中输入安装的程序组，缺省的程序组为 WebChipServer™。点击 Next 按钮后为安装程序解包和拷贝文件，安装结束后界面显示如图 8 所示。

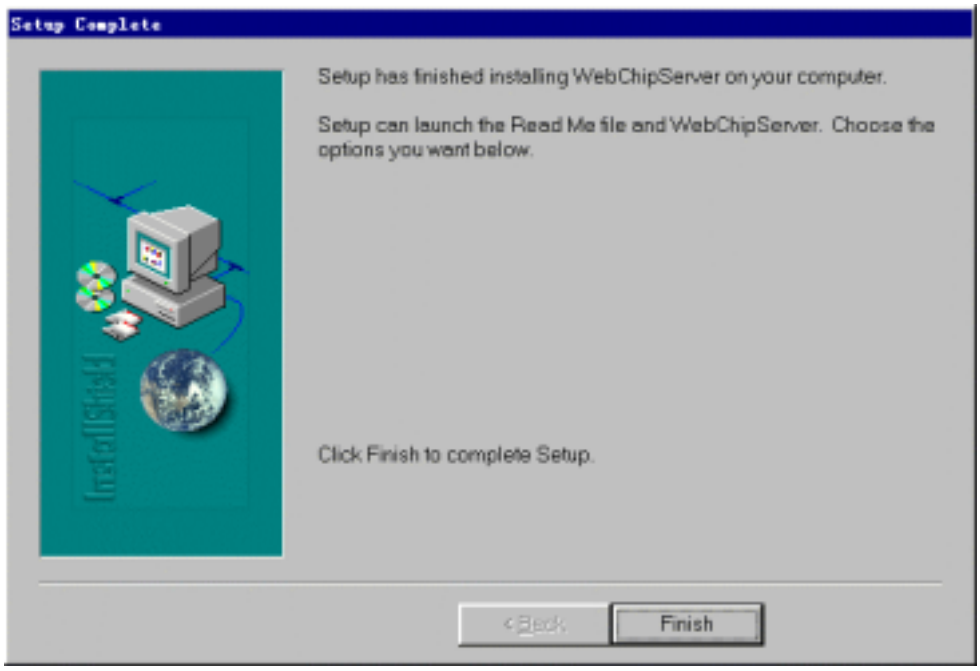

图 8

1.3.3 注意事项

WebChipServer™ 网关软件仅供用户评估或实验用，我们不能保证该软件可以用于任何其它环境。

第二部分
WebChipServer™ 网关组成介绍

2.1 前言

随着互联网的不断普及与发展，各种基于 Internet 的增值服务和特殊应用不断涌现。人们充分利用互联网的廉价，快捷以及覆盖范围广的特点，开发出满足自己特殊需要的应用系统，其中最近两年来发展起来的基于网络的（局域网，广域网和 Internet）智能设备互连技术的发展尤为引人注目，这种技术使得人们通过互联网络远程控制和监视各种智能设备（可以是基于 MCU 单片机的任何设备）成为可能。1999 年成立的国际性的行业组织 OSGi 为这种网络应用模式确定了技术框架，制订了服务网关的接口规范，这种网络应用的结构如图 9 所示。

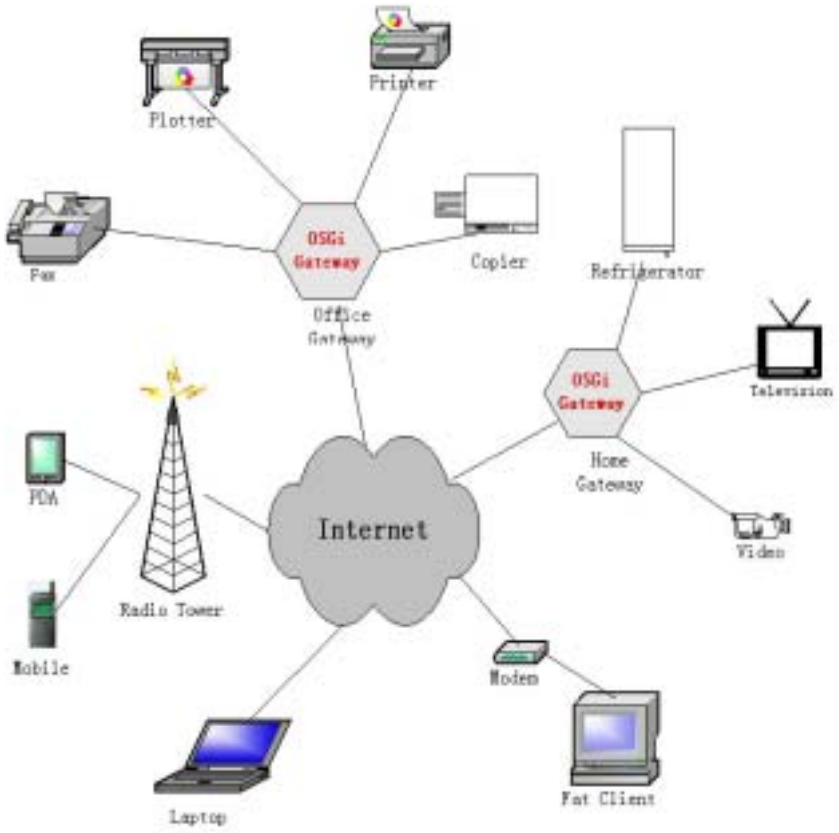

图 9 OSGi 网络应用模式

在这种应用模式下，各种智能设备可以通过 OSGi 网关相互通信，一个典型的应用情景是通过手机，PDA 等便携设备遥控和远程监视家庭中的各种家用电器。其中 OSGi 服务网关是这种应用模式的关键部分，OSGi 于 2000 年 5 月发布了 OSGi 服务网关技术规范 1.0 版本，在其中描述了一个 OSGi 服务器必须实现的服务以及每个服务为客户端提供的标准服务接口。目前有几个先行的厂商已经根据这个规范，开发出了自己的网关产品，其中 Prosyst 公司的 mBServer 是其中比较成熟的产品之一，到目前为止，mBServer 已经发展到 4.1 版本。而 WebChipServer™ 便是采用 mBServer4.1 服务网关作为应用平台，在网络环境下，实现远程监视和控制各种连接在 OSGi 服务器上的智能设备（WebChip™ 设备）。当然，WebChipServer™ 服务应用也可基于其它公司的符合 OSGi 规范的服务网关平台，甚至可以在非 OSGi 网关上实现，因此具有很大的灵活性。不过，考虑到应用系统开发的效率和系统的可移植性，在此采用 OSGi 服务器网关作为开发和应用平台。

2.2 WebChip™ 网络应用系统整体结构

基于 OSGi 服务网关的 WebChip™ 网络应用系统由三部分组成，它们分别是：智能设备，网关服务器和客户端应用。它们之间的关系如图 10 所示。

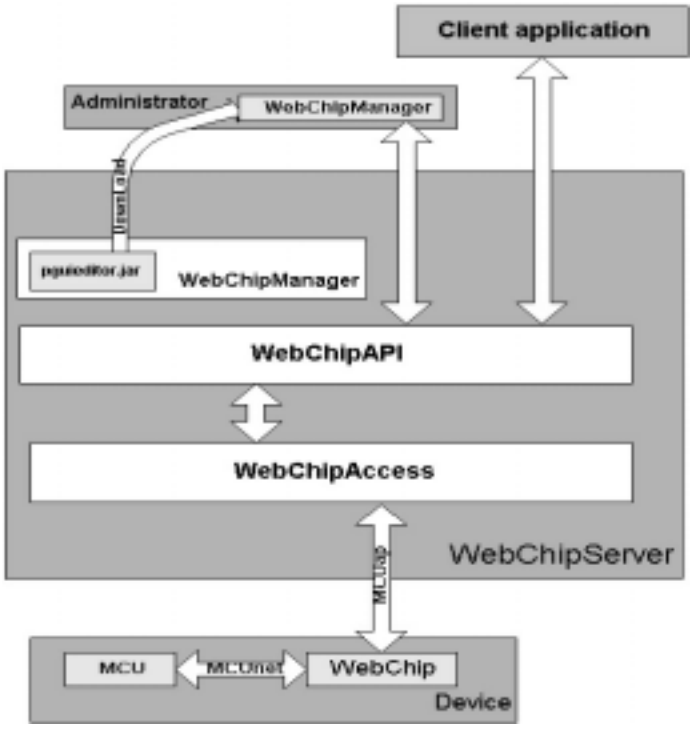

图 10 WebChip™ 网络应用系统结构图

WebChipServer™ 位于整个网络应用系统的中间，起到连接客户端应用程序和底层智能设备的桥梁作用，它运行在 OSGi 服务网关的框架（mBServer4.1）之中，主要由三个部分组成。

三个模块是：WebChipAccess、WebChipAPI、WebChipManage, 实现这三个模块功能的元素被打包到相应的 Bundle 文件中，Bundle 文件名称分别为：WebChipAccess.jar、WebChipAPI.jar、WebChipManager.jar。

这三个 Bundle 根据其间的调用关系，可以分为三个层次:

最底层为设备访问模块 WebChipAccess，它的主要功能是和智能设备的 WebChip™ 芯片进行通信，负责获取设备的常量以及变量的值，订阅设备的事件，执行设备的函数，存取设备的文件内容。智能设备（更具体地说是 Webchip™ 模块或芯片）和 WebChipAccess 模块之间的通信遵循 MCUap™1.0 协议，有关设备的属性，如常量，变量，事件，函数以及文件等定义，请参阅《MCUap™1.0 协议》。本模块发布服务 McuApAccessServiceImpl（该服务由接口 McuApAccessService 定义），该服务由它的上层模块 WebChipAPI 调用，用来访问设备的属性。

在 WebChipAccess 模块之上是用户接口模块 WebChipAPI，该模块是用户编程接口模块。该模块发布了服务 ChipAPIImpl（该服务的方法由接口 ChipAPI 定义） 为客户端应用程序（包括管理客户端应用程序）提供各种访问和控制设备的 API 方法，这些方法包括：连接设备方法，断开设备方法，读取设备的常量方法，读写设备变量方法，执行设备函数方法，读写设备文件方法，用户管理方法，设备管理方法，权限管理方法，服务器监视方法等。WebChipAPI 模块在运行过程中要调用 WebChipAccess 模块的 McuApAccessServiceImpl 服务，因此在运行本模块之前，必须先安装并运行 WebChipAccess 模块的 Bundle 文件。

与前两个模块不同，WebChipManage 模块并不是一个服务模块，它没有发布任何服务，虽然该模块的 Bundle 文件存放在 mBServer 服务器上,但它并没有注册服务供其它 Bundle 或客户端调用。从本质上讲，它是一个存放在服务器端的，与 Prosyst 公司的服务网关 mBServer 的管理工具 Administrator 集成的客户端应用模块。该应用要调用 WebChipAPI 模块提供的用户管理，设备管理以及权限管理方法来完成相应的管理功能。

2.3 WebChipServer™ 的运行环境

WebChipServer™ 是 OSGi 服务网关框架（Framework）mBServer 中的一个应用服务，它在 mBServer 4.1 版本上开发。在运行过程中要用到 mBServer 4.1 网关提供的一些基本功能和底层服务，并且这些服务之间根据其调用顺序，存在层次关系，底层的模块为上层模块提供服务，而上层模块又为更上层的模块提供服务。因此，要想正确地运行 WebChipServer™，除了提供必要的硬件资源（如 CPU、内存、硬盘资源等）和系统环境外（如适当的 OS 版本，以及 Java 运行环境），还需要适当的配置 mBSerevr4.1 的运行参数，按照正确的顺序安装并运行 WebChipServer™ 所依赖的底层服务。

2.3.1 硬件环境

运行在 mBServer 4.1 服务网关框架内的 WebChipServer™ 服务，对硬件的要求并不高，一般来讲，如果 CPU 的主频在 200MHz, 内存在 128M，硬盘有 50M 的自由空间，就可以运行，但为了获得较快的系统响应速度，建议运行该网关和 WebChipServer™ 的服务器的硬件配置为:

CPU:	主频在 400MHz 以上;
RAM:	128MB 以上;
硬盘:	100M 以上自由空间。

2.3.2 系统环境

mBServer 4.1 和 WebChipServer™ 均采用纯 Java 语言编写。由于 Java 语言本身所固有的跨平台特性，使得本系统可以运行在任何一种安装了 Java 虚拟机的操作系统平台上。

操作系统:	Windows 98 /Windows NT 4.0 / Windows2000 / Unix;
Java 运行时环境:	JRE 1.2 / 1.3;
OSGi 服务网关:	mBServer4.1。

2.3.3 应用平台环境

mBServer4.1 是一个遵循《OSGi Service Gateway Specification Release 1.0》规范的 OSGi 的服务网关，它实现了该规范中规定的所有基本服务。这些基本服务是运行第三方开发的增值服务以及 mBServer4.1 自身提供的高层服务所必不可少的，除了这些基本的服务之外，Prosyst 还提供了一些增强的服务，以便于第三方应用服务的开发。

mBServer4.1 中提供的服务位于不同的 Bundle 文件中，按照这些服务之间的调用关系，应该按照以下顺序安装并启动下面的 Bundle: init.jar、lib.jar、putil.jar、console.jar、registry.jar、log.jar、http.jar、devicem.jar、usermgr.jar、pmp.jar、telnet.jar、license.jar、pbloader。然后按以下顺序安装 WebChipServer™ 的三个 Bundle: WebChipAccess.jar 、 WebChipAPI.jar 、WebChipManager.jar。mBServer4.1 启动时，将按照 init.jar 包中文件 boot.ini 中的命令自动启动以上 Bundle。WebChipServer™ 在安装时已经配置好 boot.ini 文件，一般情况下，用户无需手工配置。下面的章节中，我们将比较详细的介绍 WebChipServer™ 的三个模块的功能和内部结构以及它们发布的服务和导入，导出的 Java 语言包（package）。

2.4 WebChipServer™ 功能模块介绍

2.4.1 设备访问模块（WebChipAccess）

设备访问模块 WebChipAccess 遵循 MCUap 协议与设备进行通信，它负责设备属性的读写操作，并将从设备中检索出来的设备属性存放在嵌入式数据库 PointBase 中，提供给 WebChipAPI 模块读写。它的基本工作过程是: 在连接设备的所有端口启动一个线程，不停的轮询（polling）连接到端口的设备，读取最新的设备属性，放入数据库中。客户端应用从数据库中读出最新的属性值，对于实时性要求比较高的属性，WebChipAccess 也提供了实时读属性的方法，同时，该模块也提供修改设备属性的方法，用于客户端对设备进行控制，其工作过程如图 11 所示。

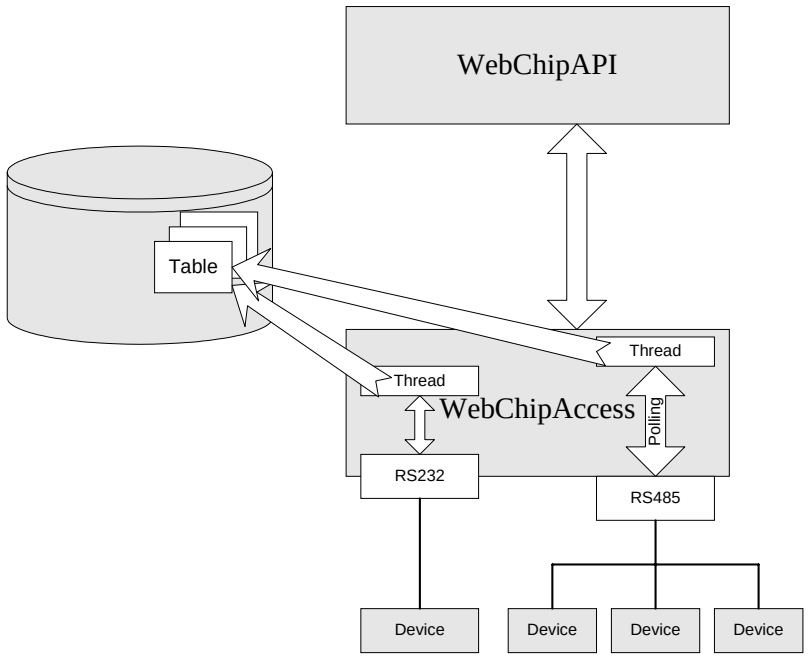

图 11 WebChipAccess 模块工作过程

设备的属性存放在 PointBase 嵌入式数据库 WebChip.dbn 中，与 WebChipAccess 有关的几个数据库表有：常量表、变量表、事件表、函数描述表、文件描述表、设备注册表。这些属性表分别存放设备相应的属性。

1. 常量表（DevicesConstants）

常量表用于存放设备的常量，该表中存放了所有设备的所有常量的描述以及常量的值。该表包括了设备标识符，常量名称，常量类型以及常量的值等字段。客户端应用可以通过 WebChipAPI 模块的常量访问方法读取常量。

2. 设备变量表（DevicesVariables）

变量表用于存放设备的变量，该表中存放了所有设备的所有变量的描述以及最新的变量的值。该表包括了设备标识符，变量名称，变量类型，变量属性，变量长度以及变量的值等字段。客户端应用可以通过 WebChipAPI 模块的变量访问方法读取和修改变量。

3. 设备事件表(DevicesEvents)

事件表用于存放设备的事件，该表中存放了所有设备的所有事件的描述以及事件最近一次触发所伴随的数据。该表包括了设备标识符，事件名称，事件类型，事件属性，事件数据的长度以及事件数据等字段。客户端应用可以通过 WebChipAPI 模块的事件访问方法查询、订阅和改写事件。

4. 设备函数描述表（DevicesFunctions）

设备函数描述表用于存放设备的函数描述，该表中存放了所有设备的所有函数的描述（函数名称，参数类型，返回值等）。该表包括了设备标识符，函数名称，输入参数类型，输入参数长度，输出参数类型，输出参数长度等字段。客户端可以通过 WebChipAPI 调用 WebChipAccess 的函数执行方法执行设备的函数，并获得返回结果。

5. 设备文件描述表(DevicesFiles)

设备文件描述表用于存放设备的文件描述，该表中存放了所有设备的所有文件的描述（文件名称，文件类型，文件属性，文件长度等）。该表包括了设备标识符，文件名称，文件类型，文件长度，文件属性等字段。客户端可以通过 WebChipAPI 模块调用 WebChipAccess 的文件访问方法读出文件的内容，或者修改文件的内容。

6. 设备注册表（DevicesRegsister）

设备注册表用于存放有关设备的信息，表中存放了所有设备的描述。该表包括了设备标识符，设备名称，访问路径，设备口令，用户连接计数等字段。该表由 WebChipAccess 模块在连接设备时使用，另外管理客户端使用该表来统计设备的连接计数。

除了上述的工作之外，该模块还提供设备的自动发现功能，实时和内存读设备属性功能，供 WebChipAPI 模块调用。所有这些服务通过接口 com.pns.webchip.McuApAccessService 发布，通过 mBServer4.1 的注册服务 Register 登记在服务器的注册数据库中。该接口的定义如下所示:

```
package com.pns.webchip;
public interface McuApAccessService
{
   public void AccessHelloWorld( );
      public boolean AccessDiscovery( String strComPort, String StrTelePhone );
      public boolean AccessConnectDevice( long LngDeviceID, String StrPWD, String StrTelePhone,
            String StrCommPort, int IntBaud );
         public boolean AccessDisConnectDevice( long LngDeviceID );
         public String AccessGetConnectState( String StrCommPort );
         public boolean AccessDeviceIsConnected( long LngDeviceID );
         public byte AccessGetDeviceCharacteristic( long LngDeviceID );
         public int AccessGetMaxRec( long LngDeviceID );
         public int AccessGetMaxSend( long LngDeviceID );
         public int AccessGetObjCount( long LngDeviceID );
         public short AccessGetProtocolVersion( long LngDeviceID );
      public byte[] AccessCallFunction( long LngDeviceID, String StrFunctionName, byte[]
            ABytParament, int IntReturn, byte BytParaType, byte BytReturnType );
```

```
    public byte[] AccessReadProperty( long LngDeviceID, String StrPropertyName, int IntOffset, int
            IntLen, byte BytDataType );
    public byte[] AccessReadMemoryProperty( long LngDeviceID, String StrPropertyName, int
            IntOffset, int IntLen );
    public boolean AccessWriteProperty( long LngDeviceID, String StrPropertyName, int IntOffset,
            byte[] ABytValue, byte BytDataType );
    public boolean AccessEnableEvent( long LngDeviceID, String StrEventName );
    public boolean AccessDisableEvent( long LngDeviceID, String StrEventName );
    public boolean AccessWritePropertyTable( long LngDeviceID, String StrSuperPWD, byte[]
            ABytData );
      public void AccessSetFrameInterval( int IntFrameInterval );
      public void AccessSetRecTimeOut( int IntTimeOut );
      public void AccessSetBackgroundTimeOut( int IntTimeOut );
      public void AccessGetReSendTimes( int IntReSendTimes );
      public int AccessGetFrameInterval( );
      public int AccessGetRecTimeOut( );
      public int AccessGetBackgroundTimeOut( );
      public int AccessGetReSendTimes( );
      public void AccessClose();
}
```

本模块的另一个主要功能是广播设备的事件，当 WebChipAccess 启动时，它将初始化事件表。若有用户订阅某个事件，它将注册该事件，并在事件发生时，向所有客户端广播该事件。如果客户端订阅了事件，就可以在事件监听器中处理这些事件。除了设备定义的事件之外，当任何设备与服务器断开时，WebChipAccess 还广播一个 Offline 事件。

发布的服务: com.pns.webchip.McuApAccessService
引用的服务: com.prosyst.mbs.services.pbloader.PointbaseService
com.prosyst.mbs.services.pmp.PMPServer
导出的包: com.pns.webchip
引入的包: com.prosyst.mbs.services.pbloader.jdbcdriver
com.prosyst.mbs.services.pbloader
com.prosyst.mbs.services.pmp
com.pointbase.jdbc
com.prosyst.util.event
javax.comm

2.4.2 用户接口模块（WebChipAPI）

用户接口模块的主要功能是为客户端应用程序提供编程接口(API),它位于设备访问模块 WebChipAccess 的上层。根据用户请求的操作，执行管理功能或者调用底层的服务来访问设备。本模块隔离了底层设备操作的复杂性，使得物理设备的操作对用户透明；本模块负责维护相关的数据库表，除了维护上述的六个表外，它还要维护用户表，权限表，为 WebChipManager 客户端提供服务；本模块注册服务名为 WebChipAPI.ChipAPI 的服务,该服务的接口定义了客户端访问设备和数据库的所有方法。接口定义如下:

```
package WebChipAPI;
import java.util.Vector;
public interface ChipAPI {
////////////////////////////Constant Access Interface//////////////////////
public Vector getAllConstants(long DeviceId);
public AnDeviceConstant getAnConstant(long DeviceId,String ConName);
////////////////////////////Variable Access Interface//////////////////////
public Vector getAllReadableVars(long DeviceId);
public AnDeviceVariable getAnReadableVar (long DeviceId,String VarName);
public String setAnWritableVar(long DeviceId,String VarName,int Offset, byte[]
VarValue,byte DataType);

////////////////////////////Event Access Interface/////////////////////////
public AnDeviceEvent getAnEvent(long DeviceId,String EventName);
public Vector getAllEvents(long DeviceId);
public String WriteEvent(long DeviceId,String EventName,int Offset, byte[]
EventValue,byte DataType);

////////////////////////////Device Connect Interface/////////////////////
public String DeviceConnect(String DeviceName,String UserName,String UserPassword);
public Boolean DeviceDisconnect(String DeviceName,String UserName,String
UserPassword);
public Vector getTotalConnection();
public Vector getUserConnection();

////////////////////////////Event Subscribe Interface/////////////////////
public Boolean SubscribeEvent(long DeviceId,String EventName);
public Boolean UnsubscribeEvent(long DeviceId,String EventName);
////////////////////////////Function Access Interface/////////////////////
public Vector getAllFunDescriptor(long DeviceId);
```

```
public FunctionDescriptor getAnFunDescriptor (long DeviceId,String FunName);
public FunctionParm ExeAnFunction(long DeviceId,String FunName,byte[] FunParm,int IntReturnLen, byte BytParaType, byte BytReturnType );
////////////////////////////File Access Interface//////////////////////
public Vector getAllFileDescriptor(long DeviceId);
public FileDescriptor getAnFileDescriptor (long DeviceId,String FileName);
public FileContent getFileContent(long DeviceId, String FileName, int Offset, int Len);
public String WriteFile(long DeviceId,String FileName,int offset,byte[] FileContent);

//////////////////////////// User Manager Interface ////////////////////////

public Vector getAllUsers(String arg);
public Boolean deleteAnUser(String UserName);
public Boolean addAnUser(String UserName,String Password,boolean Valid);
public Boolean UpdateAnUser(String OldUserName,String NewUserName,String NewPassword,boolean NewValid);
//////////////////////////// Device Register Manager Interface ////////////////////////

public Vector getAllDevices();
public Boolean deleteAnDevice(long deviceId);
public Boolean addAnDevice(long DeviceId, int ServerId,String DeviceName,String DevicePassword,String AccessPath,int TUpdate);
public Boolean UpdateAnDevice(long OldDeviceId,long NewDeviceId,int ServerId,String DeviceName,String DevicePassword,String AccessPath,int TUpdate);

//////////////////////////// user/Device privilege Manager Interface ////////////////////////

public Vector getAnUserPrivileges(String UserName);
public Boolean deleteAnPrivilege(String UserName,long DeviceId);
public Boolean addAnPrivilege(String UserName,long DeviceId,String DeviceName);

//////////////////////////// Real Time Read & Memory Read Property Interface //////////////////////

public ByteArray RealTimeReadProperty(long DeviceID, String PropertyName, int Offset, int Len, byte DataType );
public ByteArray MemoryReadProperty(long DeviceID, String PropertyName, int Offset, int Len );
}
```

有关每个方法的具体使用，请参照《WebChipServer™ 用户开发手册》。

除了为客户端提供以上的设备访问和数据库表维护的方法外，为了能够在管理客户端监视用户的连接状况，本模块注册了两个事件：UserConnectEvent，UserDisConnectEvent。它们分别于用户连接和断开时被触发，管理客户端将捕获这两个事件来统计设备的连接计数。本模块提供的方法可以分成两类:

第一类为设备属性访问类。该类方法又可分为数据库访问类和设备访问类，对于一些实时性要求不高的属性，通过数据库访问类方法从数据库中读取，如常量，变量，最近一次触发的事件等；而设备访问类方法则要调用底层 WebChipAccess 提供的方法，直接操作设备，对于实时性要求比较高的读属性操作以及修改属性的操作，调用这类方法。

第二类方法为系统管理方法。这些方法主要访问三个与系统管理有关的数据库表，它们是用户表，设备注册表和权限表，这类方法主要提供给管理客户端使用。

发布的服务: WebChipAPI . ChipAPI
引用的服务: com.prosyst.mbs.services.pbloader.PointbaseService
com.prosyst.mbs.services.pmp.PMPServer
com.pns.webchip.McuApAccessService
导出的包: WebChipAPI
引入的包: com.prosyst.mbs.services.pbloader.jdbcdriver,
com.pns.webchip,
com.prosyst.mbs.services.pmp,
com.prosyst.mbs.services.pbloader,
com.pointbase.jdbc,
com.prosyst.util.io,
com.prosyst.util.event

2.4.3 服务器管理模块（WebChipManager）

服务器管理模块为管理员提供了一个管理 WebChipServer™ 的可视化用户界面。该模块并不对外提供任何服务，它只包含一个 Bundle 属性编辑器，并将该编辑器资源在 mBServer 的 Web 服务器上注册。该编辑器作为一个 jar 文件，被命名为 pguieditor.jar,该文件又被打包到 Bundle 文件 WebChipManager.jar 中，这个属性编辑器作为一个管理工具被集成到 Prosyst mBServer 4.1 的管理工具 Administrator 的框架中。每当打开 Administrator 时，pguieditor.jar 被下载到本地机器开始执行，该模块提供了用户管理，设备管理，权限管理以及用户连接实时监视等功能。

用户管理: 提供新建用户，修改用户信息，删除用户，临时禁止用户/重新使能用户等功能。

设备管理: 提供手工增加设备，发现设备，定义设备别名，配置设备通信参数等功能。

权限管理: 利用权限管理功能，可以很方便的为用户分配设备的访问权限，规定用户可以访问某些设备，以及禁止用户访问某些设备。

连接监视：利用该功能，管理员可以直观的监视每一个用户到 WebChipServer™ 的连接和断开情况。

本文提供了 WebChipServer™ 的概要介绍，有关使用和开发的详细信息，请参阅相应的文档。

参考文档:

《MCUap™ 协议》

《WebChipServer™ 开发者指南》

《WebChipServer™ 管理器用户指南》

《WebChipServer™ 用户安装指南》

第三部分
WebChipServer™ 管理器用户指南

3.1 WebChipManager 简介

WebChip Manager 是 WebChipServer™ 系统的管理工具，它被设计用来完成各种管理功能，这些功能可划分为四个模块，这四个模块分别是:

- 用户管理模块
- 设备管理模块
- 权限管理模块
- 连接监视模块

每个模块位于一个单独的 Tab 页面中，如图 12 所示。

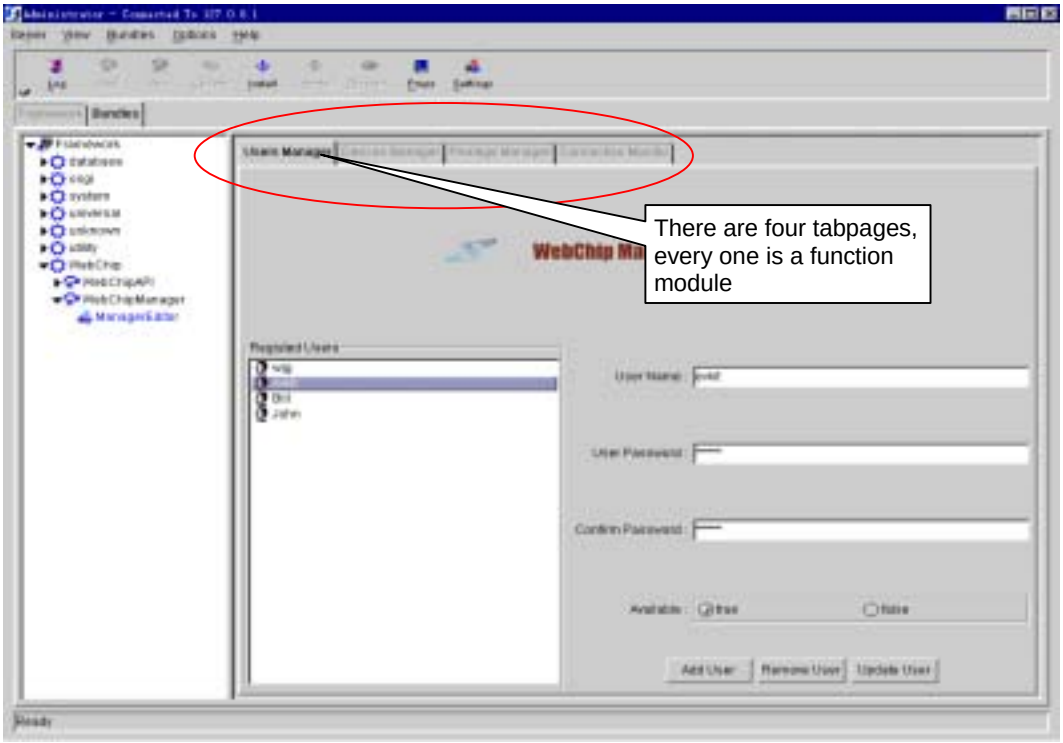

图 12 WebChipManager 的四个功能模块

WebChip Manager 实际上是一个 Bundle 属性编辑器，该编辑器被打包在 pguieditor.jar 文件中，而该文件又被打包到 WebChipManager.jar 文件中。因此在使用 WebChip Manager 之前，必须安装并启动 WebChipManager.jar Bundle。同时由于 WebChipManager 在运行过程中要用到

WebChipAPI 的服务，所以在启动 WebChipManager.jar 之前，一定要先安装和启动 WebChipAPI.jar Bundle。这个管理工具被集成到 Prosyst 的可视化管理工具 administrator 中，当管理员打开 administrator 时，pguieditor.jar 文件从运行 mBServer 4.1 的服务器下载到运行 administrator 的客户端中执行。

在内部，WebChip Manage 的功能主要是维护 PointBase 数据库 webchip 中的三个表，这三个表分别是：用户表（AllUsers），设备注册表（DevicesRegister），用户权限表（UserPrivilege）。其中用户表中存放所有合法用户的信息，包括用户姓名，口令，目前是否有效；设备注册表中存放有目前连接到服务器的设备信息，包括设备的标识符，设备的名称，访问路径，设备的口令以及设备当前的连接计数等信息；用户权限表中存放有用户对设备的访问权限信息，包括用户名，设备标识符，设备名称以及目前使用该用户名来连接到设备的连接计数。如果用户有权访问一个设备，在该表中就有一条相应的记录，当为用户赋予对某设备的访问权限时，增加一条相应的记录；而取消用户对某个设备的访问权限时，则相应的记录被清除。

在外部，WebChip Manager 采用可视化的操作界面，通过简单的点击操作完成相应的管理任务。在 Prosyst 的 Administraor 工具主界面中，点击 Bundle 标签页左面导航树中的 WebChip / WebChipManager / ManagerEditor 节点，就可打开 WebChipManager 的用户界面，如图 13 所示。

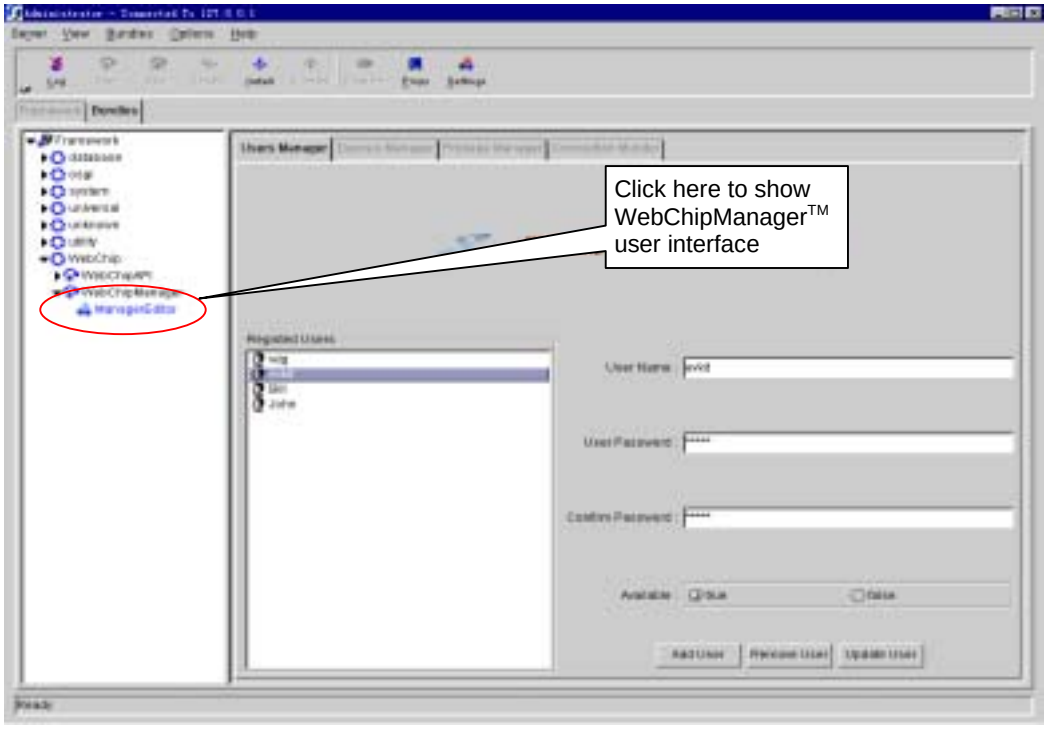

图 13 WebChipManager 的用户界面

3.2 功能模块介绍

3.2.1 用户管理模块

用户管理模块的主要功能是管理可以使用 WebChipServer™ 服务的用户，该模块可以完成新增用户，删除用户，修改用户信息，临时关闭用户（Disable，比如用户欠费情况下，临时停止为该用户提供服务），重新开通用户（Enable,比如用户交清欠费后，重新开通）等功能，该模块主要通过维护一个用户表（AllUsers）来实现其功能，当客户端连接设备时，连接设备函数首先要检索该表，察看用户输入的用户名，口令是否有效，以及用户当前是否是有效的（Enabled），只有确认该用户的合法性和有效性以后，才能继续下面的其它工作，否则，连接方法返回 “NoUser” 的出错信息。

3.2.2 设备管理模块

设备管理模块的主要功能是管理连接到 WebChipServer™ 服务器的智能设备，使用该模块可以完成注册设备，注销设备，配置设备的通信参数等操作。该模块主要通过维护设备注册表（DevicesRegister）来实现其功能，devicesregister 表中存放了有关设备的各种信息，如设备标识符，设备名称，设备口令等。如前所述，当客户端连接到设备时，连接设备函数首先要检索 AllUsers 表，察看用户输入的用户名，口令是否正确，以及用户当前是否是开通的。若检索成功，将进入设备验证阶段，在这一阶段连接方法要查询 DevicesRegister 表，看设备是否存在，若设备不存在， 将返回出错信息 “NoDevice”，只有设备存在，才能继续下一步的工作。

3.2.3 权限管理模块

权限管理模块的主要功能是管理用户是否可以访问某个设备，使用该模块可以完成为用户设定访问设备的权限，取消用户访问设备的权限等操作。在底层，这些操作的结果是对权限表（UserPrivilege）进行相应的维护。每当为用户新增一个可访问的设备时，权限表中新增加一条相应的记录；而收回用户对设备的访问权限时，相应的记录就被删除。UserPrivilege 表有两个用途：其一用来在客户端连接到设备时进行权限验证，看用户是否拥有访问某设备的权限。如前所述，当客户端连接设备时，连接设备函数首先要检索用户表，察看用户输入的用户名，口令是否有效以及用户当前是否是开通的，然后继续验证设备是否存在，这时连接方法将查询设备注册表，看设备是否存在。若设备存在，则要继续检查权限表，看用户是否有权访问要连接的设备，如果有相应的权限，则开始实际的设备连接过程，否则，返回 "NoPrivilege" 出错信息 ； 权限表的第二个用途是为连接监视模块提供用户连接计数，该计数用来向管理员提供用户连接设备的有关信息。

3.2.4 连接监视模块

连接监视模块的主要功能是使管理员能够实时监视用户连接设备的情况，该模块的界面由三部分组成：左边列表框实时监视正在连接或断开的用户以及连接和断开时间，右边的两个表格则分别列出了一个设备总的连接计数以及每个设备按用户统计出的连接计数（参见图 19 连接监视模块的界面）。

该模块的工作过程大致是：每当用户成功连接到设备时，设备连接函数都会将设备注册表的设备连接计数以及用户权限表的用户连接计数累加 1，然后触发"UserConnectEvent"事件，连接监视模块监听到该事件后，更新程序界面。

同样，每当用户断开设备的连接时，设备断开函数将设备注册表的连接计数以及用户权限表的用户连接计数递减 1，然后触发 "UserDisConnectEvent" 事件，连接监视模块监听到该事件后，更新程序界面。

3.3 使用说明

3.3.1 用户管理模块的使用

用户管理模块的界面如图 14 所示。

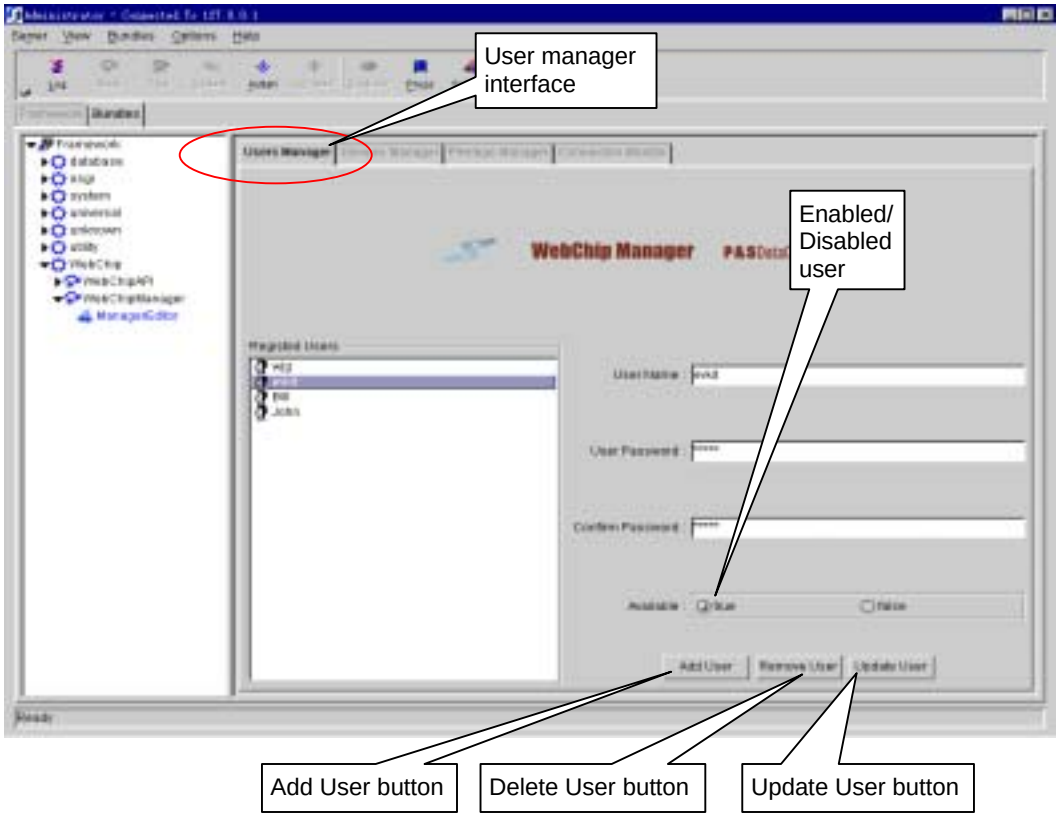

图 14 用户管理模块的界面

用户管理位于第一个 Tab 页面中，在这个界面中，管理员可以完成增加用户，删除用户，修改用户，禁止用户/允许用户的操作。

1. 增加用户

（1）点击 Add 按钮, 弹出增加用户对话框，如图 15 所示;

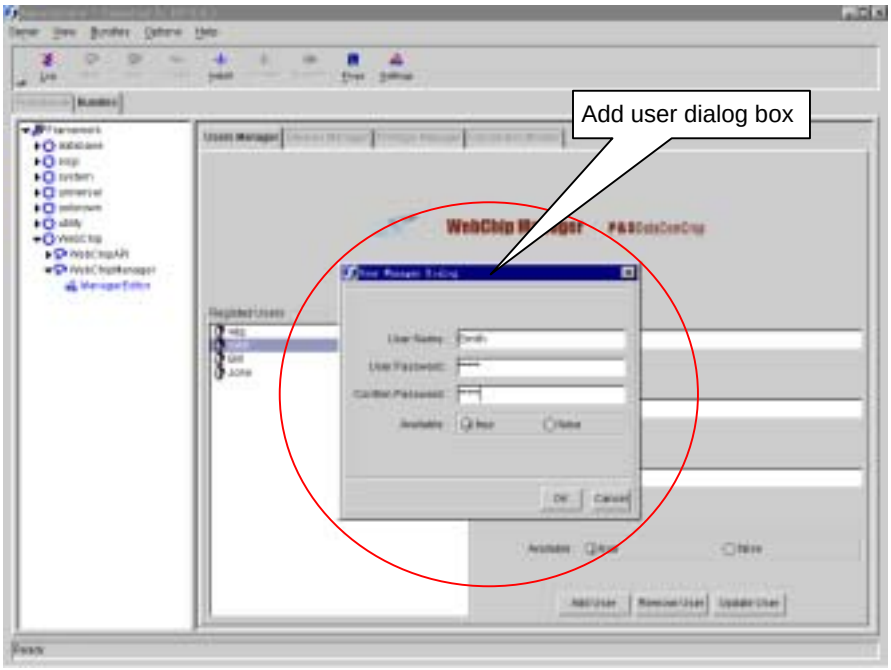

图 15 增加用户对话框

（2）输入用户名，口令，选取是否允许/禁止用户;

（3）点击 OK 按钮。

在新建用户时，要注意以下几个问题:

- 用户名和口令由字母和数字组成，其它字符不被接受
- 用户名和口令不能为空
- 两次键入的口令必须一致
- 新用户不能与已有用户同名

2. 删除用户

（1）在左边的用户列表框中，选中欲删除的用户;

（2）点击 Remove User 按钮, 弹出确认删除对话框;

（3）点击 Yes 按钮确认。

说明: 当删除一个用户时，该用户的所有设备权限都被清除。

3. 修改用户

（1）在左边的用户列表框中，选中欲修改的用户;
（2）修改用户的名称、口令，以及允许/禁止状态;
（3）点击 Update User 按钮。

说明: 修改后的新用户名称不能与已有用户同名。

4. 禁止/允许用户
（1）在左边的用户列表框中，选中欲修改的用户
（2）点击 单选钮 ○ true 或 ○ false
（3）点击 Update User 按钮。

说明: 如果用户的 Available 状态为 false，用户将不能连接到任何设备。

3.3.2 设备管理模块的使用

设备管理界面位于第二个 Tab 页面，如图 16 所示。

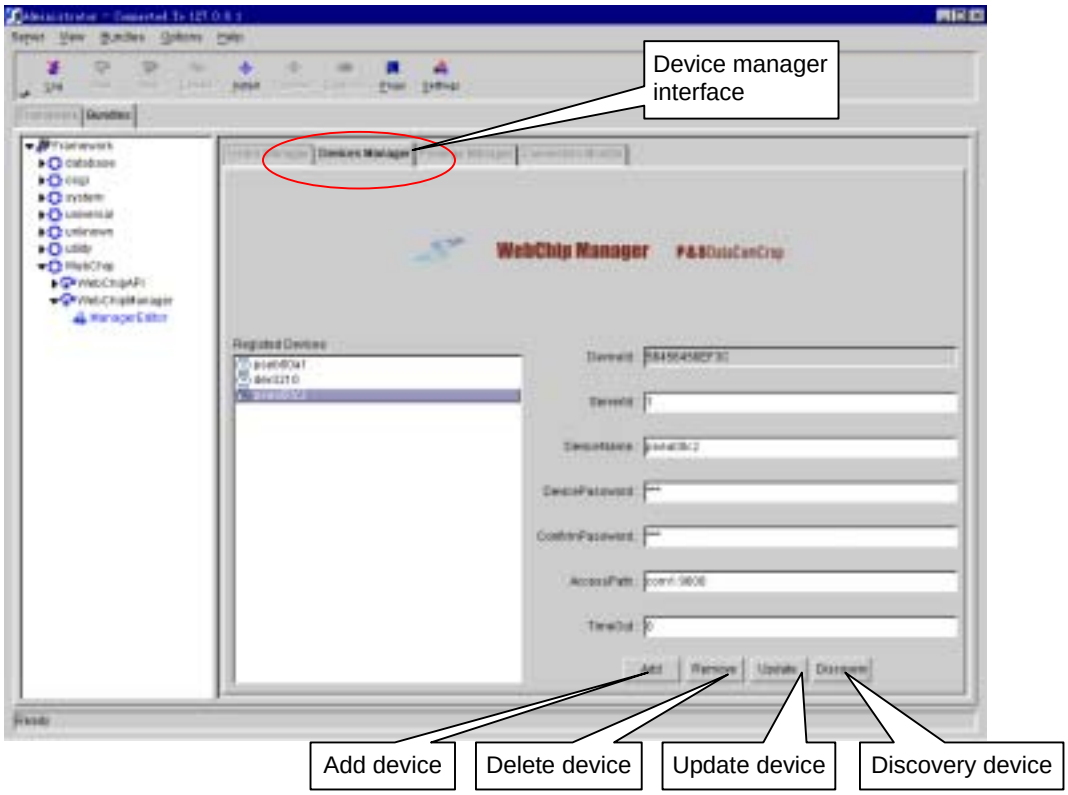

图 16 设备管理界面

1. 增加设备

（1）点击 Add 按钮，弹出增加设备对话框，如图 17 所示;

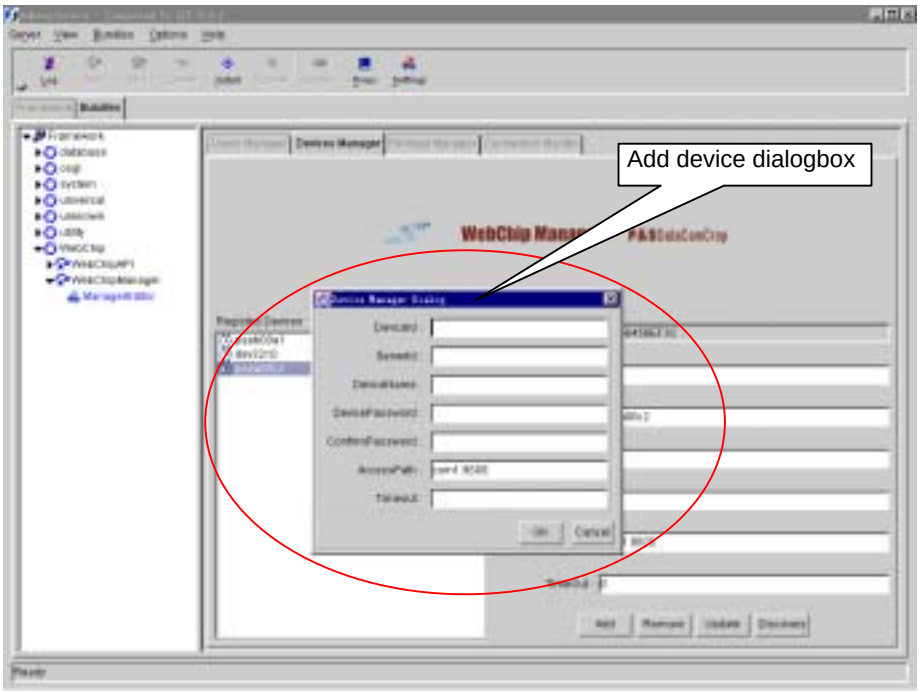

图 17 增加设备对话框

（2）键入设备标识符，服务器标识符，设备名称，设备口令，连接参数，超时常数;

注意:
设备连接参数（AccessPath）的格式为:
端口号: 波特率: 电话号码
如果不通过 Modem 连接设备，那么电话号码必须省略，此时连接参数的格式为:
端口号: 波特率:
最后的冒号不能省略

（3）点击 OK 按钮。

说明: 当注册新设备时要注意以下几个问题:

- 设备标识符和服务器标识符只接受数字
- 定时参数（TimeOut）只接受数字
- 不允许新的设备标示符和名称与已有设备的标识符和名称相同

2. 删除设备

（1）在左边的设备列表框中，选中欲删除的设备;

（2）点击 Remove ，弹出确认删除对话框;

（3）点击 Yes 按钮确认。

说明: 当删除一个设备时，该设备也从用户的权限列表中删除。

3. 修改设备

（1）在左边的设备列表框中，选中欲修改的设备;

（2）在右边的各个输入域中键入新值;

（3）点击 Update 。

注意:

- 如果修改连接参数，请依照上述格式键入
- 设备标识符禁止修改
- 修改后的设备名称不允许与已经存在的其他设备名称相同

4. 发现设备

当有新设备连接到服务器时，需要进行 ”发现” 操作，读出设备的标识符，口令，通信参数等设备信息，在设备管理界面中，提供了发现设备的功能，具体操作如下:

（1） 点击按钮 Discovery ,弹出发现设备对话框，如图 18 所示:

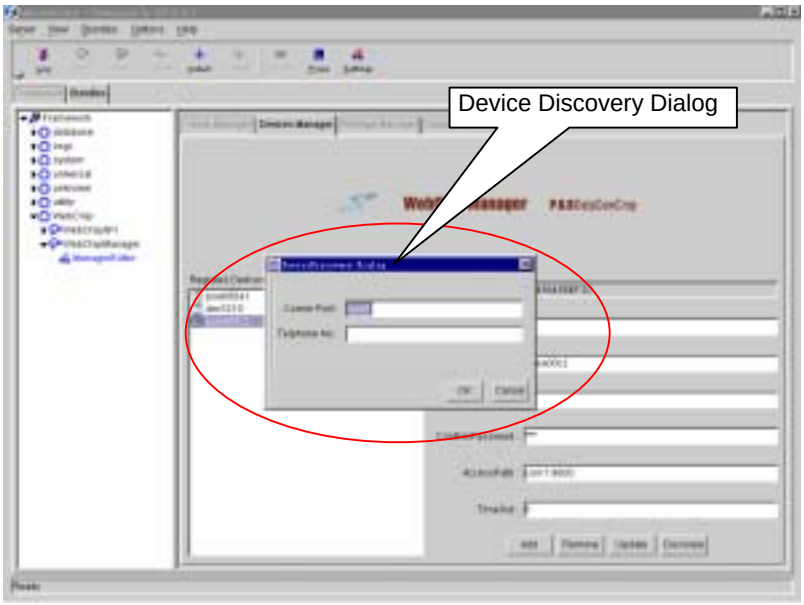

图 18 发现设备对话框

（2）在对话框中填写设备连接的端口号，如果设备是通过电话线远程接入服务器，则要填写设备的电话号码，否则，保留电话号码域为空。

（3）点击 OK 按钮，如果发现成功，设备将被加入到设备列表和用户权限列表的未授权设备列表中（参见 3.3.3）. 否则将弹出信息框说明发现失败，用户可重新填写端口号和电话号码重试。或者按 Cance 退出发现操作。

（4）发现设备成功后，应该修改设备的名称。

3.3.3 权限管理模块的使用

权限管理模块的界面如图 18 所示。

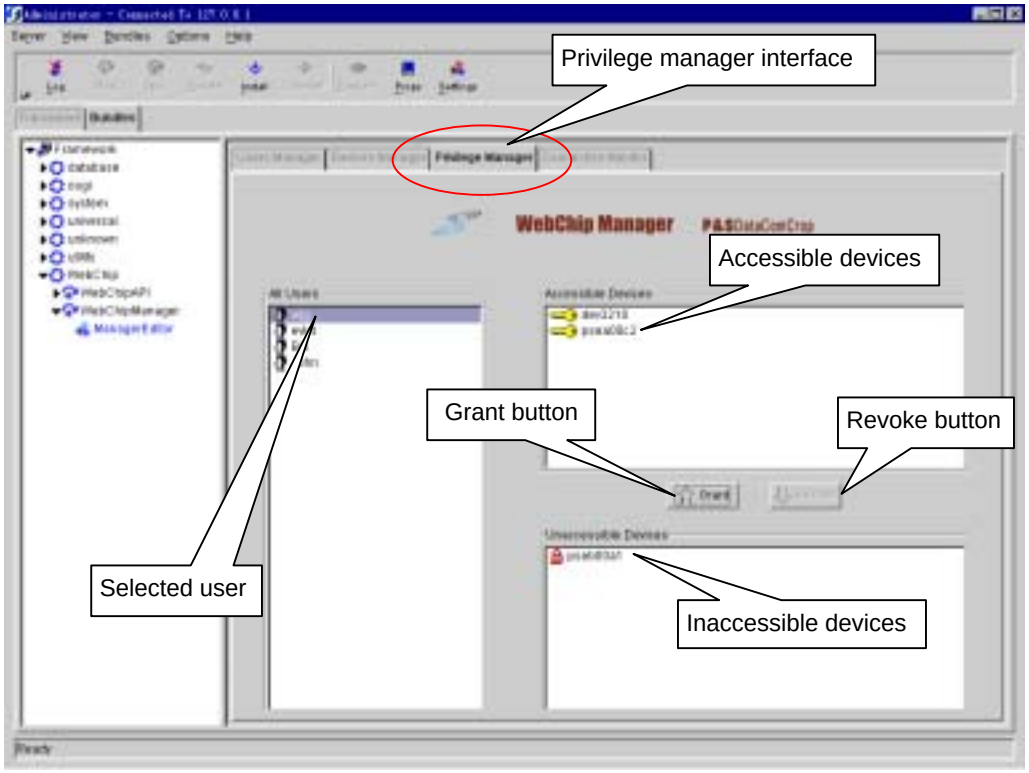

图 19 权限管理界面

1. 授予用户访问设备的权限

（1）在左边的 All Users 列表中选中欲授权的用户;

（2）在右边的下部 Unaccessible Devices 列表框中选取打算授权该用户访问的设备（可以一次选中多个）;

（3）点击 Grant 按钮。

2. 取消用户访问设备的权限

（1）在左边的 All Users 列表中选中欲取消授权的用户;

（2）在右边的 Accessible Devices 设备列表框中选取打算禁止该用户访问的设备（可以一次选中多个）;

（3）点击 Revoke 按钮。

3.3.4 连接监视功能的使用

连接监视模块的界面如图 19 所示。

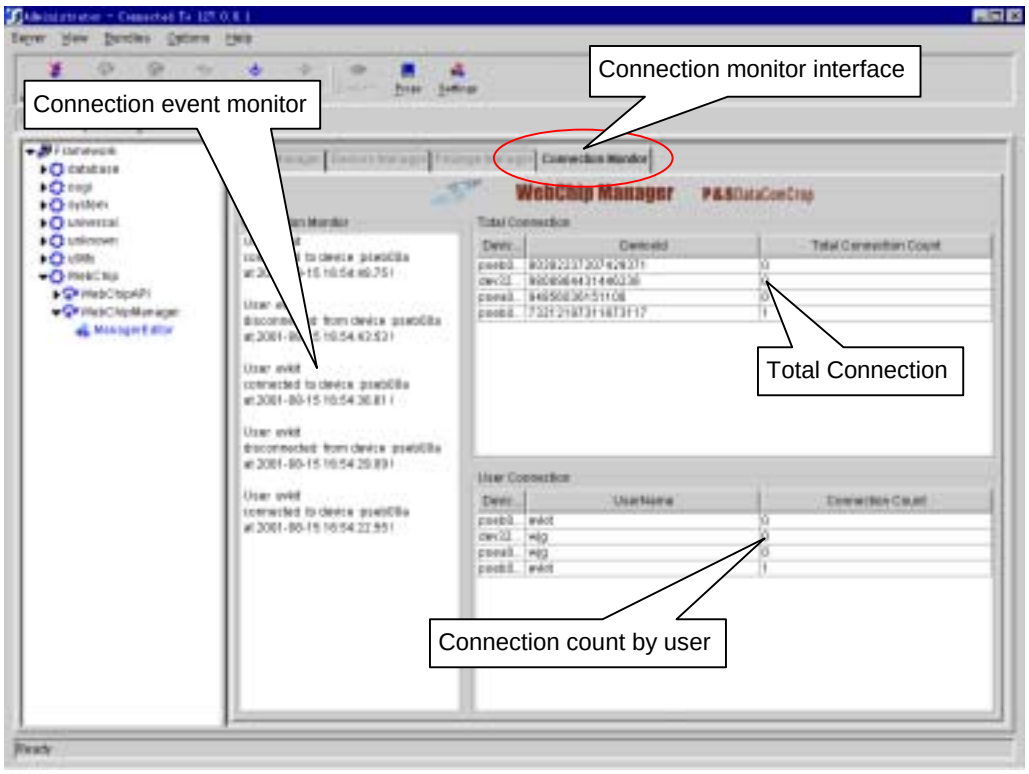

图 20 连接监视模块的界面

连接监视界面用来实时监视用户的连接情况，左边的列表框显示最近 100 次用户的连接和断开的历史记录；右边的两个表，上面是统计连接到每个设备的总的连接数，下面是统计用同一个用户名连接到一个设备的连接数。该页面不需要用户操作。