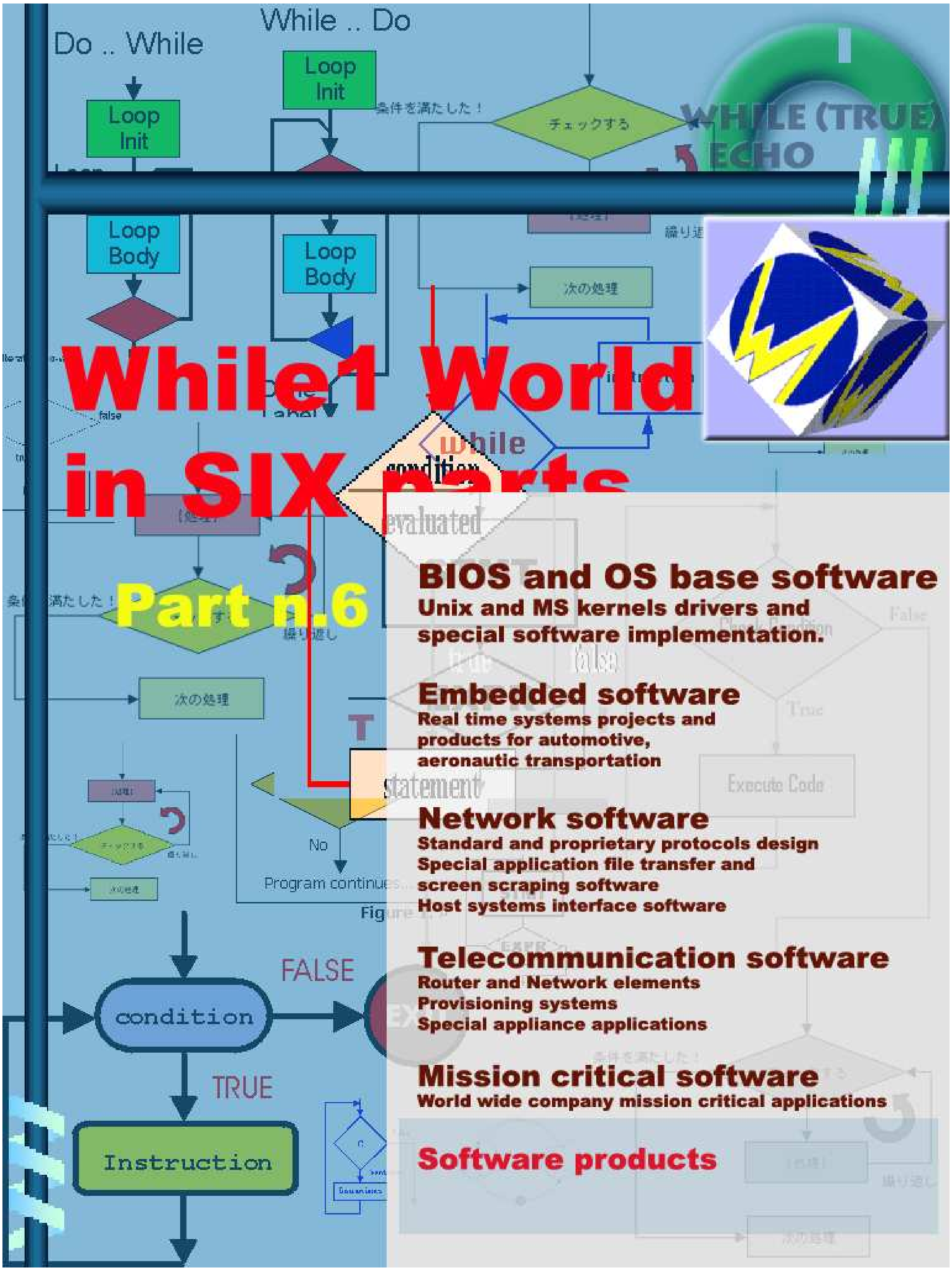


While1 World in SIX parts

Part n.6

BIOS and OS base software
Unix and MS kernels drivers and special software implementation.

Embedded software
Real time systems projects and products for automotive, aeronautic transportation

Network software
Standard and proprietary protocols design
Special application file transfer and screen scraping software
Host systems interface software

Telecommunication software
Router and Network elements
Provisioning systems
Special appliance applications

Mission critical software
World wide company mission critical applications

Software products

Some of the proprietary products

While1

(www.while1.com)

FELIX - Fast Emulator & Legacy Interface eXtender

***FELIX* provides host publishing for Legacy applications.**

Product Highlights

FELIX is a product that provides web access to Legacy systems.

It allows both raw web access to legacy system and the possibility to build enterprise applications for legacy services integration.

This means that there is the possibility to map host screenshots one to one or automatically browse between host applications, in an easy way.

Choose your best host access way for your own business.

- Choosing to map the screenshots as they were developed, you can access via internet your Legacy systems without modifying host applications but improving the security policy. Moreover you will be able to work with your internet client without installing any components, just through your internet browser.
- Using the Software Development Kit (SDK) you will be able to develop powerful applications that perform an interactive work with Legacy environment.
- For example you can write an application that receives as input parameters the data through a web page ,dealing them on more screenshots.
- With this method you will improve the performance of your application because the interaction between user and host applications is based on a new service built collecting and using old host transactions.

Other features of *FELIX service.*

This product manages, using a database, all user access to the host applications.

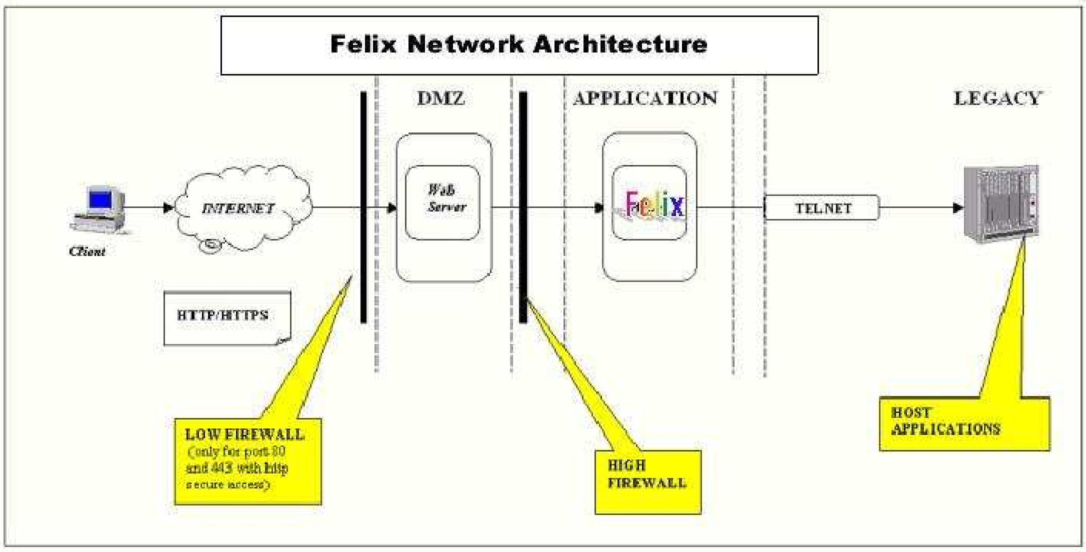
This means that it can manage the number of user accesses choosing

user and password, in order to allow a connection in a transparent mode for each user .

FELIX manages the access to host applications through following protocols:

- 3270
- 3270E
- 5250
- 5250E

FELIX is based on distributed services on the network, in order to manage load balancing to achieve scalability and reliability.



FELIX System hosting

FELIX architecture as it looks like in the picture above.

A thin client can connect through its browser via internet and can place requests for any operation to FELIX.

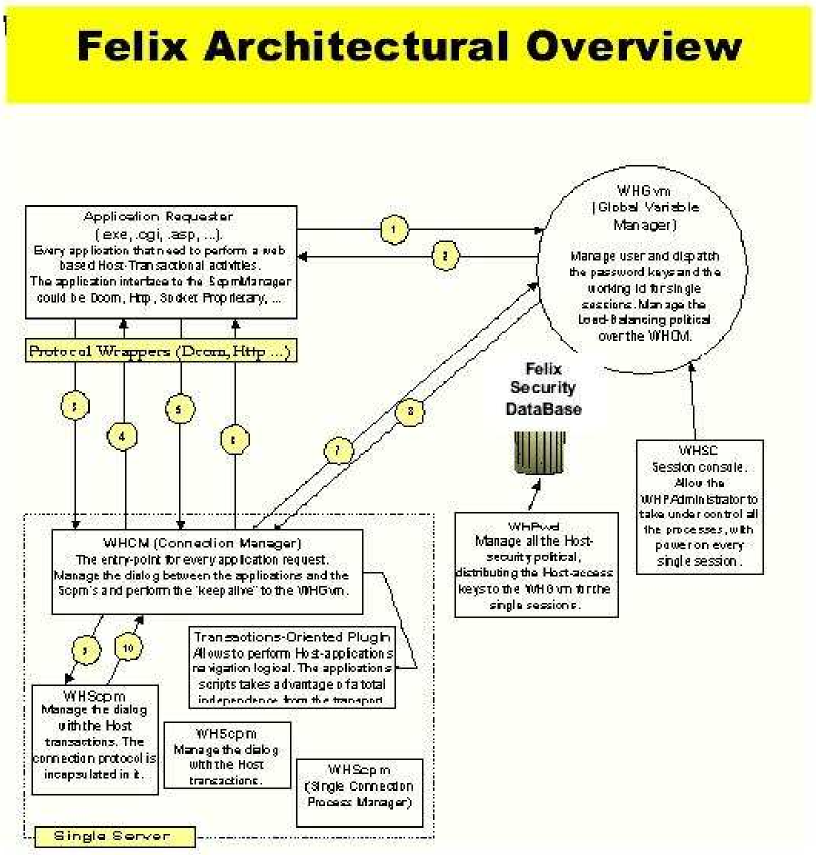
Then the requests pass to Low Firewall (access only for port 80 and 443 with secure access https) by means of protocol http/https, in order to enter in the Demilitarized zone (DMZ).

Here sits the Web Server that can be represented by "IIS ", "APACHE" , "WEBSPHERE" on WINDOWS2000 Server OS or UNIX.

Moreover all the client's requests are collected and are forwarded to the *FELIX* service residing in the Secure Application zone.

Before getting into Application zone there's the second access to firewall (in this case High Firewall) .

Here *FELIX* interacts with host applications (by means of protocol Telnet) on Legacy System .


Felix Architectural Overview
Protocol Wrappers (Dcom, Http ...)
1
2
3
4
5
6
7
8
9
10
Felix Security DataBase
Single Server

Application Requester
(exe, .cgi, .asp, ...).
Every application that need to perform a web based Host-Transactional activities.
The application interface to the ScpmManager could be Dcom, Http, Socket Proprietary, ...

WHGvm
(Global Variable Manager)
Manage user and dispatch the password keys and the working id for single sessions. Manage the Load-Balancing political over the WHCM.

WHCM (Connection Manager)
The entry-point for every application request.
Manage the dialog between the applications and the Scpm's and perform the 'keep alive' to the WHGvm.

Transactions-Oriented PlugIn
Allows to perform Host-applications navigation logical. The applications scripts takes advantage of a total independence from the transport

WHScpm
Manage the dialog with the Host transactions. The connection protocol is incapsulated in it.

WHScpm
Manage the dialog with the Host transactions.

WHScpm
(Single Connection Process Manager)

WHPwd
Manage all the Host-security political, distributing the Host-access keys to the WHGvm for the single sessions.

WHSC
Session console.
Allow the WHPAdministrator to take under control all the processes, with power on every single session.

Reference of Felix use:

- **841 Project.**

We have developed an application that uses some of *Felix* capabilities.

Our goal was to obtain something able to move all old host screenshots into new graphic interface as a common web site.

Moreover we wanted improve the performance application because we were using a very slow net on the client side.

In order to obtain our goal we realized a Felix plug-in able to use a script that automatically browse through host interface. In this way the user performs all inputs in a friendly web page that posts the data to the web server. The web server through the Felix plug-in interacts with host performing all needed transactions in different screens, returning the final results to the user.

- **Friend Web**

We developed an application that uses *Felix* to access to AS/400.
The applications on the AS/400 were developed using the screen as a buffer for read and write data.
The plug-in of this application allows to the Web to work with this methodology.
In both projects security and single sign-on fundaments were performed.
The security was assured by the use of HTTPS from internet client.
The "single sign-on" was realized using *Felix* user access feature configuring links of WEB
and Host Login Information.

- **CNH European Dealer Connection**

This is the simplest example of *Felix - Based* development.
In fact we created a **dynamic** web page that forwards user-inputs to the host (via *Felix*) and shows on the browser its response-screen.
The user has only to log into the web, and he gets into the Host!
So that it's possible web surfing among all the Focus-Multi-Market transactions. Therefore no translations are required.
7500 dealers will be connected via Felix to CNH AS400 environments.

WMTK – Real Time Kernel

Our experience in developing basic software and particularly UNIX kernels resulted in a product that provided us a **REAL-TIME** and **TIME-SHARING MULTI-TASKING** support on non standard hardware systems, which do not have and cannot have a standard operating system. **WMTK** is then used to work out all issues occurring in the so-called "embedded" systems, where there must be a kernel that hosts multiple TASKS at the same time and grants real-time operativeness. **WMTK** purpose is to be a cost-effective solution to **REAL-TIME** issue. Other well- known products of this kind undoubtedly use integrated advanced systems for development and debugging, but they are actually very expensive and need Royalties for each installation. **WMTK** has the following key features:

- It is extremely **easy to use**.
- Minimum **disk space** required: **10 Kb**.
- **Modular kernel**, which are open to any extension.
- **DDK** interface for drivers development.
- Written in **"C"** (easily portable on any hardware platform).
- Version for **Intel 80x86** and **Motorola 68xxx** environments is already available.
- **Sources** available
- **Very Low price**
- It can be integrated with **INTDEB** and **MOTDEB**

Along with a developing phase still in progress, we are also adding in **WMTK LAN** support and particularly **TCP/IP** protocol, with its **MAC** drivers for the most common adapters in **PCI** (3com, Zynix, etc.) architecture.

Applications running on **WMTK** environment will be able to use traditional **LAN** interfaces, such as **SOCKETS** or **TLI.**

TSI – Transport Session Interface

WHILE1 designing component, named **TSI** (transport session interface), can establish a connection-oriented channel for communications between two or more entities, thus releasing applications from the physical communication mode.

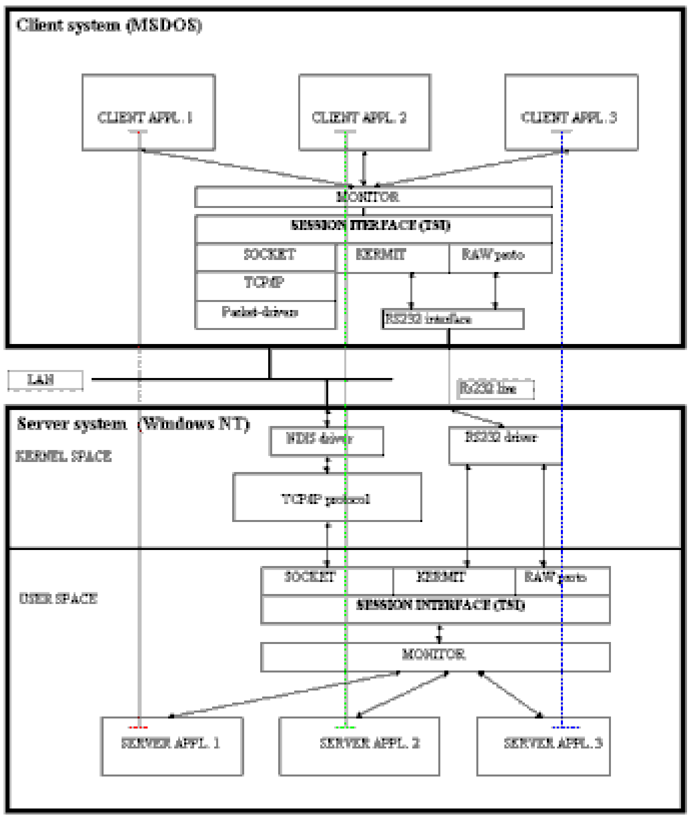
TSI implies a series of **communication devices**, such as the following:

- **Socket** on TCP/IP
- **X25** (IBM boards, EICON boards)
- **Windows Pipes** (only Microsoft WIN32)
- **Windows Mailbox** (only Microsoft WIN32)
- **TLI** (only Unix)
- **Netbios** (only Microsoft DOS and Windows 3.1x)
- **KERMIT serial protocol**
- **WISP** (While1 Internal Serial Protocol)
- **Raw mode** (no protocol)
- **WANAPPC** on Microsoft SNA server
- **HLLAPI** on any systems working in 3270/5250 emulation

With all listed protocols on ever communication channels are available next features:

- **Compression**
- **Encription**
- **7 bit Coded**

TSI use example



TSI is designed to export to applications a transport interface, which is completely independent from the communication system used. You will then be able to write applications regardless of the communication interface system you want to use at present or in future. In addition, the same application can be hosted by systems which use different communication methods, with no changes.

You might also use **TSI** if you need **resilient** links, in case you expect to connect clients with servers using more than one connection. You will then start concurrent connections, physically different, always using the same interface.

TSI interface semantics is rather like the **Socket (BSI)** interface, so it is easy to use and understand.

TSI is available on Microsoft systems: **DOS, Windows 3.11, Windows 95/98** and **Windows NT/2000**, and on **UNIX** standard systems as well, such as: **SCO, UnixWare**, **Solaris, HP-UX, Linux** etc.

In particular situations you can still execute porting even on systems which are different from the above mentioned

PLDEB – Remote PipeLine Debugger

RPLDEB is a symbolic debugging tool for the software developer. It provides an efficient, easy-to-learn environment for the otherwise complex process of assembly or machine language problem diagnosis on the systems 8086 (real mode based).

If you use an assembler or any higher level, compiled language and need to observe your program's operation to diagnose its problems, this is the tool for you.

RPLPDEB is the remote version of original PLDEB tool, and is composed by two part. The first part, named CORE, is loaded on the target system and the second part, named USER, is loaded in the PC debugger machine.

The RPLDEB use a serial RS232 line to link CORE and USER part, and using this architecture is possible to develop inside target system a little piece of code that execute a CORE debugger functions (STOP, GO, R/W-MEMORY, SINGLE_STEP, ...).

User interface part is execute on the PC machine and is separated from the CORE.

Its main features are:

• **COMMAND ARE EASY**. There are no commands. Simply fill in the blanks. The base case function keys are the only "commands" and 7 of these have consistent meaning throughout RPLDEB. No complicated key sequences. Not much to learn here...

• **SYMBOLIC ADDRESSING**. All addresses may be specified symbolicly using your own names. Addresses may be specified as any combination of symbols, register and offsets.

• **STANDARD KEYBOARD FUNCTIONS**. Full screen format with full use of standard cursor, scroll, and character control keys.

• **SCREEN COEXISTENCE/GRAPHICS**. A debugger does nor have to interfere with your program's use of the screen. The test program's screen contents and mode are constantly preserved while RPLDEB is in control and restored for the single step or breakpoint.

• **TEXT FILE BROWSING**. Any standard text file may be viewed without leaving or interfering with the debug process. Want to consult the listing?

• **SIXTEEN BREAKPOINTS**. Each breakpoint can be set for a variable number of occurrences. The program is not interrupted until that number is reached. A total for each breakpoint is displayed. How many times did it go through there? or, Stop that loop on the 999th entry?

• **LOOP BREAKER** You say your program is looping and not hitting that breakpoint? With other debuggers, program loops, even those enabled for interrupts can force you to power the machine off to regain control. With RPLDEB, program loops which have not destroyed memory or disabled interrupts can be interrupted. This can be a life saver.

• **VERSATILE SINGLE STEPPING**. From one to over 64,000 single steps can be specified at each 'go' (yes, RPLDEB is fast enough to make that number practical), and a total is displayed. Want to count instructions? Or "see" when the data goes bad?

• **CONSTANT SYMBOLIC LOCATION DISPLAY**. The symbolic instruction pointer is constantly updated and displayed while single stepping.

• **CONSTANT DISPLAY MEMORY**. 9 lines of 16 bytes each, each showing its own user-specified memory area in HEX and ASCII, are constantly displayed while single stepping. Watch the data change.

• **FAST INTERRUPT ROUTINES**. Interrupt routines, unless otherwise specified, are treated as a single instruction in single step mode. Makes single stepping a lot safer.

• **FAST "LONG" INSTRUCTIONS**. CALL or REP prefix instruction may be treated as a single instruction by using the "SKIP" function key in single step.

• **SINGLE STEP HISTORY**. The history of the last 6 single stepped instructions and contents of key maching registers and flags are displayed.

SCUBE – SNMP Super Poller

SCUBE **(SNMP Shot Sentinel => S3 => Scube)** product is an application able to collect information coming from systems and devices that exports them via SNMP protocol.

There are more reasons that push us to carry out this object, so target and architecture definition were defined after a deep study on the most common pollers.

The considerations arose after this study (positive and negative) on the other products, led to the SCUBE project definition.

High Performances

SCUBE has been designed to reach high performances. Code has been written trying to optimize every function and system resources use has been projected taking their efficiency in consideration according to the operating systems on which SCUBE is used.

System calls and library-functions set used has been chosen emphasizing performances aspect.

Measurements performed on some HP systems used as reference, allow us to assert that SCUBE can receive data of 40.000 objects per minute.

Scalability

SCUBE has been designed to be used on a wide range of systems, therefore it can operate both on small systems and big Main-Frames.

Its structure adapts easily on available resources. It requires only what the system can offer in terms of memory/processes/communication etc.

It is obvious that the more powerful host system is the higher performances you will get.

Distributed execution

To guarantee the maximum scalability to SCUBE, we adopted an Inter Process Communication architecture among the processes. In this way, some poller's modules can be run on remote system as well.

What you can get is a real distributed system, able to satisfy any load of jobs.

To avoid the excessive use of IPC, we configure "remotized" mode only if strictly necessary; in the other situations, usually we work locally using the most efficient system.

Load balancing among the processes composing operating scenario is always dynamically computed and modified according run-time progress.

Portability

SCUBE code has been completely developed in system-independent mode, so it is easily portable on every operating system.

The pieces of code strictly related to the operating systems are inserted in libraries that provide common interfaces to poller application layer.

To port the poller on a new operating system, we only have to write a new interface library to convert poller application needs in OS specific system calls.

Small dimension

"Small is beautiful"... it's true for SCUBE. The poller is composed by few parts; so for installation and run-time you have to control few components (processes, system resources, configuration files).

The target "few elements" does not reduce project complexity; indeed SCUBE core contains more than 22,000 code lines and SNMP library derived from UCD version has 28,000 circa.

Reduced use of system resources

To contain strong loads, we tried to design an architecture that does not require too many operations to handle processes and memory resources (semaphores, shared memories, message queues).

Following this dircetive, we tried to define a number of processes (threads) correctly dimensioned on the basis of polling run-time needs and not on fixes rules defined once by an administrator.

Therefore, there is an auto-tuning performed by the poller itself depending on the amount of operations to perform.

Compatibility

SCUBE is compatible with HP OpenView; it can substitute HP product without particular integration effort.

Both input and output files have same formats.

SCUBE is ready to communicate with SNMP Agents operating with SNMP protocol version 1,2 and 3.

Setting and total tuning

Poller is fully settable; that is all the levels inside it can be dimensioned and set depending on the installation to be performed.

Setting is extended from system resources use (socket, files, etc.) to SNMP protocol tuning.

WHILE 1 S.r.l.

The measure of quality

www.while1.com

www.biospc.com	www.unix-drivers.com
www.ms-drivers.com	www.scsi-drivers.com

info@while1.com

Italy Headquartier : Corso Turati, 70 - 10134 Torino

Italy office : Environment Park Via Livorno, 60 - 10144 Torino Tel./Fax. +39 (011) 2257721

Italy office : ICO Centrale, Via Jervis, 9 - 10015 Ivrea (To) Tel./Fax +39 (0125) 641607

USA office: 405 El Camino Real #219 - Menlo Park CA 94025 Tel. +1 (650)317.19.74