

***Platform Update Facility***  
**Product Brief**  
Version 1





Introduction .....	4
Features .....	5
Single 24KB Firmware Application Executable, Compresses to 12KB in ROM.....	5
OEM-Configurable in System Registry.....	5
Operates Without OS Support .....	5
Supports Manual Replacement of Upgradeable System Components.....	5
Automatic Upgrades Triggered by ODM/OEM-defined Events .....	6
Applications.....	6
Manufacturing Software Load.....	6
Automatic Device Recovery.....	7
Remote Service Calls .....	7
Sample Configuration.....	7
System Registry Section.....	7
Object Directory File.....	8
Default Security Authority User Authorization File.....	9

## Introduction

Platform Update Facility is a firmware application that allows manual replacement of upgradeable system components at any time during its operation. It can upgrade disk partitions, files within partitions, CMOS contents, and Flash memory.

This application can also operate automatically, checking selected upgradeable system components on each boot, and downloading and replacing them with new data as needed, based on ODM/OEM policies. This same automatic update can be triggered by conditions such as OS system death, or when failures are signaled by the diagnostics suite.

To accomplish this, Platform Update Facility uses MD5 crypto-based hashes of objects, eliminating costly boot-time delays. The definitions of updateable objects are specified by the ODM/OEM in a ROM-resident policy database. Standard Firmware® Technology TCB user-level security is provided on all objects. And, this facility is remotely manageable with any web browser, over the internet.

Common applications include loading the OS and application during the manufacturing process; updating the OS, application, and BIOS images on devices in the field; repairing software on devices in the field; and backup/restore operations of partitions, files, CMOS, and Flash memory in the target.

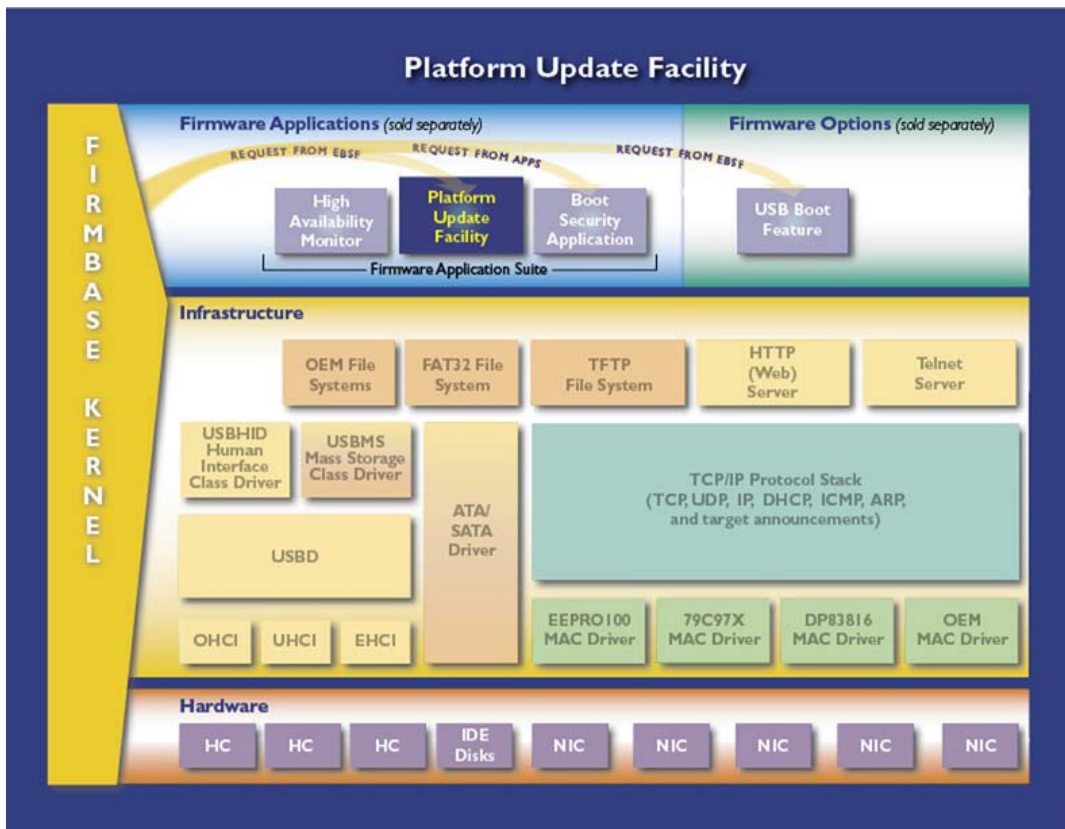


Figure 1. System Component View

## Features

### Single 24KB Firmware Application Executable, Compresses to 12KB in ROM

Typically merged as a compressed resource in the BIOS Flash ROM, this application program has a low-overhead footprint that leaves plenty of room for other system components. The application can also be loaded from FAT32 volumes residing on ATA or USB mass storage devices.

### OEM-Configurable in System Registry

Platform Update Facility is configurable in the system registry's [UPDATE] section. Working with the fundamental types of nonvolatile data in the system, the ODM/OEM can define any object as any byte range from Flash memory, the CMOS register file, and individual files on file systems; in addition, sector ranges on mass storage devices and their partitions can be used to define objects as well.

Examples include the entire BIOS ROM, video ROM extensions, the RTC range of CMOS, the standard CMOS cells for BIOS factory defaults, configuration files like CONFIG.SYS, standard Windows DLLs, or perhaps a whole FAT32 partition containing the OS binaries or application files.

This flexibility gives the OEM the ability to define proprietary objects that give a competitive edge over the competition. The simple text-based directives in the configuration files make it easy to build a library of system object definitions that add value quickly and make the system more versatile.

### Operates Without OS Support

As with all firmware applications that employ Firmware<sup>®</sup> Technology, the Platform Update Facility does not require any OS to perform its functions. It can operate in the pre-boot environment during POST, or when the OS is loading, running, crashed, or missing altogether.

As a standalone application that loads from ROM, the system is always able to be loaded in the manufacturing process, or reloaded in the field.

### Supports Manual Replacement of Upgradeable System Components

Because the Platform Update Facility registers as a service with the HTTP web server and telnet server, it can be accessed both directly on the local LAN, and remotely over the Internet through platform-independent interfaces like a web browser or telnet terminal program.

The following screen shot shows how, through the web interface, to select the Platform Update Facility as a registered service:

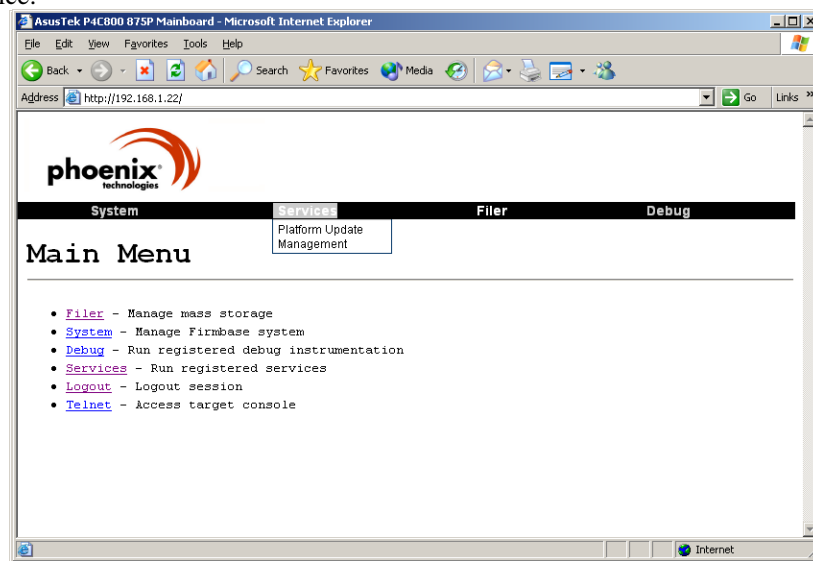


Figure 2. Selecting Platform Update as a Service

The following is a sample screen shot of the Platform Update Facility once the service has been selected:

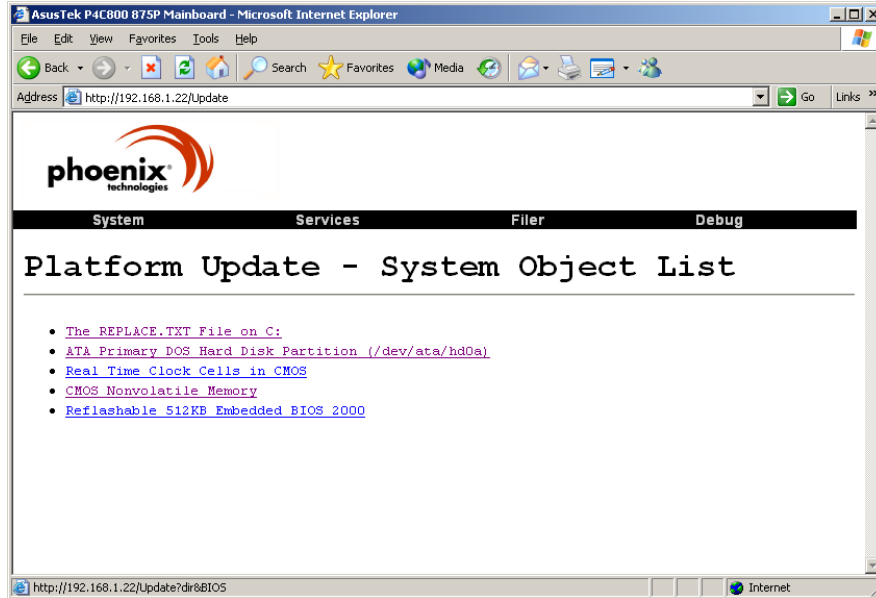


Figure 3. Platform Update Main Menu

### Automatic Upgrades Triggered by ODM/OEM-defined Events

While Platform Update Facility can be operated interactively by a local or remote user, it can also automatically be triggered by automation, in response to events such as system startup (POST), HA subsystem events like OS system death detection, diagnostic suite failure events, and boot security events.

- System Startup – Triggered by Embedded BIOS® POST
- HA Subsystem Events – Triggered by HA Monitor firmware application
- Diagnostic Suite Failure Events – Triggered by Diagnostic Suite firmware application
- Boot Security Events – Triggered by Boot Security firmware application

When Platform Update Facility is triggered, it can check objects for integrity or simply replace them, as specified by ODM/OEM-defined policies. For example, on each system startup, it can automatically scan key system files to ensure that they are the best versions that should be run. It is possible for updates to be introduced to a single system via USB mass storage for subsequent detection in this way, or on a TFTP server for large populations of devices.

As part of its configuration, each object definition contains a replacement strategy for automatic updates. This allows multiple replacement candidates to be presented on mass storage or TFTP servers for selection based on a priority code or a date/time stamp. Other objects can have no replacement strategy, making them only replaceable through interactive means.

## Applications

### Manufacturing Software Load

One of the most straightforward applications of this application is loading of additional Flash, CMOS, and hard disk partitions for newly-manufactured units on the assembly line. Because Platform Update will run when the device first boots, simply connect the device to the manufacturing network before booting it, and have a TFTP server running, containing the images of the updateable components ready for installation. Platform Update will automatically detect that the (nonexistent or not-yet-written) components need to be updated, and will perform the updates all automatically, in an unattended fashion.

Images to be placed on the TFTP server are taken from a known-working device that has passed Q/A. Interactively, the Q/A technician can make checkpoint copies of these objects and save them on mass storage, such as a USB hard disk. These can then be transferred to the manufacturing TFTP server for downloading during the manufacturing process.

### Automatic Device Recovery

Another basic application of Platform Update is an automatic, unattended, reload of corrupted components in the field, without requiring the associated down time waiting for a service call. When a device's components are deemed to be corrupt or tampered with (as determined by either Platform Update or the Boot Security application) or when the HA Monitor or Diagnostics Suite determine that the device is not functioning properly (i.e., is blue-screened), then Platform Update can automatically reload all configurable components (hard drives, Flash, CMOS, and so on) and bring the device back to a known state.

Known-good images for reload can be placed on backup hardware (extra hard disk partitions not available to the OS) or on TFTP servers on the network to which the device is connected.

### Remote Service Calls

Still another basic application of Platform Update is the interactive mode, whereby a remote technician connected to the Internet can invoke the Platform Update Facility without having to make an in-person service call. Remote service calls can eliminate costs, unnecessary down time, and customer inconvenience associated with in-person service calls.

Known-good images for reload can be placed on backup hardware (extra hard disk partitions not available to the OS) or on TFTP servers on the network to which the device is connected. These TFTP servers may also be located on the Internet if the device is connected to the Internet.

## Sample Configuration

The Platform Update Facility is configurable in several ways. The definitions of replaceable objects in the system are specified in the system registry. These definitions are given ODM/OEM-specified names (i.e., BIOS, CMOS, or DISK.) These objects contain a keyword (Directory) that point to an object directory file, which contains a list of possible replacement candidates and their MD5 hashes. Finally, the default security authority, if used, requires that authorized users have an UPDATE keyword in the authorization file.

Examples of these three files are given below.

### System Registry Section

```
[UPDATE]
Object          = BIOS
BIOS.Name       = ASUS875P-BIOS
BIOS.Comment    = "Reflashable 512KB Embedded BIOS"
BIOS.Type       = Flash          # Flash, file, CMOS, disk, etc. (default: Flash).
BIOS.Address    = 0xffff80000
BIOS.Size       = 0x00080000
BIOS.Directory  = /dev/satafat/update.txt
DISK.Hash       = /dev/satafat/cmoshash.txt # location of hash if specified.
BIOS.Strategy   = None           # None, Newest, Priority.
BIOS.Interactive = 1             # enable/disable HTTPSRV support.
BIOS.Reboot     = 1             # 1=reboot if updated, 0=don't reboot.

Object          = CMOS
CMOS.Name       = CMOS-FILE
CMOS.Comment    = "CMOS Nonvolatile Memory"
CMOS.Type       = CMOS          # Flash, file, CMOS, disk, etc. (default: Flash).
CMOS.Address    = 0x00000010    # start of CMOS nonvolatile memory.
CMOS.Size       = 0x00000070    # number of CMOS cells that support replacement.
CMOS.Directory  = /dev/rom/update.txt # directory containing updates.
CMOS.Strategy   = None           # None, Newest, Priority.
CMOS.Interactive = 1             # enable/disable HTTPSRV support.
CMOS.Reboot     = 0             # 1=reboot if updated, 0=don't reboot.
```

```

Object          = RTC
RTC.Name        = RTC
RTC.Comment     = "Real Time Clock Cells in CMOS"
RTC.Type       = CMOS           # Flash, file, CMOS, disk, etc. (default: Flash).
RTC.Address    = 0x00000000     # first addressable CMOS cell in RTC.
RTC.Size       = 0x00000010     # number of RTC cells that support replacement.
RTC.Directory  = /dev/rom/update.txt # directory containing updates.
RTC.Strategy   = None           # None, Newest, Priority.
RTC.Interactive = 1             # enable/disable HTTPSRV support.
RTC.Reboot     = 0              # 1=reboot if updated, 0=don't reboot.

Object          = DISK
DISK.Name       = ATA-DISK
DISK.Comment    = "ATA Primary DOS Hard Disk Partition (/dev/ata/hd0a)"
DISK.Type       = DISK           # Flash, file, CMOS, disk, etc. (default: Flash).
DISK.Address    = 0x00000000     # first addressable sector on disk.
DISK.Size       = 0              # sectors that support replacement (0=entire disk).
DISK.Device     = /dev/ata/hd0a # name of object to be updated.
DISK.Directory  = /dev/rom/update.txt # directory containing updates.
DISK.Hash       = /dev/satafat2/diskhash.txt # location of hash if specified.
DISK.Strategy   = None           # None, Newest, Priority.
DISK.Interactive = 1             # enable/disable HTTPSRV support.
DISK.Reboot     = 0              # 1=reboot if updated, 0=don't reboot.

Object          = FILE
FILE.Name       = CONFIG-SYS
FILE.Comment    = "The CONFIG.SYS File on C:"
FILE.Type       = FILE           # Flash, file, CMOS, disk, etc. (default: Flash).
FILE.Filename   = /dev/satafat/config.sys # name of object to be updated.
FILE.Directory  = /dev/rom/update.txt # directory containing updates.
FILE.Hash       = /dev/satafat/hash.txt # location of hash if specified.
FILE.Strategy   = None           # None, Newest, Priority.
FILE.Interactive = 1             # enable/disable HTTPSRV support.
FILE.Reboot     = 0              # 1=reboot if updated, 0=don't reboot.
FILE.UpdateOnBootFail = 1       # perform update if EBSF boot failure.
FILE.UpdateOnRecovery = 1       # perform update if system recovery required.
    
```

**Object Directory File**

Object directory files are accessed by the Platform Update Facility through standard Firmware I/O system services; therefore, they may be located in the BIOS ROM, on FAT32 file systems, on TFTP file servers, and any other file system for which a file system driver firmware application is installed in the system.

The following is a sample object directory file:

```

# This a Firmware Update directory file. It contains definitions for
# firmware available for this platform. This example was stored on
# /dev/hd1/biosdir.txt.
#
# Each line in this file represents a possible download file, and has the
# following format (white space separates parameters). Note that any
# parameter may be enclosed in double quotes, allowing embedded white space.
#
# <logical-image-name> <rw> <priority> <date> <time> <comment-string> <fb-file-name>
# [<hash-file-Name>]
#
# Note the <rw> field indicates whether the file supports reading or writing.
# You can have one or the other, or both.
#
# There may be many entries with the same logical image name, which is
# the name the automation uses to determine if an image matches any component
# of the target. When the user interacts with the Firmware Update program
# via HTML, then all options are offered that match a given logical image
# name, and the user may interactively select one. The hash filename is
# optional and if not specified, is simply not used. The hash feature provides
# a way to optimize the update so that duplicate updates are not performed.
#
# Priority codes are a simple unsigned integer. When the automation picks
# an image based on priority, higher numbers are chosen over lower ones.

asus875p-bios r 100 04/04/08 09:03:00 "Standard BIOS on /dev/hd0a/test"
"/dev/hd0a/test/asus875p.bin" "sdfo8iwfo8FHLKsfd"
asus875p-bios r 100 04/04/08 09:03:00 "Standard BIOS on /dev/hd0/test"
"/dev/hd0/test/asus875p.bin" "sdfo8iwfo8FHLKsfd"
asus875p-bios r 50 04/04/08 08:57:00 "Backup BIOS /dev/hd0a/test"
"/dev/hd0a/test/asus875p.bak" "ON/vLfYw3HG+zVcebDehlw=="
asus875p-bios r 50 04/04/08 08:57:00 "Backup BIOS /dev/hd0/test"
"/dev/hd0/test/asus875p.bak" "ON/vLfYw3HG+zVcebDehlw=="
    
```

## Platform Update Facility Product Brief Version 1

```
asus875p-bios r 25 04/04/01 09:17:23 "Basic BIOS; no FW update feature"
"/dev/hd0a/asus875p.bas" "sdSDFLIW298234Ffhysd"
asus875p-bios r 101 04/04/01 09:17:23 "Always works"
"/dev/tftp/192.168.1.242/asus875p.bin" "sdo8F23823fDFsfliqwer"
asus875p-bios r 200 04/04/06 15:02:00 "Sample 256-byte text file" "/dev/hd0a/foo.bin"
"DoinWEFOIWDLKSDfuhWEFu"
asus875p-bios r 200 04/04/06 15:02:00 "Sample 256-byte text file" "/dev/hd0/foo.bin"
"HWEF98whef8Yf8hwefo89jh"
asus875p-bios r 500 04/04/07 17:00:00 "Latest Programmed BIOS"
"/dev/hd0/test/asus875p.bin" "DoinWEFOIWDLKSDfuhWEFu"
asus875p-bios r 500 04/04/07 17:00:00 "Little text file" "/dev/hd0/foo.bin"
"DoinWEFOIWDLKSDfuhWEFu"

CMOS-FILE r 100 04/04/01 09:17:23 "CMOS Reload" "/dev/hd0/cmos.dat" "digest-string"
CMOS-FILE rw 100 04/05/10 09:17:23 "CMOS Checkpoint" "/dev/hd0/cmosbkp.dat"
"digest-string"
RTC r 100 04/04/01 09:17:23 "RTC Reload" "/dev/hd0/rtc.dat" "digest-string"

CONFIG.SYS rw 100 04/06/23 17:30:00 "CONFIG.SYS in root" "/dev/satafat2/config.sys"
"S4us/ymHeQfPqN/OYvGLUQ=="

ATA-DISK rw 100 04/06/24 17:40:00 "SATA Drive Partition Image" "/dev/ata/hd0b"
"digest-string"

USB-DISK r 100 04/04/01 09:17:23 "MS-DOS Disk Reload" "/dev/hd0/msdos.dat"
"digest-string"
USB-DISK r 100 04/04/01 09:17:23 "Win98 DOS Disk Reload" "/dev/hd0/win98.dat"
"digest-string"
```

## Default Security Authority User Authorization File

Since UPDATE provides remote access to objects in the system, uses the standard TCB user-level security to protect access to its services to only those users who are authorized. Below is a sample USERS.TXT file that would support the default security authority (the [TCB] system registry section points to this file) to enable users to access the Platform Update Facility remotely:

```
*** USERS.TXT - Firmbase User Authorization File for Integrated TCB.
#
# Functional Description.
# This file contains a list of users, passwords, and capabilities
# queried by the Firmbase TCB's integrated Security Authority when
# authenticating users and authorizing requests. This is a sample
# authorization file, and is intended to be a model on which a
# production authorization file might be based.
#
# This users.txt file should be located on /dev/rom. It authorizes
# users for the BASIC class (i.e., HTTPSRV HTTP 1.0 logins) and
# for the INTERNET realm (that is, users coming in over the internet,
# not local users.
#
# Modification history.
# S. E. Jones 04/03/16. #5.3, original.

USER guest
PASS gensw

RESOURCE HTTPSRV:SystemMenu*
RESOURCE TELNETD:*

RESOURCE BIOS:STRT # start system.
RESOURCE BIOS:BOOT # boot OS.
RESOURCE BIOS:BDRA # boot from drive A.
RESOURCE BIOS:BDRB # boot from drive B.
RESOURCE BIOS:BDRC # boot from drive C.
RESOURCE BIOS:BDRD # boot from drive D.
RESOURCE BIOS:BCDR # boot from CDROM.
RESOURCE BIOS:BWCE # boot Windows CE.
RESOURCE BIOS:BDOS # boot DOS in ROM.

RESOURCE HTTPSRV:UPDATE

USER root
PASS abracadabra

RESOURCE HTTPSRV:*
```



**Embedded Products**

915 118<sup>th</sup> Ave. SE, Suite 320, Bellevue, WA 98005, 800-850-5755, 425-576-8300  
[www.phoenix.com](http://www.phoenix.com), [embedded\\_sales@phoenix.com](mailto:embedded_sales@phoenix.com)