



# METEOR

## HIGH CAPACITY TERMINAL SERVER FOR MOBILE COMPUTERS

### PRODUCT OVERVIEW

#### Product Summary

Meteor is a High Capacity Terminal Server Product, enabling you to rapidly develop fast and stable applications for mobile computers using industry-standard programming tools.

Meteor-based solutions allow flexible and powerful integration of mobile devices into existing systems, including AS/400, Unix and Windows-based systems.

The speed and reliability with which Meteor achieves this can provide a very rapid return on investment. One customer reported a full ROI within 12 weeks of installation.

Key benefits of Meteor are:

- § Rapid application development using standard programming languages
- § Restart recovery from any system outage, without loss of work in progress
- § Able to support hundreds of HHTs from a single server
- § Model independence
- § Mobile printing support
- § Full device control
- § Pooled application instances

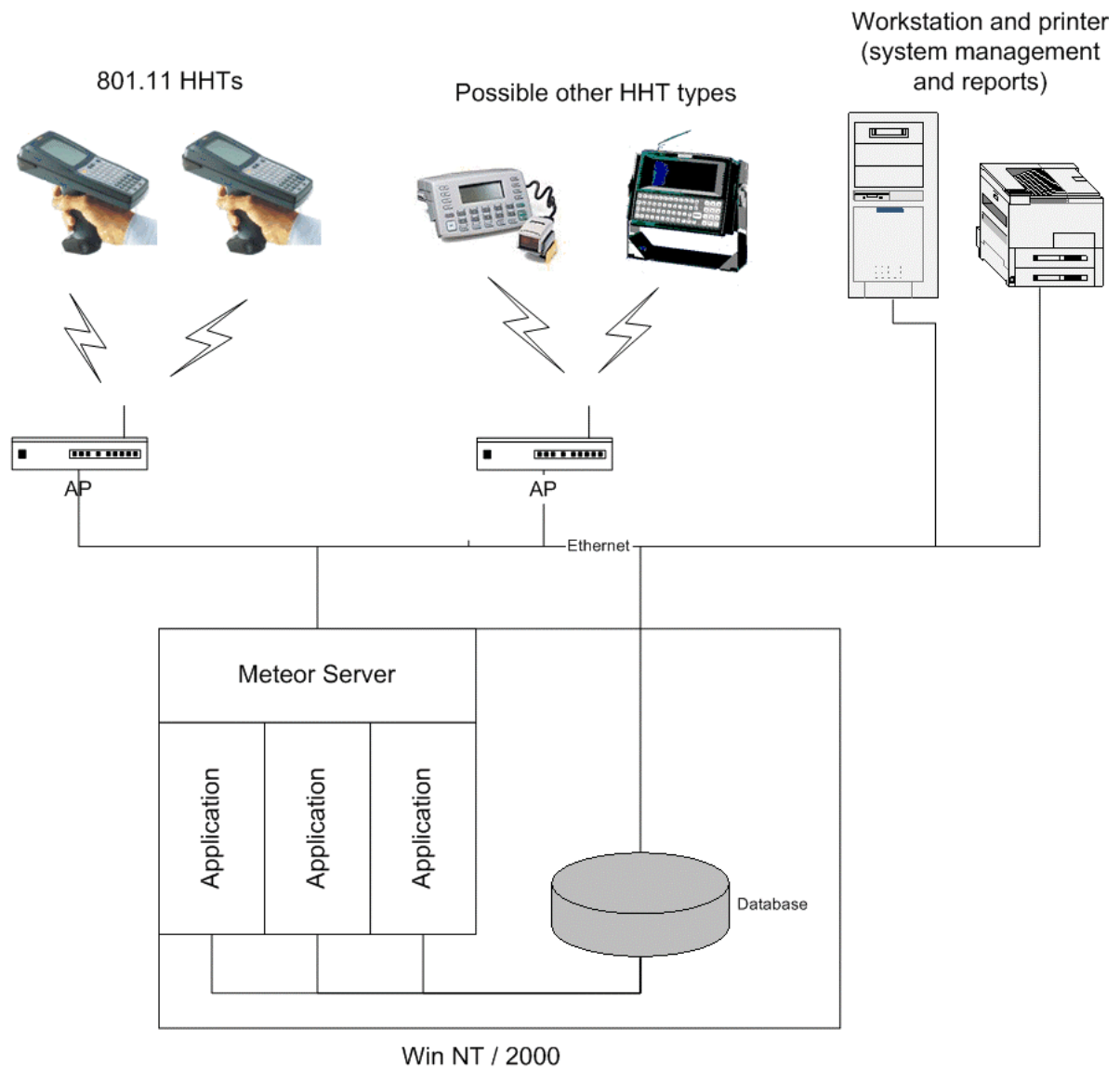


## Infrastructure

Meteor can operate within any IP network infrastructure, typically via 802.11b RF (2Mbit FH 802.11, GPRS and narrow-band RF can also be supported).

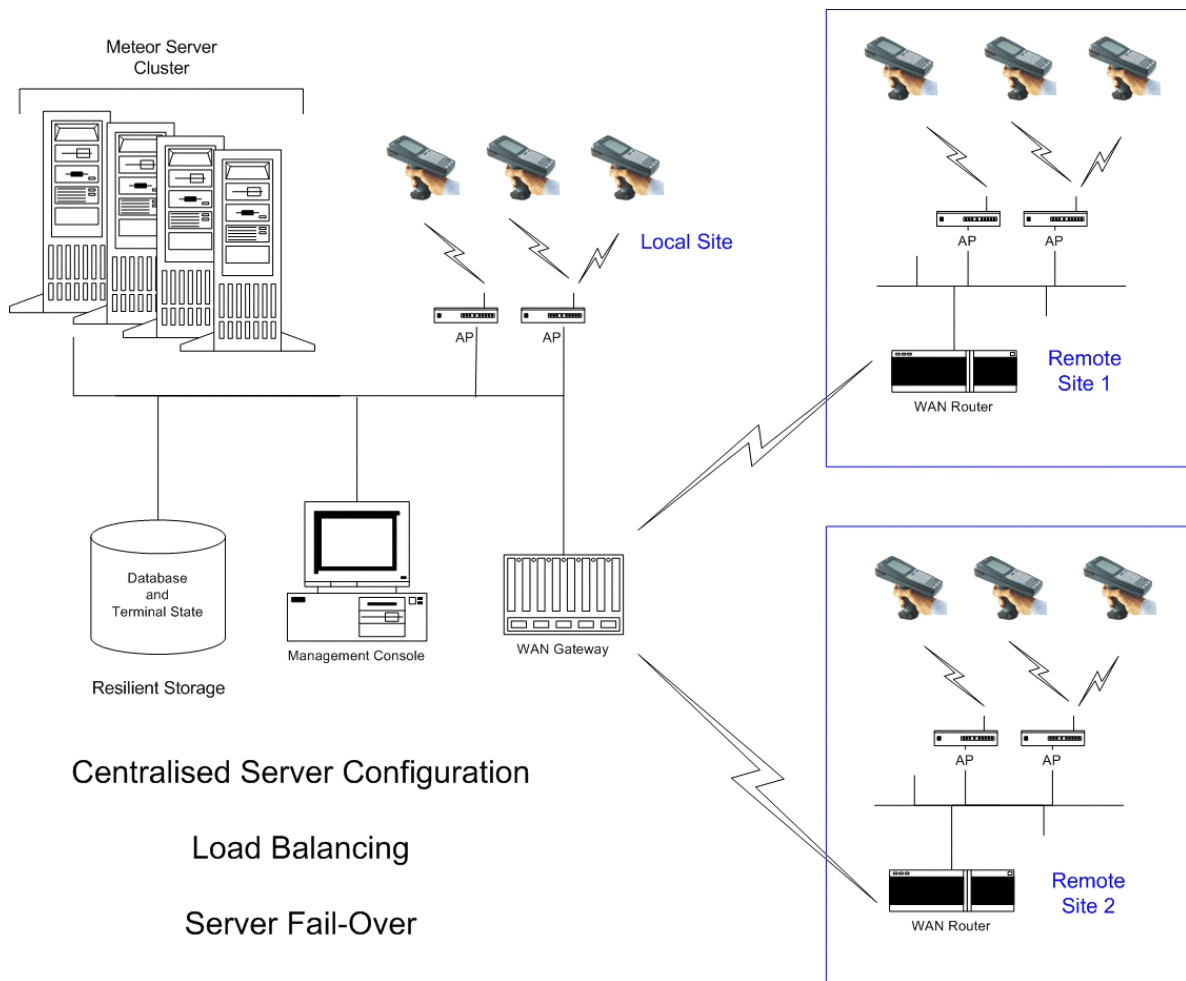
Example solutions are shown below:

### Single Site, Single Server



- § HHT application runs on the server; devices are thin clients
- § HHT application interfaces directly to the database (local or remote)
- § Business logic lies in HHT application on server
- § Multiple HHTs run from a shared pool of HHT application instances
- § Reports and system management run on separate PC workstation(s), attached to server database

### Multi-Site, Centralised Servers



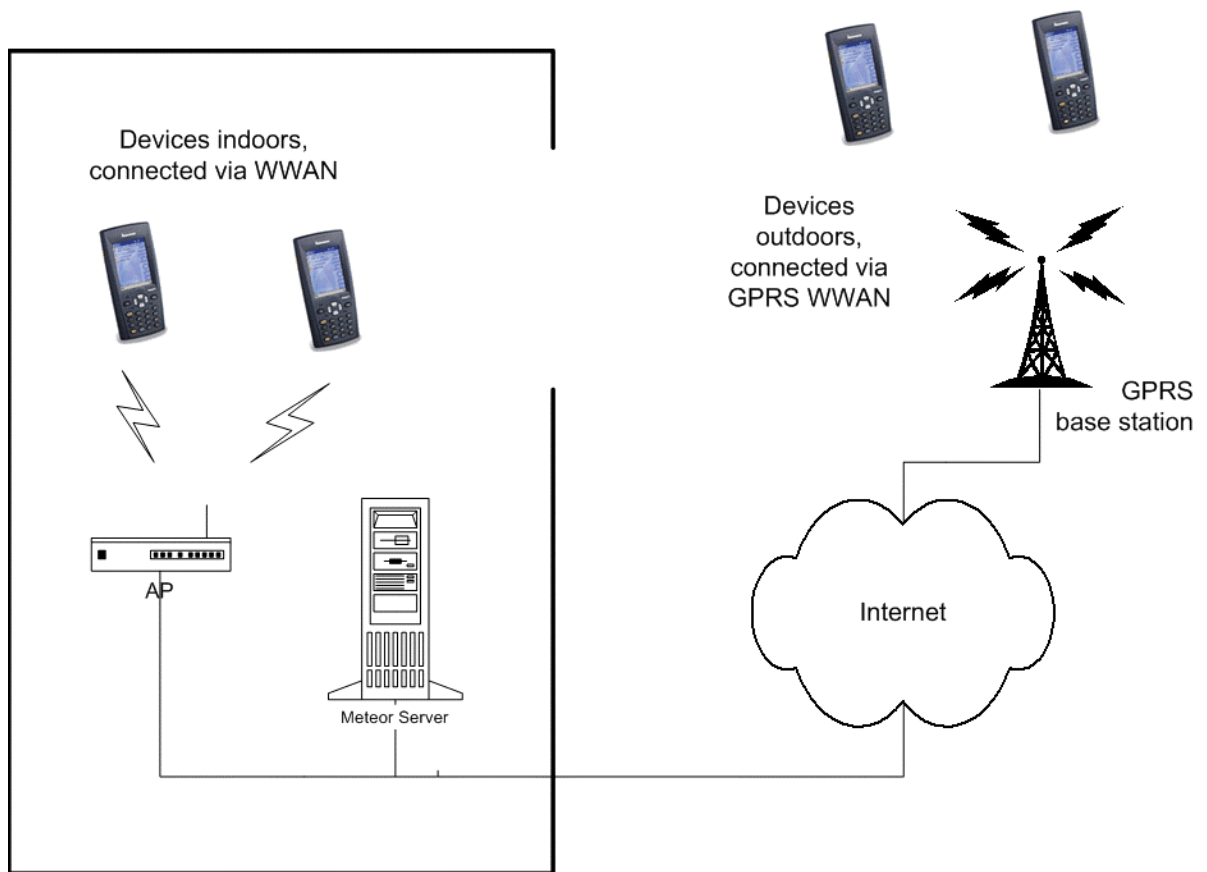
Each site potentially utilises hundreds of HHTs. Each server runs Meteor Server, and stores HHT context information to resilient shared storage. Devices can then move between servers (for load balancing or due to server failure), without losing their context.

Additional RF Access Points and Servers can be added in order to increase the data volume capabilities of the system.

Benefits of this architecture are as follows:

- § Server fail-over, providing transparent resilience in case of individual failure
- § Server load balancing, allowing busy areas of the warehouse to utilise system capability being released by less busy areas
- § Easier management and maintenance

### WWAN with Fall-Back to WLAN



While in use indoors, the devices communicate with the server via 802.11 RF. As they move outdoors, and out of RF coverage, Meteor Client automatically switches to GPRS communications, *without loss of connection or session context*.

On return to RF coverage, the connection returns to 802.11.

Benefits of this architecture are as follows:

- § Coverage of wide areas, such as yards and outdoor storage facilities, without the need to cover all areas with 802.11 RF, or
- § Continued operation outside of normal coverage area, for occasional use

## Product Benefits

### Rapid Application Development

Meteor-based applications are hosted on the server(s), offering the following benefits:

- § Development of application is carried out using main-stream modern development tools (VB, VC++, any .NET tool).
- § An input screen can be created in minutes (see [Programming Outline](#) for an example).
- § Direct connection to local databases or remote systems (since the application runs on the server itself)
- § Ease of distribution and upgrade – new HHT applications can be distributed without upgrading the device itself
- § All features of the HHT hardware (display, keyboard, scanner, audio and printing) are handled by Meteor and presented as a simple, consistent interface
- § No need to develop a separate data interface server.
- § Full transaction and event logging, for problem resolution and audit purposes.

### Restart Recovery from any type of System Outage

Meteor-based systems can restart **without loss of data or session context** from any of the following types of outage:

- § HHT failure – reboot of device, or switch to alternative device
- § RF network outage – devices restart as soon as service is restored
- § RF network ‘brown spot’ – device will automatically resume when device moves to an in-coverage area
- § Server failure – system restarts on server restart
- § Server failure – clustered servers can be utilised to provide automatic fail-over on individual server failure (also provides load balancing)

The guaranteed message delivery protocol employed by Meteor ensures that events from the device are always received and only ever processed once.

### Support Hundreds of HHTs from a Single Server

The design of Meteor allows large numbers of HHTs to be run from a single server.

As a guideline, a standard 2GHz Pentium 4 PC with 256MB of RAM is capable of supporting in excess of 400 devices simultaneously. Obviously performance varies depending on the processor and memory requirements of the server-based applications, the frequency of the transactions and on the other processes running on the server.

Higher specification machines or clusters of machines can be used to provide greater capacity.

For a data collection application to be effective, a typical maximum response time of one second (scan to scan) must be achieved. Meteor-based systems typically respond in a considerably shorter time than this, although this is of course affected by server application response time (including database delays), RF and wired network latency, and of course by the performance of the remote device itself.

An example performance profile is as follows:

Server hardware	Pentium III 600MHz 192MB RAM
Operating system	Windows 2000 Professional
Client devices	153 attached and active client devices
Application instances	10 instances of an application required to service these devices
Transaction rate	Average of 1600 transactions per minute (one transaction per device every 6 seconds). Peak of 2000 transactions per minute.
Application response time	Between 5 and 15 hundredths of seconds per transaction
Device response time	Consistently less than one second scan to scan
CPU usage	Generally between 40 and 70% (100% at peaks)
Network usage	0.1% on a 100mb/s link

## Model Independence

Meteor-based applications will run on any supported client type (see [Mobile Devices](#)), with little or no changes to the HHT application on the server.

- § Meteor Client is pre-installed on each device, making all devices type appear identical to the server, and thus to the application
- § Provides future proofing – different hardware can be added to the system without any software re-engineering
- § Mobile printers are also supported in a device independent fashion

## Mobile Printing Support

Meteor can print to mobile or fixed mount printers via IP (802.11b or wired), serial, Bluetooth or Symbol PAN, or to integral device printers. Any manufacturer's printers can be supported, although the system has been specifically validated with Zebra and Paxar printers.

## Full Device Control

Meteor allows the application on the server to control all aspects of the device, including:

- § Screen layout
- § Entry field format control
- § Full barcode scanning control (enable/disable, symbology settings, etc)
- § Audio feedback
- § Inactivity timeout

## Pooled Application Instances

Meteor is usually configured to 'pool' application instances on the server, so that 200 devices would require only (for example) 20 instances of application.

- § Reduced server load (lower hardware spec)
- § Reduced number of database connections
- § Dynamic instance pool management
- § Large implementations – a single server can support hundreds of devices
- § Centralisation of servers – low bandwidth usage provides a rapid response even across a WAN.

## Platform

### Mobile Devices

Meteor supports mobile devices from a number of manufacturers, including Symbol, Intermec and Paxar.

Device independence is achieved by ensuring that all devices offer the same user interface and programming API regardless of whether they use WindowsCE, PocketPC, DOS, PalmOS or a proprietary operating system.

This provides full future proofing of your applications, and offers a consistent user interface, even in a mixed hardware environment.

### Mobile Printing

Meteor provides full support for mobile printers, regardless of manufacturer or connectivity. Supported connectivities include 802.11b, serial, BlueTooth, integral and Symbol PAN.

### Server

Meteor will operate on Windows NT, 2000 or XP. Desktop or server editions may be used. Clustering can be achieved without relying on Server edition clustering.

## Licensing

Meteor is licensed on a per-device basis, regardless of the type of mobile device. Device types can be freely mixed in a single population.

## Case Studies

### Shoe Zone (Picking and Despatch)

**Warehouse picking, loading and despatch.**

- § Meteor based
- § Handles 800,000 pairs of shoes per week
- § Despatches to over 400 stores
- § 80 pickers using Symbol WS1040s
- § Labelling to belt mount and fixed printers
- § Code start to system live in seven weeks

### Focus (In-Store)

**Replaced two legacy HHT systems with a Meteor-based system.**

- § Meteor's flexibility allowed the legacy host interfaces to be maintained
- § No changes to host systems required
- § Maintained near-identical look and feel
- § Minimal user retraining required
- § 200 stores, 3 terminals per store
- § Small server footprints allows Meteor Server to run on existing in-store server

### Anlisco plc (Warehouse Management)

**Warehouse put away, stock audit and management functions, interfacing to customer's existing ERP system.**

- § Multi-site warehouse
- § Devices running into single server, across LAN and WAN
- § Real-time interface into existing ERP system
- § **12 week payback on investment**, based on Anlisco's own internal ROI calculations

### Marks and Spencer (Repricing)

**Re-labelling of goods with updated pricing within the distribution centre.**

- § Rapid scan and print operation
- § Belt mounted Monarch printers
- § Different label sizes and formats available for different price and product types

## Programming Outline

Meteor applications are event driven, that is the program responds to events triggered by the user of the remote device.

Event types include Key Press, Barcode Scan, Field Entry, etc.

An application program is usually written as a state machine, that is, a separate discreet program function is executed depending on the current position or 'state' of the process. For example, a barcode scanning prompt might be one state, and quantity entry another.

### Displaying a Screen

A typical section of code (in VB6) to set up a barcode scanning screen might be as follows:

```
Dim aiExitKeys(0 To 0) As Integer      ' Array for loading with exit keys

' Set up the screen for this state
oMeteor.ClearScreen
oMeteor.DisplayText 0, 0, "    PRICE MARKDOWN    "
oMeteor.DisplayText 0, 1, "ITEM:"
oMeteor.DisplayText 0, 4, "NORMAL PRICE:"
oMeteor.DisplayText 0, 6, "    NEW PRICE:"

aiExitKeys(0) = K_CLEAR
oMeteor.EntryField 6, 1, "#####", "BM1", "", aiExitKeys
```

or in VC++:

```
const short aiExitKeys[] = { K_CLEAR };

oMeteor.ClearScreen();
oMeteor.DisplayText(0, 0, _T("    PRICE MARKDOWN    "));
oMeteor.DisplayText(0, 1, _T("ITEM:"));
oMeteor.DisplayText(0, 4, _T("NORMAL PRICE:"));
oMeteor.DisplayText(0, 6, _T("    NEW PRICE:"));

oMeteor.EntryField(6, 1, _T("#####"), _T("BM1"), _T(""),
                  aiExitKeys, _countof(aiExitKeys));
```

You can execute methods on an object that represents the remote device (in this case *oMeteor*). Any methods cause the corresponding process to be executed on the device itself. Methods are batched together automatically by Meteor and sent to the client in a single message.

The 'BM1' specifier in the parameters to *EntryField* specify entry via barcode scanner, with a minimum length of 1.

## Processing the Event

Three possible events may occur as a result user actions at this state; BarcodeScan or FieldEntry for scanned or keyed data, or KeyPress if the exit key (CLEAR) is pressed. The code to handle this might look as follows (VB6):

```
' Process the event from the screen
If eEventType = eFieldEntry Or eEventType = eBarcodeScan Then

    If Len(sData) <= 13 Then
        oPersistence.Data("Barcode") = sData
        If bLookup(sData) Then
            oPersistence.Data("CurrentState") = ePriceMarkdownEnterPrice
        Else
            oMeteor.PopUpMessage "UNKNOWN ITEM", 1
            ' Fall through to re-do this state
        End If
    Else
        oMeteor.PopUpMessage "INVALID CODE", 1
        ' Fall through to re-do this state
    End If

ElseIf eEventType = eKeyPress And Val(sData) = K_CLEAR Then
    oPersistence.Data("CurrentState") = eMainMenu

End If
```

or in VC++:

```
switch (eEventType)
{
    case eFieldEntry:
    case eBarcodeScan:

        if (sData.GetLength() <= 13)
        {
            oPersistence[P_BARCODE] = sData;
            if (bLookup(sData))
                eCurrentState = ePriceMarkdownEnterPrice;
            else
                oMeteor.PopUpMessage(_T("UNKNOWN CODE"), 1);
        }
        else
        {
            oMeteor.PopUpMessage(_T("INVALID CODE"), 1);
        }
        break;

    case eKeyPress:
        if (_ttoi(sData) == K_CLEAR)
            eCurrentState = eMainMenu;
        break;

    default:
        break;
}
```

The *oPersistence* object is a data persistence object provided by Meteor to persist application data between states. It is the use of this object that allows Meteor to share program instances between devices, and to allow devices to switch to alternative servers, without the loss of session context.

## Summary

You can see how simple, quick and powerful Meteor's programming API can be. Similar methods exist for controlling the barcode scanner, for printing, audio feedback, etc. Methods also exist for writing trace files of application activity to central log files, for audit and field problem determination.

In addition, since the program code is running on the server, you also have full access to the vast array of powerful programming interfaces available to Windows programmers, including database access, XML interfacing, network communications, even serial comms and file I/O, all in *real-time* direct from your mobile computer application.

All of this is achieved without having to worry about RF networking, message transfer, heterogeneous devices, mobile printer connectivity, restart recovery, server fail-over; it's all there in Meteor, ready to use.