About This Document

This document discusses some of the operational and technical details of the NetMotion Mobility XE® Mobile VPN. It is particularly useful to network administrators who require a deeper understanding of how Mobility XE functions before deploying it in their environment. NetMotion Mobility XE and its technology are protected by copyrights and patents both issued and pending, in the U.S. and in other countries.

An Overview of the Mobility XE Architecture

Mobility XE is a highly scalable, software-based mobile VPN. It supports both active/active and active/passive high availability, and works with standard network infrastructure including routers, switches, and firewalls. There are two main components of the Mobility XE system: the Mobility server and the Mobility client. They communicate using remote procedure calls (RPC) and the Internet Mobility Protocol (IMP), which run on top of the User Datagram Protocol (UDP).

Mobility Server

The Mobility server manages wireless network connections for mobile devices. It resides on the local area network, on a machine running either Windows Server 2003 or Windows Server 2008 R2.

The Mobility server acts as a transport-level proxy for each mobile device running the Mobility client. The server maintains the state of each client and handles the complex session management required to maintain continuous connections to systems that host network applications. When a mobile device becomes unreachable, suspends operation, or moves to a different network, the Mobility server maintains the connection to the client’s peer applications by acknowledging receipt of data and queuing requests.

The Mobility server also manages network addresses for the mobile devices. Each Mobility client receives a virtual IP address on the wired network, typically obtained from DHCP (Dynamic Host Configuration Protocol), or assigned from a range of addresses reserved for this purpose on the Mobility server. Mobility XE also supports the ability to statically assign a virtual IP address to an individual device or user.

Multiple Mobility servers can operate as a server pool with failover and load balancing capabilities. The discussions in this document assume an installation with a single Mobility server. The Mobility server provides tools and metrics that a system administrator can use to configure and manage remote connections and troubleshoot remote connections. A web-based interface allows system administrators to configure Mobility XE settings and manage the server from another PC on the network.

Mobility Client

The Mobility client software resides at the transport driver interface (TDI) layer on supported Microsoft platforms and performs indirection and redirection of application network calls. When an application wishes to use the network, the TDI calls are intercepted, the parameters are marshaled, and the call is forwarded for execution on the Mobility server. The client works transparently with operating system features to allow client-side application sessions to remain active when the device loses contact with the network.

A Mobility client device is a standard mobile device or off-the-shelf computer running Windows 7, Windows Vista, Windows XP, Windows XP Tablet, Windows Mobile or Windows CE.
Application Persistence

Application persistence is achieved by having the highly available Mobility server execute the marshaled network calls on behalf of the client. When a client goes into a coverage gap, suspends or hibernates, the server maintains the TCP connection with the application server or host system. While in this state, applications on the client see pending TDI calls and simply wait until the call is completed. Applications running on the Application server or host system see a reliable TCP connection provided by the Mobility server. The network calls are completed when the client leaves the coverage gap and resumes communication with the mobility server.

Other Infrastructure

Installations running the Analytics Module require a reporting server and reporting database. Mobility also integrates with external authentication services.

Operation

Remote Procedure Call and Internet Mobility Protocols

Mobility XE's Remote Procedure Call (RPC) protocol and Internet Mobility Protocol (IMP) form the technological backbone that connects the Mobility server to each mobile device.

A remote procedure call is a way of allowing a process on a local system to invoke a procedure on a remote system. With Mobility XE, the client's network calls are sent to the server for remote execution. If Mobility operated at the Winsock layer, these would be calls such as open socket, bind, connect, send and receive. Because Mobility XE operates at the TDI layer, the TDI equivalent of these calls is what is sent to the server for remote execution.
The application on the local system is not aware that the procedure call is executed on a remote system. The strength of Mobility XE’s RPC-style approach is that it allows the mobile device to go out of range or suspend operation without losing active network sessions. Because session maintenance does not depend on customizing or rewriting applications, off-the-shelf applications will run without modification in the wireless environment.

The RPC protocol is encapsulated by the Internet Mobility Protocol (IMP), which is encapsulated in UDP. The Internet Mobility Protocol compensates for differences between wireline and less reliable networks by adjusting frame sizes and protocol timing to reduce network traffic. This is important when bandwidth is limited, there is high latency, or when battery life is a concern.

Mobility XE also strengthens data security by encrypting all traffic between the Mobility server and clients, and allowing only authenticated devices to connect to the Mobility server. See Security for Wide Area Wireless Networks for more information about Mobility XE security.

**Device Registration**

When a Mobility client connects to a Mobility server for the first time, the server registers the mobile device’s permanent identification (PID) number, a unique number that the client will use for all subsequent connections. This registration occurs only on the first connection and does not require any action by the user or administrator. The identification number is stored in the client system registry and in the Mobility warehouse (using Lightweight Directory Access Protocol, or LDAP).

The Mobility server stores the PID based on the computer name. As long as the client computer name does not change, the server can restore the PID to the client device even if the client registry is lost. This may happen, for example, if the client device’s hard drive is re-formatted in order to re-install the operating system. When the Mobility client is re-installed and connects, the server searches for a match on the device name. If it finds a match, it will restore the same PID to the device.

**Device Connection**

At the administrator’s option, Mobility may first require the presence of a properly provisioned X.509v3 digital certificate on the device before allowing a connection. This confirms that the device is trusted, so that the client and server may establish a secure tunnel prior to user authentication. Enterprises that use unencrypted user authentication methods may enable device-level authentication to ensure that the user credentials pass through a secure tunnel. In addition, through device authentication, Mobility XE can maintain a secure tunnel while a user is not actively logged in. This is described further below under Device Authentication and Unattended Mode.

Once the Mobility client establishes a connection to the Mobility server, it then prompts the user to authenticate. If authentication fails, the device is disconnected.

The mobile worker can use standard Windows logon credentials to authenticate to the network. The Mobility server authenticates that user against the enterprise’s domain using NTLMv2 for native Microsoft authentication, RADIUS authentication, or RSA SecurID authentication. When configured for RADIUS authentication, Mobility XE uses RADIUS PEAP or EAP-TLS protocol. This enables support for strong user authentication to public-key infrastructure (PKI) using smart cards and/or user certificates, as described under the Advanced Authentication section.
For a Microsoft deployment, a three-way handshake occurs between the Mobility client and server:

- The client sends a list of the supported authentication types. This packet includes the NTLMv2 HELLO packet.
- The server responds with an NTLMv2 challenge.
- The client completes the authentication with the response to the challenge.

After authentication, the server and client exchange signed ECC (elliptic-curve cryptography) public keys and related cryptographic materials to perform the Diffie-Hellman key exchange and create the secure VPN tunnel. Symmetric keys are derived from the public keys; these are not transmitted. This applies to all supported authentication methods. If the deployment supports unattended mode, the administrator has the choice of configuring Mobility XE to persist the VPN tunnel after a user logs off, or to tear down the tunnel and create a new one.

**Virtual IP Addresses**

Each Mobility client has a virtual IP address on the wired network, obtained from DHCP or assigned from a range of addresses reserved for this purpose on the Mobility server. In addition, static virtual IP addresses can be assigned to specific devices or users. For each active client, the Mobility server relays data directed to the client’s virtual address to its current, actual address (the point of presence, or POP address). While the POP address of a Mobility client may change when the device moves from one coverage area to another, the virtual address remains constant for the duration of the session.

**VPN Tunnel Persistence**

Unlike IPsec VPNs or SSL VPNs, the Mobility XE VPN does not require a fixed local address. The tunnel is maintained between the Mobility server, which is at a fixed address, and the Mobility client, which can have an ever-changing POP address. By mutual agreement, the client and server maintain the secure tunnel until one endpoint issues a disconnect: this could happen when the user logs off, the administrator quarantines the device, or a configurable link inactivity timeout occurs.

- The tunnel remains available and application sessions persist in any number of scenarios:
  - Suspending operation on the mobile device and later resuming it
  - Moving to a different location on the network
  - Connecting a mobile device over slow, bandwidth-challenged, or high-latency networks
  - Encountering interference from microwaves, stairwells, elevator shafts — anything that interferes with radio signals
  - Changing network interfaces (for example, from a WLAN to a WWAN card)
  - Moving across gaps in coverage

The configurable timeout ensures that the resources on the Mobility server consumed by inactive sessions are not consumed indefinitely. But in test scenarios, devices have been suspended in the middle of an application transaction, awakened a week later, and the transaction resumed exactly where it left off.
User Re-Authentication

Periodic user re-authentication confirms that the user is still in possession of the device. Through settings at the Mobility console, the administrator controls the interval between re-authentications, the grace period during which the user is expected to respond, and whether or not the user must re-authenticate when the device resumes after suspending or hibernating. Mobility maintains the secure tunnel and application sessions throughout the challenge process and grace period. If the grace period lapses and the user fails to re-authenticate, Mobility blocks all further network activity. Re-authentication supports all the authentication methods that Mobility supports, including those detailed under the Advanced Authentication section below.

Managing the Connection State

Mobility XE does not “test” to see if a device is reachable before sending — it simply sends the required data and looks for a return acknowledgement. If the return acknowledgement isn’t received, it will re-send for a number of tries, then back off, conclude the device is unreachable, and re-send at a later time.

To determine whether an inactive mobile device is reachable, the Mobility XE system uses keep-alives: the Mobility client periodically sends frames to the Mobility server. The frequency of these keep-alive frames is user-configurable and may be decreased to reduce traffic on bandwidth-constrained networks. If the Mobility server fails to receive expected keep-alive frames during the configured timeframe, it indicates that the device is unreachable in the server’s management console. Keep-alives are sent only in the absence of application or other traffic.

Device Is Unreachable

Mobility XE’s use of RPC preserves the state of interrupted data transfers and holds pending data in queue.

When the application server is transmitting data to the Mobility client and the client becomes unreachable due to a disruption in the network or suspension of the device, Mobility XE’s flow control algorithms notify the Mobility server’s RPC layer to stop accepting data from the application server. The Mobility server’s TCP buffers fill, resulting in the TCP window size adjusting to zero (0), which in turn notifies the application server that data transmission should be paused until the Mobility client is able to receive data. In this state, the Mobility server and application server exchange TCP acknowledgements to keep the connection alive indefinitely or for a length of time configured by an administrator.

When the client becomes reachable again, the RPC layer receives a flow control indication notifying it that the data transfer may continue. The RPC layer then resumes taking data from the TCP stack, causing the TCP connection’s receive buffers to empty. When buffer space becomes available, the TCP stack sets the connection’s receive window to a non-zero value indicating to the application server’s TCP stack that it is again ready to receive data, and the transfer continues.
Device Has Roamed
Mobility XE uses DHCP to detect and facilitate “roaming” — when mobile devices move across a subnet boundary or to a different network. The user does not need to restart the system or close existing network connections to obtain a new address.

Subnet Roaming
Mobility XE allows mobile devices to cross subnet boundaries without losing their network connection or application sessions. To do this, the Mobility client must be able to detect that it has moved to a new subnet and must be able to obtain a new point of presence address from a DHCP server.

InterNetwork Roaming™
A Mobility client can maintain a connection and application sessions when moving between different network media. If network interface cards for different types of network connections have been installed and properly configured on the mobile device, Mobility XE provides application persistence as the client moves between a LAN, a wireless LAN, and a wireless WAN — or any other type of IP-based network. No additional configuration is required on the Mobility client or server.

The operating system and network hardware on the mobile device determine how much user intervention is required for InterNetwork Roaming™. A Mobility client with multiple active network interfaces will switch from one interface to another automatically. For example, if a client device with both WLAN and WWAN cards goes out of range of any wireless LAN access point, communications may automatically transition to the WWAN medium.

On a client device with multiple network interface cards simultaneously active, Mobility XE uses whatever interface the operating system is using. The operating system’s interface selection may depend on the order in which the interfaces become active, and may not always result in the use of the “best” interface. The Mobility server modifies the metric for a defined route in the client operating system based on the speed of the network interface, so the mobile device uses the available interface with the greatest bandwidth. If the Roaming — Use Fastest Interface option is disabled, the mobile user may have to manually stop and start interface cards, depending on the operating system.

Sometimes, network connections are available via two interfaces, but the Mobility server can be reached only via one of these networks. Mobility XE’s dead gateway detection allows the client to connect even when the preferred interface is available, but cannot connect to a Mobility server. More detailed information on roaming detection and behavior can be found in the Mobility XE System Administrator Guide.

NAT (Network Address Translation)
Because Mobility uses UDP, it is not susceptible to core problems common to IPSec VPNs and NAT. IPSec VPNs are forced to implement NAT-T (NAT traversal — see RFC 3947), which encapsulates the IPSec ESP packets in UDP in order to traverse firewalls or routers that are not NAT-friendly. Because Mobility XE is NAT-friendly by design, it does not require the additional layer of encapsulation and overhead required to traverse the intermediary nodes between the Mobility client and server.
Advanced Authentication

Mobility XE supports multiple authentication methods. In addition to a user name, password or PIN (something the user knows), Mobility implements two-factor and multi-factor authentication. These require a certificate, key fob, authorized/provisioned device, etc. (something the user has); and physical confirmation of user identity supplied by a fingerprint reader (who the user is). Supported methods include:

- Security tokens (such as an RSA® SecurID key fob)
- Smart cards (read by a card reader), including those that use a biometric scanner to read a user’s fingerprint in place of a PIN or password
- User certificates stored in the Windows user certificate store
- Device certificates stored in the Windows local computer certificate store

RSA SecurID

RSA SecurID authenticates users based on their ability to enter a frequently-changing string of numbers and letters generated by a token generator, after first entering their personal identification number (PIN). The token generator can be a hardware generator — a key fob or USB token that displays a time-based code which the user enters manually. Or it can be a software application running on their device that outputs the string.

In an RSA SecurID installation, the RSA Authentication Agent software runs on the Mobility server. It encrypts the supplied credentials with a one-way hash, and then passes them on to the RSA Authentication Manager which runs on a separate server. The RSA Authentication Manager handles the task of verifying the credentials and passing the authentication status back to the Mobility server. Mobility XE also supports periodically re-establishing a new SecurID PIN (New PIN Mode) for provisioning new tokens, as well as entering successive token codes (Next Token Mode). The Mobility XE product has achieved RSA Secured Partner Program Certification for the RSA SecurID product. This certification signifies that a technical partnership has been established to increase security for joint customers using both products.
Public Key Infrastructure/RADIUS

Mobility XE supports standards-based two-factor authentication using:

- Digital user and device certificates,
- PKI (for certificate validation)
- RADIUS Extensible Authentication Protocol (EAP) authentication.

Mobility XE’s use of RADIUS EAP is a widely used embodiment of the 802.1x security standard. The PKI technology is built into Microsoft operating systems as well as many other third-party vendors’ solutions, and it supports the X.509v3 international standard for digital certificates.

To meet the requirements of PKI-based authentication, the Mobility clients and RADIUS server(s) must have certificates installed for mutual authentication. The RADIUS server has the following installed and configured:

- X.509v3 digital certificate and private key
- Certificate for the trusted Certificate Authority (CA) that signed the certificates on the client devices

The Mobility clients have the following installed and configured:

- Certificate for the trusted Certificate Authority that signed the RADIUS server’s certificate
- X.509v3 certificate installed and configured, or a smart card provisioned with a valid certificate deployed through the enterprise’s Certificate Authority
- For smart cards, a reader that supports the Microsoft cryptographic service provider (CSP)

The Mobility server acts as a Network Access Server (NAS) in the RADIUS security system. The EAP-TLS authentication protocol is passed from the client through the RADIUS server to the authentication server. If the client uses password-protected digital certificates stored on the hard drive, the user must enter the certificate password to unlock them. If the certificate is stored on a smart card, the user must enter the associated PIN. The Mobility server and Mobility client then create a mutually-authenticated secure tunnel protected by the x.509 certificates.

The Mobility server passes the user credentials to the RADIUS server. The RADIUS server completes the authentication by validating the certificate with the certificate authority. If the RADIUS server authenticates the credentials, it notifies the Mobility server, authorizing user access to Mobility XE services. If the RADIUS server is unable to authenticate the certificate, it rejects the authentication request and the Mobility server terminates the connection attempt.

The two-factor authentication method supports biometric systems, such as fingerprint scanners, when those systems are used in place of a password to unlock access to stored certificates.

Device Authentication and Unattended Mode

Device authentication enables the device and Mobility server to establish an encrypted VPN tunnel based on device certificates prior to authenticating the user. Device authentication uses the RADIUS EAP-TLS protocol and requires signed X.509v3 certificates installed on each device as well as on the RADIUS server. If either the device authentication or a subsequent user authentication fails, the device is disconnected.
Device-based authentication can be loosely or tightly tied to a user’s authentication. Each user may be assigned one or more specific listed devices, and only user logins from those devices are explicitly allowed.

Because device authentication occurs independent of the user login, administrators can be assured that a device is authorized and connected via a secure tunnel, even when a user is not actively logged in. This allows device management after-hours for applying security patches and software updates using device-management solutions such as Microsoft’s Active Directory Group Policies, Microsoft SC-CM, Sybase Afaria and many others. Policy Management settings associated with the device and Network Access Control (described below) remain operational in unattended mode. When unattended mode is enabled, it should always be used in conjunction with enforcement through Policy Management to restrict the applications that may access the network.

Mobility offers four authentication modes that define how device authentication works in concert with user authentication. These can be assigned globally, for classes of devices, or individual devices.

For devices that do not have a pre-desktop mode (Windows Mobile and Windows CE), Mobility XE supports unattended mode if the device is configured to display the prompt for logon credentials on the full screen.

<table>
<thead>
<tr>
<th>Device-Authentication Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User authentication only</td>
<td>The system performs only user authentication. Device authentication is not attempted even if there is a valid device certificate installed.</td>
</tr>
<tr>
<td>Multi-factor</td>
<td>The system performs both device authentication and user authentication before a VPN connection is established. When user authentication is configured to require a username and password or other single-factor authentication method, Mobility uses the device certificate to perform a true two-factor authentication. When combined with other two-factor user authentication schemes (smart card, user certificate, RSA SecurID), Mobility can perform three-factor authentications.</td>
</tr>
<tr>
<td>Unattended</td>
<td>The system establishes a VPN tunnel after successfully authenticating the device, maintaining a secure tunnel for management while a user is not logged in. If a user accesses the desktop as a continuation of the boot-up sequence or on a subsequent log-in, the user must authenticate successfully or else the device is disconnected. After the user logs off, Mobility XE can either preserve the existing tunnel or force the device to re-authenticate, based on the client settings.</td>
</tr>
<tr>
<td>User required/Device optional</td>
<td>If the device is configured for device authentication, the system authenticates the device, but will still allow user authentication to proceed and the VPN to connect so long as the user authentication is successful. This is a test mode which allows the administrator to transition to device authentication and confirm success without adversely impacting users.</td>
</tr>
</tbody>
</table>
Link Optimizations

There are many behaviors of TCP/IP that make it less than optimal in a wireless environment. To address these, Mobility XE is designed to provide optimum performance over intermittent and bandwidth-challenged network links. Its architecture includes enhancements that allow network traffic over IP to more effectively deal with loss of connectivity from a mobile device, whether due to coverage outages or external factors, such as power management or user intervention. It makes the most efficient use of the given bandwidth using many advanced features that reduce the “ chattiness” of transport protocols:

- Selective acknowledgments
- Data and acknowledgment bundling
- Message coalescing
- Reduced and synchronized retransmissions
- Fragmentation optimizations
- Data compression
- Error-reduction algorithms
- Web acceleration

Working in concert with each other, these advanced features provide for the most efficient movement of data. In addition, when “Use Fastest Interface” is enabled (the default), Mobility XE automatically switches to the fastest bandwidth network connection when multiple connections are active.

Below are descriptions of how these features help in a wireless environment, especially a wide-area one.

Selective Acknowledgment Algorithm

When transmitting data over a wide area wireless connection, packet loss can have a severe impact on performance. When either side is transmitting a train of packets, it is quite possible for some of the frames to “evaporate” and never reach their ultimate destination.

Mobility XE takes packet loss into account when sending and receiving data. Using sequence numbers, Mobility can detect whether a frame has been received out of order. If it receives a frame carrying a sequence number greater than the one it is expecting, it alerts the transmitting peer that it did not receive one or more of the frames in the sequence. The transmitting peer then retransmits the missing frame(s).

In contrast, implementations without selective acknowledgment not only retransmit the missing frames, but may also re-send the frames that were correctly received by the peer. These gratuitous retransmissions waste bandwidth and processing resources.

Message Coalescing, Data and Acknowledgment Bundling

Most standard network protocol implementations use acknowledgment frames as feedback to the sender that the peer has successfully received the sent packet(s). The feedback is continuous: every other frame (at most) is acknowledged, generating a stream of small control frames in the reverse direction. These extra frames, though small, can add up to significant overhead.
Mobility XE can reduce the reverse flow of information from every other frame to one in four or greater. It does this by using its selective acknowledgment policy, and adjusting its metrics for determining network latency. This allows more application-level data to be transmitted instead of using the bandwidth for additional control information.

To further reduce bandwidth consumption, Mobility XE uses a message bundling (or multiplexing) technique. With this algorithm, both control and application data from multiple distinct message streams are passed in the same frame. Without this feature, information is encapsulated in its own frame when an application transmits data. As more connections are established, more protocol overhead is required. By bundling these streams together, Mobility allows more application data to be transferred in the same amount of space, making significantly better use of bandwidth.

Another benefit of this technique is that it allows each frame to carry the maximum amount of payload. In this example, two applications send and receive data in a non-enhanced implementation as follows: Application A sends 100 bytes of data to its peer, and Application B sends 150 bytes to its peer.

Here is what this looks like on an IPSec VPN (SSL VPNs have similar challenges):

```
<table>
<thead>
<tr>
<th>Application 1</th>
<th></th>
<th></th>
<th>Data</th>
<th>ESP-T</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP</td>
<td>IP</td>
<td>ESP-H</td>
<td>IP</td>
<td>TCP</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>16</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>100</td>
<td>6</td>
</tr>
</tbody>
</table>
```

Legacy VPNs generate separate frames with protocol overhead for each application.

This normally generates two separate frames, one with 100 bytes of data (plus protocol overhead), and one with 150 bytes (plus protocol overhead). Each frame then requires a separate acknowledgment from the peer stating that it was received, so a total of four frames are required to move the 250 bytes of data over the wireless link.

Here’s what happens with Mobility XE using the same example: The data from application A (100 bytes) and application B (150 bytes) traverse the link in one 250-byte packet (plus protocol overhead). Only one acknowledgment is generated, so only two frames are required to move the same 250 bytes of data. Mobility becomes even more efficient as more applications transmit data.

```
<table>
<thead>
<tr>
<th>Both Applications</th>
<th></th>
<th></th>
<th>Data from App 1</th>
<th>Data from App 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP</td>
<td>UDP</td>
<td>IMP</td>
<td>Data from App 1</td>
<td>Data from App 2</td>
</tr>
<tr>
<td>20</td>
<td>8</td>
<td>20</td>
<td>100</td>
<td>150</td>
</tr>
</tbody>
</table>
```

Mobility XE increases efficiency by combining application data.

**Reduced and Synchronized Retransmissions**

Much of the poor performance over WWAN links can be attributed to retransmission policies that are over-aggressive. Most IP implementations have been tuned to work in high-bandwidth, low-loss environments (like wired infrastructures). But wireless wide area connections experience variable latency, which can easily be misconstrued by normal implementations as packet loss. The transport protocol then incorrectly retransmits a copy of a previous frame.
This retransmission affects performance in two ways:

- Duplicate data is forwarded over an already bandwidth-challenged link. This unnecessary transmission consumes the precious resources of a limited pipe and delays transmission of new data and acknowledgments for transmitted data.

- When a transport retransmits a frame, it normally assumes that network congestion has caused the loss and employs a back-off algorithm, reducing the total amount of data that can be transmitted at any one time.

By increasing the actual retransmission time when a frame is really lost, the recovery time may be significantly increased, thus reducing overall performance for no good reason. This can result in a spiraling effect, further reducing the transmission of data to an almost a stop-and-wait exchange.

Mobility XE uses many heuristics beyond the standard round-trip calculation to determine network latency. These extra calculations allow Mobility to adjust more rapidly to varying latency and the other conditions that WWANs exhibit. Mobility significantly reduces the amount of duplicate data that is sent, allowing more application data to fit into the communications pipe.

A simple but effective example of these heuristics is Mobility XE’s synchronized retransmission policy. When a frame is transmitted in current implementations, a timeout is set for when an acknowledgment for the frame should be received. If the acknowledgment has not been received, some transports blindly submit another copy of the frame. But when Mobility decides to retransmit a frame, it checks to see if the underlying network layer has processed the previous copy of the frame. If not, it delays submitting another copy. If a mobile device is temporarily out of range, for example, there is no need to send another copy if the previous one has not yet left the local system.

Another example of MobilityXE’s advanced features is its ability to share information about the link characteristics across all application-level connections.

Here’s how current transport implementations operate:

- As each application creates a connection to a peer, the transport mechanism determines the characteristics of the link.

- Once this information is gathered, performance increases. But it takes time for enough application data (possibly upwards of 64k) to be passed through the link.

- When the connection is terminated, all the learned information is lost.

In contrast, Mobility XE does the following:

- Mobility XE retains the learned information; if an application starts another connection, it starts out using the tuned parameter settings.

- When Mobility XE roams from one network interconnect to another, the new characteristics are applied to all active connections at once. Each connection does not need to go through the exercise of recalculating these values, possibly causing more retransmissions.
Fragmentation Optimizations

The fragmentation of IP packets is regarded by the networking community as a necessary evil that should be avoided. It uses resources in a number of ways, including:

- An intervening system (for example, a router) may have to do further processing on fragmented frames instead of just forwarding them to their ultimate destination.

- It can consume significant resources on the receiving system to reassemble a frame.

- If any part of the fragmented frame is lost, the entire frame must be retransmitted again.

But when roaming from one network interconnect to another (with a possible change in the maximum frame size), fragmentation might be necessary. Here’s how Mobility XE optimizes it:

There is a maximum message size that can be transferred across a link, and Mobility XE periodically probes the network to discover what that size is. If the application submits a request to send data that is too large to fit in a single message, it fragments it before passing it to the underlying network layer. The benefit to this is that the data traverses the network as “normal” (unfragmented) frames and does not cause any extra overhead on intermediary systems.

The Mobility XE message fragmentation algorithm has been optimized to ensure that a minimal amount of resources, both computational and memory-related, are consumed to both fragment and reassemble the message. In the event of a retransmission, the fragmentation is reassessed. If the maximum message size actually grew, the frame may be retransmitted in its entirety, again conserving network overhead.

Data Compression

Data compression can improve throughput over “bandwidth-challenged” connections or in congested network environments, saving customers a significant amount of time and money (depending on the network billing policies). With Mobility XE, the system administrator can customize compression functionality and determine when it should be turned on, and whether it should be enabled globally (for all users and devices), for a mobile device class, a specific device, or an individual user. Or Mobility can be configured to automatically switch compression on or off based on the current interface speed. Users can roam between high-bandwidth 802.11 WLANs and lower-bandwidth GPRS or 1xEV-DO WANs and automatically get the best performance possible.

Mobility XE compresses data that is transmitted between the Mobility server and client. It employs the standard algorithms outlined in RFC 1951 (LZ77 Deflate/Inflate). Only the application payload of each frame is compressed — the transport headers are not modified. This allows Mobility to operate through any policy enforcement equipment, such as firewalls and network address translators (NATs). It also operates over any IP-based network.

Effect of data compression on file transfer time

(341k text file, WLAN link)
As outlined in RFC 1951: “A simple counting argument shows that no lossless compression algorithm can compress every possible input data set. English text usually compresses by a factor of 2.5 to 3; executable files usually compress somewhat less; graphical data such as raster images may compress much more.”

In other words, mileage may vary. Mobility XE also abides by the “no expansion policy” as defined in RFC 1951: transmitted data will not increase in size.

Since the compression process is computationally intensive, a trade-off must be made to provide the maximum benefit to the user. As a rule of thumb, with end-to-end data rates above one megabit per second, compression may not yield any significant savings. Mobility XE takes this into account and uses other advanced algorithms that detect network latency and compression ratios. Based on the network latency and the percentage of savings gained by transmitting the compressed frame instead of the original, Mobility may elect to transmit a frame in its uncompressed format. This reduces the CPU cycles required to decompress the frame upon reception and provides the maximum benefit to the user. As always, Mobility consumes network and computational resources as efficiently as possible.

Web Acceleration

To speed up web browsing on slow networks, Mobility XE gives the option of compressing images. The level of compression is configurable, and co-exists with Mobility XE’s mobile VPN security (in contrast, the web acceleration solutions provided by most wireless carriers cannot be used in conjunction with a VPN):

- JPEG images are compressed using the JPEG quality metric (the fastest setting results in a file size that is about 28 percent of the original size).
- The number of bits per pixel in GIF images is reduced, which reduces the number of colors. The image is also “flattened”: animated GIFs are reduced to one image and textual comments are removed.

Web acceleration is available in two different places:

- Policy Management module: Using policies, the administrator can selectively turn web acceleration off and on, change the level of compression, or change the HTTP ports, based on the current network characteristics, on a specific application, or any of the other available conditions.

- Client Settings: Web acceleration is available in the core product without additional licensing. The level of compression can be configured; when on, all HTTP traffic on the designated ports will have the images compressed at the configured level.
Packet Sizes

TCP/IP’s packet sizes are not always optimum for wireless transmission. In a wireless environment, error rates rise as transmission power drops, making errors much more likely in marginal coverage areas. Sending large packets in these situations increases the probability that an entire packet has to be thrown away and re-sent. Smaller packets may increase overall efficiency by decreasing the number of re-sends. Many network administrators are unaware that even WLAN’s can have substantial numbers of dropped packets and packet errors.

The UDP protocol is much more appropriate for use over wireless networks. It avoids the overhead and inefficiencies of TCP, which was not designed with wireless networks in mind. Instead, Mobility’s Internet Mobility Protocol (IMP) rides on top of UDP, and implements its own methods for dynamically adjusting both packet sizes and timing parameters for the network conditions. IMP, used in conjunction with UDP, handles the far greater variety of transmission speeds and connection conditions encountered in wireless networking. Using this approach also allows IMP to apply its own compression and link optimizations, which can significantly improve the throughput over bandwidth-constrained networks.

Reliability

Because the UDP protocol is connectionless, Mobility’s Internet Mobility Protocol handles the job of guaranteeing reliable data delivery. It uses its own algorithms for selectively acknowledging packets, handling timeouts, detecting dropped packets and retransmitting them. It is far more sophisticated than TCP/IP — and indeed, it has to be. It not only has to verify delivery amid the uncertainties of a wireless environment, but also has to do so with minimal overhead and with as few retransmitted frames as possible without over-consuming limited bandwidth.

Traffic Shaping

Quality of Service

Mobility XE implements sophisticated Quality of Service (QoS) using the Policy Management module to set the parameters that prioritize and shape network traffic. QoS is crucial to maintaining productivity as workers move from high-speed, high-bandwidth networks, to lower capacity, higher latency networks. For example, while connected to the LAN via Ethernet, performance may be just fine for the mission-critical enterprise application or a voice-over-IP call (VoIP), running alongside e-mail, web browsing, and other applications. But on a WWAN, administrators want to prioritize use of the narrower bandwidth, and make sure that a web browser and e-mail client do not use capacity needed by the enterprise or VoIP application.

QoS plays a vital role as voice and video traffic demand a greater share of network resources. These are high-bandwidth uses that depend on timely, reliable packet delivery. Using QoS policies, administrators are able to give these applications the priority they need to function properly.
Mobility XE pre-defines five broad classifications for traffic priority: High Priority, Voice, Video, Best Effort and Background. Each classification has a preset configuration for the various QoS-related settings, which provide fine-grained control over traffic shaping, packet queuing behavior, timing, and other related mechanisms. For instance, critical VoIP traffic can be assigned the Voice priority for optimum traffic-shaping, or else assigned Best-Effort or Background priority if it is not an essential use of the network.

Mobility XE allows for different settings based on specific applications or IP addresses. This degree of control over network use is the hallmark of the Mobility XE Mobile VPN. Conventional VPNs may only allow administrators to shut off non-essential applications. In contrast, with Mobility XE, applications can continue to run but at a throttled-back (e.g. “trickle”) level. And when the application(s) with highest priority finish transmitting data, full access to the network tunnel is automatically restored to the remaining applications.

If enabled, the QoS rules affect all traffic that passes through the Mobility XE tunnel.

**Packet-Loss Recovery (PLR)**

Packet-Loss Recovery is part of the Mobility XE QoS mechanism, and is specifically beneficial for media streams such as voice or video traffic. These rely on continuous, time-sensitive, sequential packet delivery. Since retransmitting a packet takes a relatively long time, especially on lower-bandwidth wireless networks with higher latency and jitter, lost or dropped packets often result in momentary picture loss or a break in the conversation. The PLR technique applied by Mobility XE QoS policies uses a sophisticated mathematical model that adds a small amount of overhead to each packet. When packets are lost, PLR reconstructs them using information from the packets that were received without retransmitting. The Packet Loss Recovery Level Settings adjustment allows administrators to balance the need for recovery against the network conditions and the amount of additional payload added to each packet. A low PLR setting adds less data, and is generally sufficient in situations with only minor packet loss. A high PLR setting makes recovery more effective, but also increases the amount of payload in each packet. PLR is enabled through the QoS Policy Management settings, and applied by default for traffic classified as Voice or Video. The default Packet Loss Recovery setting is medium.

**Management**

**Policy Management**

The Mobility server maintains user policies and pushes them out to the Mobility clients in the field where they are enforced. Policy documents are stored in the Mobility warehouse and are shared by all servers in a pool.
Mobility policies are pushed to the mobile device when it connects. These policies can be defined to be specific to the device or the user. In addition, separate policies may be defined for enforcement when a device connects in unattended mode; this is a recommended practice for allowing only designated device-management applications to run unattended. Policy updates or modifications are applied in real-time after the administrator publishes them for distribution.

For storing, retrieving, and validating rules between client and server, Mobility uses an XML-based rules language. A policy document contains an XML definition of the policy, as an ordered list of rules referenced from the rules documents (a rule or set of rules that control client network behavior). The Mobility server parses the verbose server-side XML into usable code objects, and validates rules against an XSD (XML schema definition) when saving or opening the generated document. When rules are created and saved, it performs an XSL transformation from verbose server-side XML to the client-side format, creating a resource-efficient policy document containing only the information needed by each client.

The policies allow extremely flexible and fine-grained control over user and device access to network resources. While the rules are enforced at the device level, the human-readable rule sets are maintained at the Mobility server.

### Policy Enforcement

The basic Policy Management actions that have an impact on connections are: allow, block, disconnect, passthrough, and bypass traffic. Mobility XE enforces these actions at the client. When an application attempts to send over the network, the Mobility client checks the policy list for the application, port, destination address, battery status, NAC status and other parameters to see if action should be taken. The descriptions, below, assume that actions described are the base action for the policy. Behavior may change if the action is invoked after a session has been activated.

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allow</td>
<td>The Mobility client passes the request to the Mobility server and on to the destination.</td>
</tr>
<tr>
<td>Block</td>
<td>The Mobility client sets the I/O status of the connection request at the TDI layer to indicate the connection is pending. The Winsock interface sends a corresponding response to the application that is consistent with the I/O method used. To the application, it appears that the session is still active.</td>
</tr>
<tr>
<td>Disconnect</td>
<td>The Mobility client passes an indication to the Winsock interface that the server on the other end requested a disconnect. This makes it appear that the server on the other end terminated the connection, and the session is torn down.</td>
</tr>
<tr>
<td>Passthrough</td>
<td>The Mobility client passes the traffic directly onto the TCP/IP transport, so it passed on as normal TCP/IP traffic, outside the encrypted tunnel between the Mobility client and server.</td>
</tr>
<tr>
<td>Bypass Traffic</td>
<td>The traffic bypasses the Mobility XE virtual network adapter as well the encrypted tunnel.</td>
</tr>
<tr>
<td>Set QoS Parameters</td>
<td>The Mobility client prioritizes the traffic as it passes through the VPN tunnel, using the specified settings.</td>
</tr>
</tbody>
</table>
In addition to control over the traffic flow, Mobility XE supports other actions such as the ability to launch applications, execute command-line statements, set various parameters, display information to the user, and others. Policy Management makes granular management of wireless bandwidth, security, and mobile productivity straightforward and achievable — even over networks the administrator neither owns nor controls.

**Network Access Control (NAC)**

The Mobile Network Access Control (NAC) module detects that security measures are enabled and up-to-date and that the device is configured in accordance with defined policies. Based on the device’s lack of compliance with the NAC module policies, it can take a variety of actions, from simple warnings, to enforcing remediation, to disconnecting or quarantining the device.

NetMotion Mobile NAC differs from conventional NAC in its ability to maintain worker productivity by customizing the response to a non-compliant client. A worker in the field should not always be forced to interrupt their workday because of a minor security concern. Operating system updates and antivirus signature downloads that can take many minutes over a cellular network can be delayed until the end of the working day, and/or put off until the worker is within reach of a higher-speed link.

Administrators use the NAC module to create policies that check the security posture of a client device.

<table>
<thead>
<tr>
<th>Category</th>
<th>Parameters Checked</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antivirus and Antispyware</td>
<td>Specified product installed, real-time protection enabled, signatures up-to-date, date and result of the last scan</td>
</tr>
<tr>
<td>File</td>
<td>Specified file either present or not present on the client</td>
</tr>
<tr>
<td>Firewall</td>
<td>Specified product installed and running</td>
</tr>
<tr>
<td>Process</td>
<td>Specified application or service running or not running</td>
</tr>
<tr>
<td>Registry Key</td>
<td>Keys in the HKEY_LOCAL_MACHINE\section of the registry present or not present, and have the expected values</td>
</tr>
<tr>
<td>Windows Update</td>
<td>Auto-update enabled, and/or specific patches present</td>
</tr>
<tr>
<td>Mobility Version</td>
<td>Version of the Mobility client</td>
</tr>
<tr>
<td>Operating System</td>
<td>OS version, service pack, processor and other platform information</td>
</tr>
</tbody>
</table>

The NAC module functions in much the same way as the Policy Module. The Mobility Server stores the NAC rules and pushes them out to each device when it connects. The Mobility client on each device checks for NAC compliance at connection time and periodically at intervals (default is five minutes).
NAC Enforcement

If a client device fails a NAC policy check, it can be assigned one of four states.

<table>
<thead>
<tr>
<th>Status</th>
<th>Description/Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warn</td>
<td>The client does not comply with one or more checks in a rule. The Mobility client device is allowed to connect, but the Mobility client displays a warning.</td>
</tr>
<tr>
<td>Remediate</td>
<td>The client does not comply with one or more checks in a rule that requires remediation. The action required to bring the device into compliance is determined by Policy Management rules that apply to the remediation level.</td>
</tr>
<tr>
<td>Disconnect</td>
<td>The client does not comply with one or more checks in a rule that causes the client to be disconnected.</td>
</tr>
<tr>
<td>Quarantine</td>
<td>The client does not comply with one or more checks in a rule that causes the device to be quarantined. The system administrator must clear the quarantine before the device can connect.</td>
</tr>
</tbody>
</table>

When the status is Remediate and the NAC and Policy Modules both have an active license present on the Mobility server, the administrator has a number of flexible options, including taking specific action based on the speed of the current connection, time of day, etc. A typical example might be: If anti-virus signatures are more than seven days old, send a reminder message to update; if more than 14 days old and on a WWAN connection, send a reminder to update as soon as possible; if more than 14 days old and on a Wi-Fi or LAN connection, download new signatures immediately; if more than 21 days old, quarantine the device. The same capability can be used to automatically download and install updates and operating system patches, at a time and in a manner that does not impact worker productivity while protecting the device and the network.

Client Activity/Mobility Console

The Mobility server collects a number of different details from the active device’s session, including:

- Device name
- User name
- Client status
- Authentication mode
- Device description
- Device class & device ID
- Server name
- Virtual address
- POP address
- Bytes sent
- Bytes received
- Client registered (date/time)
- Connection established (date/time)
- Last re-authentication
- Total connect time
- Client version and OS
- Client (docked/undocked)
- Power source (battery/external)
- Battery (power percent)
- Wireless AP SSID
- Wireless AP BSSID
- Apps using the network (bytes sent/received)

Upon initial connection, the Mobility client sends the session details to the Mobility server, which populates the information on the session details page of the Mobility console (for each connected device). Once the state of the connection is recorded on the details page, the information is updated when the information changes or on a periodic basis. For example, the POP address details will only change on a roam event. When the mobile device
acquires a new POP address, the Mobility client will inform the Mobility server of the new session detail — no other details will be sent if they have not changed. Information that requires more frequent updating, such as the state of battery life on a device, is updated based on a configurable timer.

Because the Mobility server proxies the application traffic on behalf of each Mobility client, the bytes transmitted per application (process) can be derived from the Mobility server without having to burden the client or the network with transmitting this information.

**Analytics Module**

The Analytics Module adds reporting and notification capabilities to Mobility deployments. It requires two additional components: a reporting server and a reporting database. The reporting server collects data from all Mobility servers in a pool, and forwards the data to the database for archival and querying. The reporting server also monitors various system conditions and sends notifications when it encounters them. The database collects the data in normalized form and makes it readily available for reporting.

Because the Mobility XE server functions as an application proxy, and each Mobility client manages the wireless connections, the reporting server is able to capture an extraordinary amount of detail about mobile resource usage. This goes far beyond monitoring connections and logon/logoff events. It includes, for instance, the applications used, amount of bytes transferred per application and in aggregate, and the name of the wireless interface (and therefore, the network used.) The data collected delivers insight into individual devices, application and VPN usage; bandwidth consumption and connectivity patterns; even battery life. Because much of this information is already available by virtue of the Mobility server’s application proxy architecture, the Analytics Module adds very little additional overhead to wireless networks. It simply collects, analyzes and makes the data accessible and useful.

As the data arrives, the Analytics Module analyzes it against a set of administrator-defined conditions, and issues notifications when a condition is met. These conditions detect abnormalities in connections, devices, wireless networks and the Mobility deployment itself. Some notifications have administratively configurable thresholds. If a condition that triggered a notification no longer exists, the reporting server sends a follow-up notification — essentially an “all-clear” that the situation is no longer an immediate concern. There is also a report in the Mobility console that shows all of the notifications that have been triggered over time. In addition, administrators can configure whether to be notified automatically via e-mail, through an SNMP manager or Syslog daemon on the network.
The Analytics Module includes a set of pre-defined reports, accessed through the Mobility console, with a straightforward interface for selectively filtering the data. This allows the administrator to group and isolate users, networks, applications, time periods and more.

The componentized design guards the reporting infrastructure against data loss if a single component fails. For instance, if the Analytics Module is disconnected from the network for some reason, the Mobility server queues up the data and forwards it to the Analytics Module when it comes back online.

The NetMotion Mobility XE Analytics Module white paper describes the capabilities of the Analytics Module, gives examples and includes a list of the reports and notifications.

**Related Information**
