MSN Mobile My Services 6.0 Software Porting Kit

Summary

This document describes the overall product delivered by Microsoft MSN Mobile Services and marketed as MSN Mobile My Services 6.0 Software Porting Kit. The product described in this document is scheduled for delivery to the market in Q2 of 2003.

Introduction

This introductory section is intended to give you a preliminary overview of MSN Mobile My Services 6.0 Software Porting Kit (SPK). This section includes a brief explanation of what MSN Mobile My Services is (from a developer's perspective), why you would be interested in using it, and how to get started with your own custom MSN Mobile My Services client implementation. You will learn about the high-level MSN Mobile architecture and the primary options available for connecting client devices to the MSN Mobile service. You will also learn about the services available for implementation on client devices by using the SPK, including instant messaging, e-mail, and alerts. Finally, you will learn about this documentation, including the technologies you will need to know to make use of the SPK and where to go to find additional resources.

MSN Mobile My Services Overview

MSN Mobile My Services 6.0 SPK enables telecommunications applications developers to create applications for mobile client devices with MSN Mobile content. MSN Mobile comprises both a service that resides at a Microsoft data center and client software that resides on the handset and maintains the user's state with the server. The service is called MSN Mobile My Services and the client software applications are called Pocket MSN clients. MSN Mobile My Services provides access to MSP, a standards-based protocol for communications between Pocket MSN client devices and the Microsoft data center. This protocol gives the calling client application schemas for sending and receiving information from MSN Mobile content properties in XML format.

Client application developers can give mobile users real-time or near real-time access to their information and communications tools through a variety of mobile devices. By using a single application interface on a mobile phone or personal digital assistant, users can view, edit and create information in their MSN Mobile accounts. They can access and interact with the same data, whether they are on a PC or a mobile phone, and they can communicate with their contacts regardless of whether the other party is using a PC or a mobile phone. For example, users can:

- Update an address book entry.
- Send an instant message to a buddy.
- Receive an e-mail and forward it to a friend.
- Scan news headlines and other content alerts.

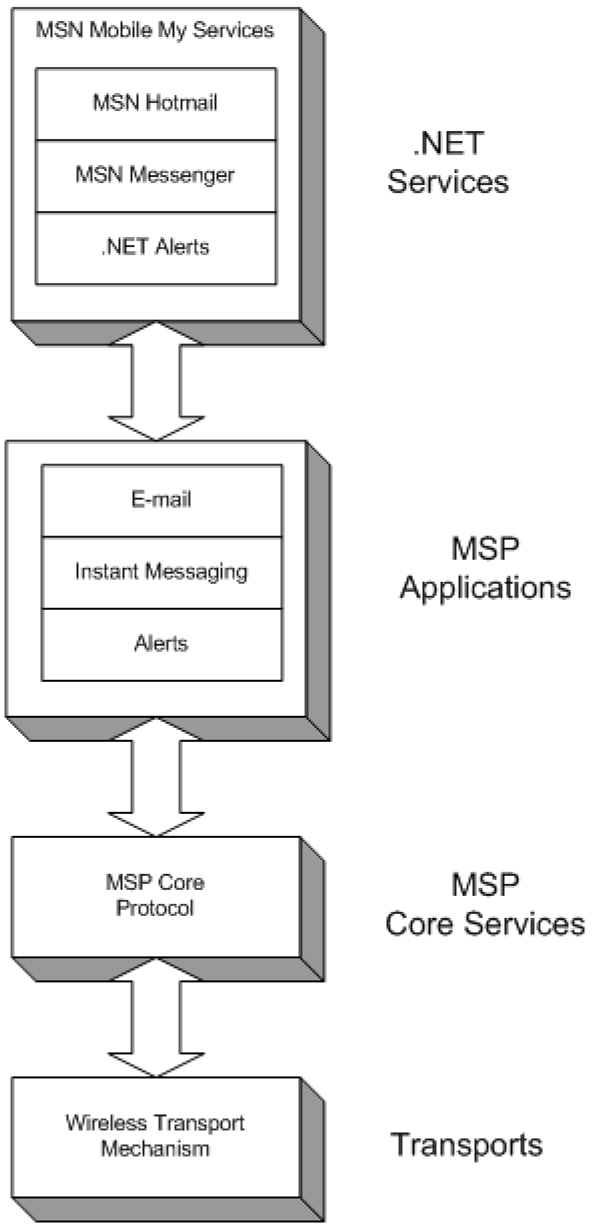

Figure 1. MSN Mobile My Services

About Mobile Services Protocol (MSP)

Mobile Services Protocol (MSP) is a Microsoft proprietary protocol used to support communication between Pocket MSN client devices and MSN Mobile services such as Hotmail. The MSP protocol is based upon commonly used and extensible Internet standards such as Extensible Markup Language (XML). The protocol defines how messages are sent and received between the client and the service. All MSP messages are either requests or responses. Requests typically originate from the client; however it is possible for them to originate from the server. Responses typically originate from the server; however, again it is possible for it to work the other way.

The MSN Mobile service input and output is expressed as XML document fragments. Each of these document fragments must conform to an XML schema document, which is available from the interaction with the MSN Mobile service. However, MSP messages are not sent in actual XML files, but in MSP packets. An MSP packet comprises a binding header followed by one or more compact binary representations of the XML. A compressed data format is used to minimize bandwidth usage on mobile networks. The binary format is based on WBXML (WAP Binary XML Content Format), but with some Microsoft proprietary modifications intended to provide even more efficient network utilization than standard WBXML.

The MSP protocol is designed to be essentially indifferent to the underlying transport method used for communicating the MSP messages. However, initially the protocol supports three transport methods:

- **UDP (User Datagram Protocol).** UDP is supported on port 50000 for both mobile-originated (MO) and mobile-terminated (MT) messages.
- **TCP/IP (Transmission Control Protocol/Internet Protocol).** TCP/IP is supported for both MO and MT messages.
- **SMS (Short Message Service).** SMS is supported for MT messages only.

Note: It is possible for MSP request messages to be sent over one transport method (for example, UDP) while the resultant response message is received over a different method (for example, SMS). When responses are received over a different transport method than the requests, they are said to travel over a "back channel".

For more information about the MSP protocol including supported transport methods and data encryption, see "Section 1. Working with the MSP Core Protocol" on page 16.

Pocket MSN Clients and MSP

Pocket MSN client devices may be developed using a variety of technologies for mobile platforms and developers may communicate with MSP regardless of the platform they're targeting. Clients must support one or more of the transport methods (UDP, TCP/IP, etc.) used for sending MSP messages.

Additionally, in order to interact with MSN Mobile services, a typical Pocket MSN client should be able to format XML messages and deliver that message to the MSN Mobile data center using SOAP (Simple Object Access Protocol). SOAP can be used to specify exactly how the message headers and XML files are to be encoded and deliver the binary XML payload of an MSP message. While SOAP is often used for sending MSP messages, MSP does not derive from SOAP and is really indifferent to the information exchange protocol.

Clients must authenticate with the MSN Mobile service to establish a session. However, clients do not need to remain connected to maintain the session. Rather, sessions expire after a given length of time. The client must re-authenticate with the service and establish a new session after the session expires.

Client Device Requirements

Pocket MSN clients must be able to manipulate XML document fragments and perform some simple, public-domain cryptographic operations on portions of the message. While typical client devices should be able to work with XML, SOAP, and compressed representations of the XML, the ability to parse XML and use SOAP to deliver messages are not absolute requirements. What is essential is that the client can compress and de-compress the MSP binary data streams and use the XML document fragment payloads they contain. While clients must be able to work with the MSP binary data streams, it is not necessary to write a WBXML-based binary data interpreter from scratch. This SPK contains code samples that you can use to build a WBXML-based interpreter.

Additional minimum requirements for client devices include:

- 3-line black and white screen display.
- Client-side APIs for sending and receiving packet data (via SMS, 1XRTT, GPRS, or 3G). One-way SMS can be used if UDP or TCP/IP support is available.

Pocket MSN Client Features

The following features are available for implementation on Pocket MSN 6.0 client devices:

- **Synchronized Contacts List.** Users can view the same information in their contacts list on the PC or on the mobile phone. The MSN Mobile contacts list is synchronized with the Messenger (either MSN or Windows) contacts list. It can provide a variety of information to the end-users including the display name (friendly name), e-mail/IM addresses, up to three phone numbers, plus the presence (online, mobile, offline) and status (on the phone, busy, and so forth) for each contact. When one of the user's contacts updates their personal information (such as changing a phone number), then the new information can be updated on the user's phone automatically.
- **Instant Messaging.** Users can send and receive instant messages with individuals or groups by using a simple interface that supports basic text messaging and graphical emoticons. They can maintain several instant messaging conversations simultaneously and switch between them. They can hear audible alerts or view pop-up alerts when a new instant message arrives. They can interact with buddies who are online on a PC or on another mobile phone that provides access to MSN Messenger.
- **E-mail.** Users can access their MSN Hotmail Inbox or any other Hotmail folder including user-defined folders. They can compose, reply, forward, delete, and move messages just like they can with Hotmail on the PC. They can also receive notifications on their phone to alert them when they receive new e-mail messages.
- **Alerts.** Users can access MSN Alerts that provide specific content such as news, weather, sports scores, stock quotes, traffic updates, and so forth. Other alerts can be sent when new instant messages or e-mails arrive for users. Users can manage their alerts, including subscribing or unsubscribing to specific alert services, by logging in to a special Web page on the MSN Mobile Web site.

MSN Mobile Network Utilization

The MSN Mobile service is designed with the intention that only absolutely essential data is sent to the mobile phone to maximize network efficiency. For the end user, the MSN Mobile client offers an "always on" service—from sign-in to sign-out, the user is considered "online." Despite providing an "always on" service from the user's perspective, the MSN Mobile service takes steps to provide efficiency of bandwidth utilization for the telecommunications carrier.

These steps fall into three categories:

- **Connection Management.** The service logs users in to specific MSN services such as Hotmail and Messenger so the client only needs to actually connect to the service to transmit data. Clients synchronize with the server upon logging in and are disconnected after remaining idle for a while. To manage synchronization, client application developers must track and manage the "change numbers" provided with communications between the client and service. When back channel carrier implementations are used, the service has the ability to open a data connection via SMS and "wake up" the phone when an event happens that is of sufficient priority.

- **Data Compression.** MSN Mobile increases network efficiency by compressing transmitted data. The compression method used is WAP Binary XML (WBXML), a binary form of XML especially well suited for applications on wireless networks.
- **User Behavior Management.** The service carefully tracks the data that is stored on the phone and only sends down the content that has changed since the last time the user logged in to the service. So if a user logs in to Messenger or Hotmail on the phone, the client can limit downloaded information to just the contact or e-mail that changed since the previous login (if any). Additionally, detailed information about e-mails may be only downloaded only upon a specific request of the user.

For more information about the steps taken by MSN Mobile to achieve efficient network utilization, see the *MSN Mobile Client Product Description* document.

Implementing MSP on Mobile Networks

There are four options available to telecommunications network operators at this time for implementing MSP on their mobile network. These options are:

- Option 1: UDP with SMS back channel
- Option 2: UDP without a back **channel**
- Option 3: TCP/IP with SMS back channel
- Option 4: TCP/IP without a back channel

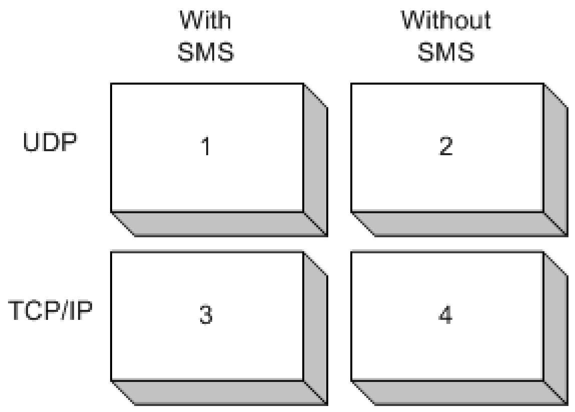

Figure 2. MSP Implementation Options

Note: The first of these options, UDP with an SMS back channel, is the implementation preferred by Microsoft. It provides the best user experience including a connectionless data interchange channel between service and client, fast connection times and low latency. When the client has an IP address that is accessible to the service, the user may have an “always connected” user experience without the network overhead associated with keeping a socket open.

Option 1: UDP with SMS back channel

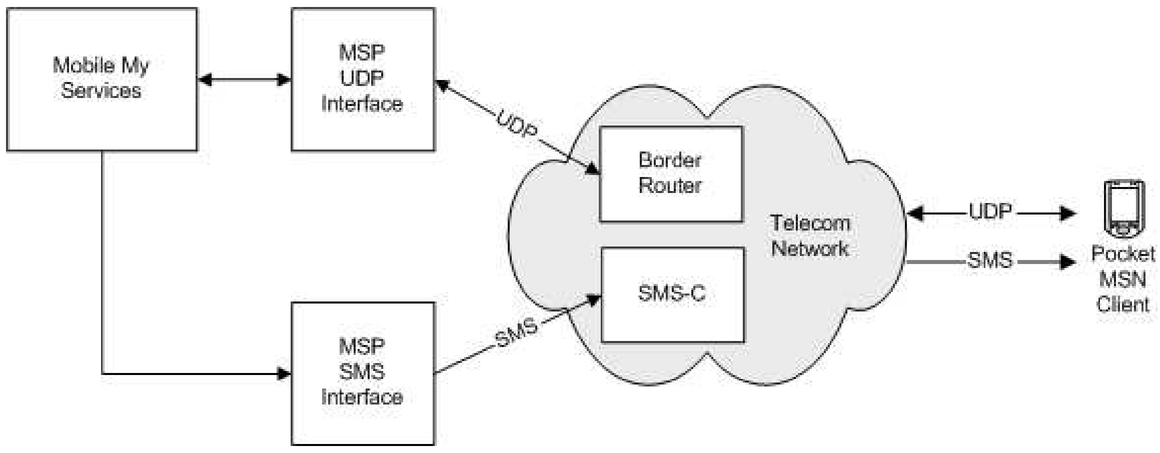

Figure 3. UDP with SMS back channel

MSP over UDP works with either packet switched IP connections or circuit switched IP connections. Examples of supported packet switched connections include General Packet Radio Service (GPRS), single carrier (1x) radio transmission technology (1xRTT), and wireless fidelity (Wi-Fi). Supported circuit switched IP connections include Code-Division Multiple Access (CDMA or IS-95) networks.

In this implementation, the SMS back channel is used to push packets to the Pocket MSN client device when the device does not have an open connection. As a result, users will receive updates about their MSN Mobile services regardless of whether or not their device is not connected to the network. Compared to implementation options without a back channel, users will therefore receive data in a more timely fashion. Also, users may experience shorter wait times when connecting to the service compared to the implementation of UDP without a back channel, because there may be fewer updates to download each time they connect to the service.

An MSP packet contains a time-to-disconnect (TTD) that can be used by the UDP interface as a “clue” from the client that suggests whether or not the current IP address is valid. Any MSP responses or new MSP requests are sent to the client only if the TTD has not passed. The TTD is renewed when a new MSP packet is received from the client. If the TTD has passed, then the current IP address for the device is invalidated and any packets intended for the client that are received by the MSN Mobile service will not be sent. Instead, packets intended for the client will either be queued for sending when the client reestablishes an IP connection or they will be sent to the client over SMS.

The service uses SMS binary and SMS segmentation and reassembly when they are supported by the telecommunications service provider. If SMS binary is not available, then packets are base64 encoded. If the packet exceeds the telecom operator’s specified maximum SMS message size, then the packet is queued for sending when the client reestablishes an IP connection. When this occurs, an MSP ping request is sent to the client instead of the queued MSP packet. The ping request includes information about the queued MSP packet that the client may use to determine whether or not to reestablish an IP connection.

Note: Currently the MSN Mobile service does not have a way to determine whether or not the client is unreachable using UDP, so the service treats the client’s IP address as valid for the full amount of the TTD period.

Option 2: UDP without a back channel

MSP over UDP without an SMS back channel is also supported. As in the previous option, both packet switched IP connections and circuit switched IP connections are supported by the MSN Mobile service.

The only difference between this option and having a back channel is that the service does not send packets if the Pocket MSN client device does not have an open connection to the service. As a result, users will not receive updates about their MSN Mobile services whenever their device is not connected to the network. Instead, the service stores the packets intended for the user's device until the next time the user logs in to the service. Additionally, users could experience longer wait times when connecting to the service compared to the implementation of UDP with a back channel, because there may be a greater quantity of updates to download each time they connect to the service.

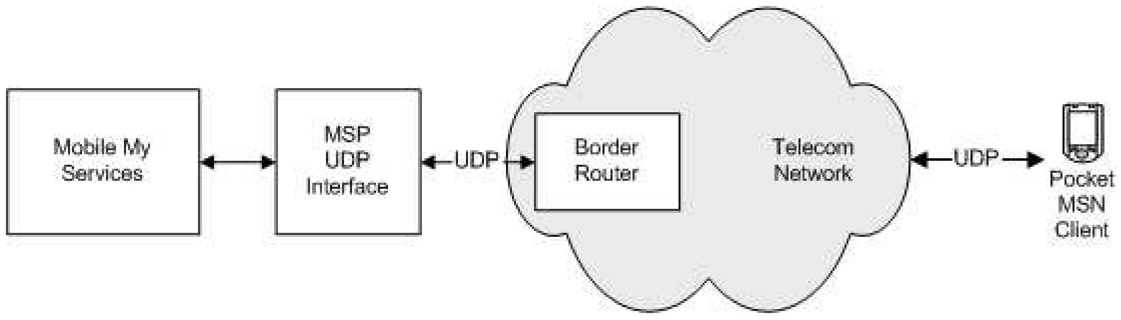

Figure 2. UDP without back channel

Option 3: TCP/IP with SMS back channel

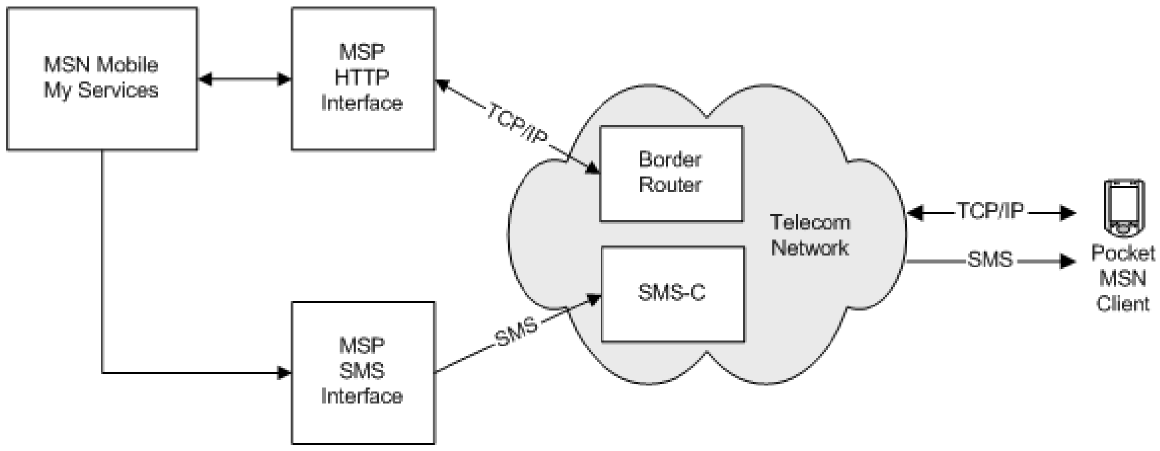

Figure 4. TCP/IP with SMS back channel

MSP over TCP/IP works with either packet switched (for example, GPRS, 1xRTT and Wi-Fi) or circuit switched (for example, CDMA/IS-95) IP connections. The SMS back channel is used to push packets between Pocket MSN client polls to the TCP/IP interface.

The Pocket MSN client polls the TCP/IP interface to which the service responds by sending all the pending MSP packets. However, if the poll contains any MSP requests, then the TCP/IP interface waits for the response before sending all the pending MSP packets. Between polls, the service either queues incoming packets intended for the client or sends the packets to the client over the SMS back channel.

The service uses SMS binary and SMS segmentation and reassembly when it is supported. If SMS binary is not available, then the packet is base64 encoded. If the packet exceeds the telecom operator's specified maximum SMS message size, then the packet is queued for sending when the client reestablishes an IP connection. At that time, the service sends an MSP ping request to the client in place of the queued MSP packet. The ping request includes information about the queued MSP packet which the client uses to determine whether or not to poll the TCP/IP interface.

Option 4: TCP/IP without a back channel

It is also possible to implement the MSP service over TCP/IP without a back channel. This approach works in the same fashion as with a back channel (see "Option 3: TCP/IP with SMS back channel", above), except that the service only sends packets to the Pocket MSN client when it is polled via the TCP/IP interface.

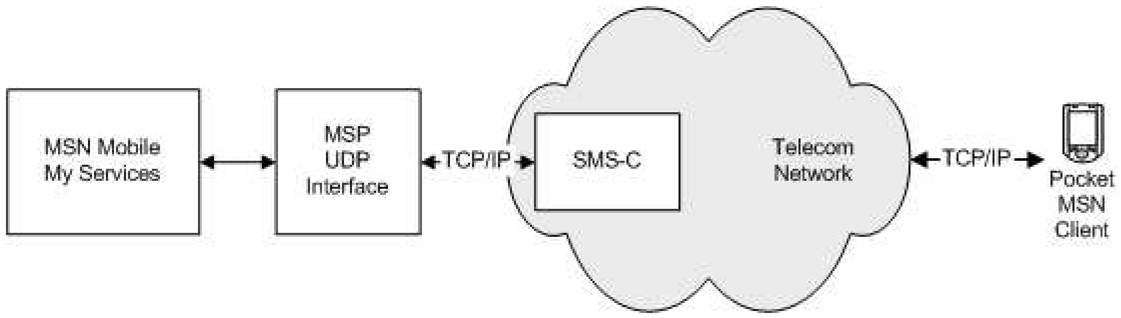

Figure 4. HTTP without back channel

About the Documentation

This section provides information on how the Mobile My Services SPK is organized and about additional documentation resources available.

How This Guide is Organized

This guide is organized in a way that presents information on MSP and its Web services as individual sections. Additional information, including a glossary and troubleshooting assistance, can be found in the appendices.

This guide contains the following sections:

- **Section 1. Working with the MSP Core Protocol**
- **Section 2. Authenticating Users**
- **Section 3. Working with Contacts**
- **Section 4. Working with E-mail**
- **Section 5. Working with Messaging**
- **Section 6. Working with Profiles**
- **Section 7. Working with Alerts**

Additionally, there are appendices that provide a reference to the MSP compression tables and notes about the sample applications provided with the porting kit.

Who Should Use This Guide

This guide will be useful primarily to persons responsible for planning, developing and implementing Pocket MSN client applications on mobile devices.

Developers using this guide must be familiar with the programming languages and technologies used by their target application platform, plus XML and WBXML technologies.

Additional Documentation Resources

In addition to the Mobile My Services SPK documentation, there are additional documentation resources available to assist with creating Pocket MSN client applications. For general information about Microsoft development processes, standards and technology, see the MSDN home page at http://msdn.microsoft.com/default.asp.

The following Microsoft documentation may be of particular benefit to you:

- **MSN Mobile Client 6.0 Product Description.** This resource provides a sample reference client implementation with user interface screen samples and process flows.
- **Mobile Services Protocol Version 2.0.** Product development specifications with details on the core services protocol.
- **Mobile Services Protocol Version 2.0 Extensions for Instant Messaging.** Product development specifications with details on IM protocol extensions.
- **Mobile Services Protocol Version 2.0 Extensions for E-mail.** Product development specifications with details on e-mail protocol extensions.

Additional Resources

Additionally, you can find more information about some of the technologies related to MSP:

- **Wireless Application Protocol Wireless Session Protocol Specification** (http://www1.wapforum.org/tech/documents/WAP-203-WSP-20000504-a.pdf). Defines an industry-wide specification for developing applications that operate over wireless communication networks. This document provides background information related to WSP that some developers may find useful for understanding core MSP technologies. However, understanding WSP is not a requirement.
- **Binary XML Content Format Specification** (http://www1.wapforum.org/tech/documents/WAP-192-WBXML-20010725-a.pdf). Defines a compact binary representation of XML. The binary designed to allow more effective use of XML data on narrowband communication channels. WBXML, with some proprietary modifications, is the basis for the compression used in Mobile My Services.
- **Simple Object Access Protocol (SOAP) 1.1 Specification** (http://www.w3.org/TR/SOAP/.) Defines the SOAP protocol for distributed information exchange. May be useful for developers wanting to use SOAP as a container for working with MSP messages.
- **Session Initiation Protocol (SIP)** (http://www.ietf.org/html.charters/sip-charter.html) Defines the SIP proposed standard RFC 2543 for initiating interactive communication sessions between users.
- **XML Path Language (XPath) Version 1.0** (http://www.w3.org/TR/xpath). XPath is a language for addressing parts of an XML document. It is not fully supported in MSP; however, some XPath notations may be used for navigating XML documents as described in this SPK document.

Section 1. Working with the MSP Core Protocol

This section describes the underpinnings of the MSP core protocol: its protocol data unit, bindings, and transport methods. You will also see samples of content contained in the SOAP messages, including requests, responses, headers and the message body. The core methods of the MSP protocol are also described, including inserts, replaces, updates, deletes, notifications and pings. The remainder of the section describes the workings of user sessions and data synchronization. In particular, you will learn about working with change numbers.

This information will be useful in providing a general understanding of the core MSP methods. Client authentication processes and MSDL methods specific to individual MSN Mobile services, such as e-mail or messaging, are described in later sections.

Understanding the Protocol Data Unit and Bindings

This diagram shows the PDU and data binding. The elements of the diagram are discussed in this section.

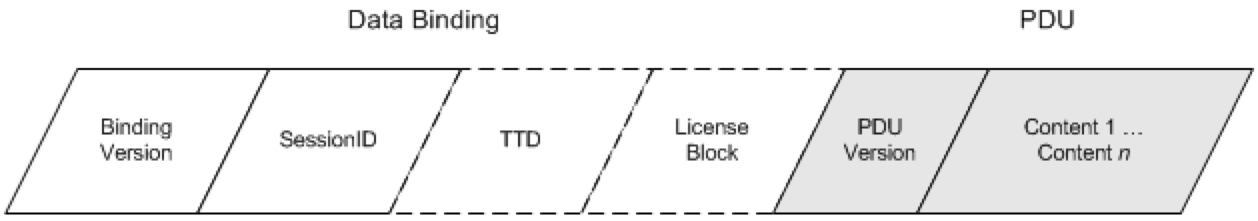

Figure 5. PDU and Data Binding

Element Name	Type	Source
Binding Version	Uint8	The IP binding version.
SessionID		The SessionID parameter.
TTD		The time to disconnect. UDP transport only.
License Block		The license.
BindingEnd	Uint8	The binding end tag (0x00).
PDU Version	Uint8	The MSP version (that is, 2.0).
PDU Content 1 … Content *n*	*n* octets	The SOAP message content.

Binding Version Element

This element contains the data binding version number, both major and minor versions. This implementation of MSN Mobile is IP binding version 2.0. The major version number is stored in the high-order 4 bits. The minor version number is stored in the low-order 4 bits.

Session ID Element

This element contains the Session ID tag. This tag specifies the client or server session number. The server session number is globally unique. Each client session number is unique to the specific client.

Each element of this tag is described in the following diagram and table.

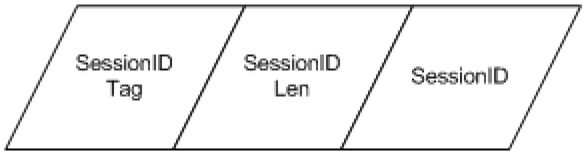

Figure 6. Session ID Element

Name	Type	Source
SessionIDTag	Uint8	The session ID tag (0x01).
SessionIDLen	Uint8	The session ID length.
SessionID	SessionIDLen bytes	The client or server session ID.

TTD Element

This element contains the expected time to disconnect (TTD) for the wireless network connection. It is used for the UDP transport method only. It is an optional element.

Each element of this tag is described in the following diagram and table.

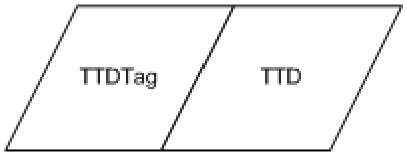

Figure 7. TTD Element

Name	Type	Source
TTDTag	Uint8	The TTD tag (0x02).
TTD	Uint16	The time to disconnect (seconds). This information is used as a "clue" to indicate whether or not a current connection with the client is available.

License Block Element

This element contains the license that ensures the integrity of the message. Both the client and the server should check the license block for each of the following: acceptable packet sequence number and valid keyed hash. The license block is an optional element.

Each element of this tag is described in the following diagram and table.

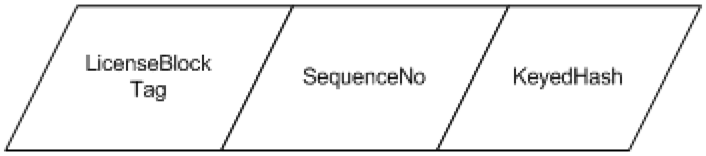

Figure 8. License Block Element

Name	Type	Source
LicenseBlockTag	Uint8	The license block tag (0x03).
SequenceNo	Uint32	The session ID length.
KeyedHash	20 bytes	The client or server session ID.

Acceptable Packet Sequence Numbers

Acceptable packet sequence numbers are greater than the current packet (- 30). If numbers are significantly less than current (- 30) or equal to the current, they are unacceptable. Clients should silently ignore packets with an unacceptable sequence number.

Note that the sequence number check must also account for possibility of packets that may be missing or received out of order.

Keys

Both the client and server derive their keys from the session key.

K_c = Hmac-SHA1(K, ”clientLicenseKey”)
K_s = Hmac-SHA1(K, ”serverLicenseKey”)

Where:

K is the session key.

K_c is the client license key.

K_s is the server license key.

Note: In this example, the hash strings are case sensitive and the length of the license key is 20 bytes.

Keyed Hash

A valid keyed hash is a function of the packet sequence number and PDU using the client or server keys. Clients should silently ignore packets with an invalid keyed hash.

KH_c = Hmac-SHA1(K_c, f(SN_c, PDU))
KH_s = Hmac-SHA1(K_s, f(SN_s, PDU))

Where:

- KH_c is the client keyed hash.
- SN_c is the client sequence number.
- KH_s is the server keyed hash.
- SN_s is the server sequence number.

PDU Version

This element contains the MSP version number, both major and minor versions. This implementation of MSN Mobile is MSP version 2.0. The major version number is stored in the high-order 4 bits. The minor version number is stored in the low-order 4 bits.

PDU Content

The length of the PDU content field is determined from the PDU size as provided by the underlying transport mechanism. A PDU may comprise multiple content blocks (from 1 to *n*). Each content block is delineated by the Envelope element in the SOAP message.

The maximum compressed content length of a single PDU is 16 Kilobytes. For more information, see “MSP Message Content Examples” on page 20.

Note: This element is always transmitted in a compressed state. For more, see

"Understanding Data Compression" on page 41.

About Transport Methods

The following table summarizes key features of two transport methods that may be used with MSP.

	PDU Size	**Port Address**
UDP	The PDU size is determined from the IP Total Length field.	The initial port number is 50000.
SMS	The PDU size and Content field size is determined from the TP-User-Data-Length field(s).	There is currently no application port address for MSP.
TCP/IP		The initial port number is 50000.

User Datagram Protocol (UDP)

MSP supports the UDP as an alternative to Transmission Control Protocol (TCP). UDP offers a limited amount of service when messages are exchanged between computers on an IP network. It provides port numbers to help distinguish different user requests.

When using the UDP transport method, use the Domain Name Service (DNS) to resolve the domain name for mobile services. Note that the domain name for MSN Mobile Services is `services.myservices.mobile.msn.com`.

Short Message Service (SMS)

MSP supports the SMS (Short Message Service) for sending messages of a limited number of characters to mobile phones that use Global System for Mobile (GSM) communication.

Note: A single MSP message may span multiple SMS packets.

Binary SMS Messages

Some mobile devices and network infrastructures do not support binary SMS messages. If binary SMS messages are not supported, then the MSP message should be base 64 encoded.

Run-time Environment Header

Some run-time environments (RTE) support the conditional dispatching of SMS messages to applications. You can use the RTE to route SMS messages in these environments such as Pocket MSN and QUALCOMM's Binary Runtime Environment for Wireless™ (BREW™) platform. MSN Mobile clients may define the run-time environment header by specifying a device capability.

The Pocket MSN RTE header is:

```
//MSN:
```

BREW allows SMS messages to be routed based on their run-time environment header (RTE) header. The BREW RTE is:

```
//BREW:<Class ID>:
```

Note: The BREW RTE is limited to text SMS messages. Class ID is an 8 char hex value (for example, 010ECD01).

MSP Message Content Examples

This section shows MSPmessages containing sample content. Examples are included for requests and responses, message headers and the message body.

```
<s:Envelope xmlns:s=”http://schema.xmlsoap.org/soap/envelope”
    xmlns:srp=”http://schemas.xmlsoap.org/soap/rp/”
    xmlns:ss=”http://schemas.xmlsoap.org/soap/security/”
         xmlns:ms=”http://schemas.microsoft.com/ms/2001/1/core”>
    <s:Header>
        …
    </s:Header>
    <s:Body>
        <ms:method>
            …
        </ms:method>
    </s:Body>
<s:/Envelope>
```

Request Messages

This is a sample request message.

```
<s:Envelope>
    <s:Header>
        <srp:path>
            <srp:to>”http://mycontacts.mobile.msn.com/quentin@msn.com”
                </srp:to>
            <srp:id>202</srp:id>
        </srp:path>
        <ms:request service=”myContacts” method=”insert”
document=”content”
            genResponse=”always”/>
    </s:Header>
    <s:Body>
        <ms:insertRequest>
            …
        </ms:insertRequest>
    </s:Body>
</s:Envelope>
    <s:Header>
        <srp:path>
        </srp:path>
    </s:Header>
</s:Envelope>
```

Response Messages

This is a sample response.

```
<s:Envelope>
    <s:Header>
        <srp:path>
            <srp:from>"http://mycontacts.mobile.msn.com"</srp:from>
            <srp:id>23</srp:id>
            <srp:relatesTo>202</srp:relatesTo>
        </srp:path>
        <m:response service="myContacts" document="content"/>
    </s:Header>
    <s:Body>
        <ms:insertResponse>
            …
        </ms:insertResponse>
    </s:Body>
</s:Envelope>
```

Message Headers

Path

This is a sample message header.

```
<srp:path>
  <srp:to>0..1</srp:to>
  <srp:from>0..1</srp:from>
  <srp:id>1..1</srp:id>
  <srp:relatesTo>0..1<srp:relatesTo>
</srp:path>
```

The meaning of the attributes and elements shown in the preceding sample document fragment are listed in the following table.

Name	Required	Description
srp:to	Required	The to element contains the address of the destination of the message. It typically appears only in MT messages, rarely in MO messages. The absolute URI defaults to user's service entry if the name is not included in the URI. For example, if the user name is "someone@msn.com" then to= "http://mycontacts.mobile.msn.com" represents to= "http://mycontacts.mobile.msn.com/someone@msn.com".
srp:from	Optional	The from element contains the address of the sender of a message. It is in the format of an absolute URI.
srp:id	Required	The id element contains the ID number for the message. It is an unsigned variable length integer. Note that the ID is generated by the sender. It is limited to 16383 (14 bits).
srp:relatesTo	Required	The relatesTo element relates the message to a specific previous request. It is an unsigned variable length integer. Responses must contain this element.

Request Messages

This is a sample request message.

```
<ms:request service="…" method="…" document="…" changeNumber="…"
    genResponse="…">
  <ms:documentRef service="…" document="…"
changeNumber="…"/>0..unbounded
</ms:request>
```

The meaning of the attributes and elements shown in the preceding sample document fragment are listed in the following table.

Name	**Required**	**Description**
ms:service	Required	The service attribute specifies the name of the service to access. Valid values are: • mobileMyServices • alerts • passport • myContacts • myProfile • messaging • myInbox • deviceCapabilities
method	Optional	The method attribute specifies the method to access for the service. The core methods are: • insert • query • delete • replace • update • authenticate • notification • ping
document	Optional	The document attribute specifies the document class to access for the service. Valid document classes are: • content • folder number (myInbox)
changeNumber	Optional	The changeNumber attribute contains the change number that the client has cached for the object. It is an unsigned variable length integer. For more information about change numbers, see "Change Numbers" on page 38.
genResponse		The genResponse attribute controls the generation of a response to the request

		The valid values are: • always – always generate a response. • never – never generate a response. • faultOnly – only generate a response when the request results in a fault message. • faultOrFailure – only generate a response when the request results in a fault or failure response messages.
ms:documentRef	Optional	The documentRef element contains the current change number the client has cached for a related document. It must be included wherever a contact ID is referred to by another service.

Response Messages

This is a sample response.

```
<ms:response service="…" document="…" previousChangeNumber="…"
changeNumber="…"/>
```

The meaning of the attributes and elements shown in the preceding sample response are listed in the following table.

Name	Required	Description
service	Required	The service attribute contains the service to access. The valid values are: • mobileMyServices • alerts • passport • myContacts • myProfile • messaging • myInbox • deviceCapabilities
document	Optional	The document attribute specifies the document class to access for the service. The valid document classes are: • content • folder number (myInbox)
previousChangeNumber	Optional	The previousChangeNumber attribute contains the previous change number the server has stored for the document. This attribute enables chunks to be atomic. It is an unsigned variable length integer. For more information about change numbers, see “Change Numbers” on page 38.
changeNumber	Optional	The changeNumber attribute contains the current change number the server has for the document. It is

		an unsigned variable length integer.

Message Body

The message body contains Mobile Services Data-manipulation Language (MSDL) as described in the following section, “Working with the MSDL Methods”. A sample message is shown below.

```
<s:Body>
    <ms:method>
        …
    </ms:method>
</s:Body>
```

Working with the MSDL Methods

All interactions with the MSN Mobile services are based on simple XML commands. MSDL is the name for the XML elements used while interacting with MSN Mobile services. MSDL is a data manipulation language based on HSDL, the Microsoft .NET My Services data manipulation language that drives interaction with Web services. Like HSDL, MSDL allows for queries, updates, inserts, and other actions as well as allowing their XML payloads to interact with MSN Mobile services.

MSDL is the command interface for the MSN Mobile services. Various core methods are used in the course of interaction with the various services, along with the specific methods designed for common tasks in the particular services (e-mail, instant messaging, and so forth). These methods are used for five common tasks: insert, query, replace, delete and ping. Specific command syntax including supported parameters and common error scenarios for methods related to these tasks are described in this section.

Note: The attribute and element values of MSDL methods are not case sensitive.

Understanding the Data Structure and XPath

MSDL methods are primarily focused on transporting data in and out of XML documents, but their design and syntax also reflect a specific data structure. MSDL has established two man types of nodes in XML documents. The first type is referred to as *blue* and represents the primary data elements of the service. Every immediate child node of any service root element (such as <myProfiles> or <myContacts>) is a *blue,* or primary, element.

In MSDL, XPath is used to target a specific element or portion of a service document to a specific node. In other words, consider the following document fragment:

```
<myContacts>
        <contact id=”some_id”>
```

Both *myContacts* and the contact element are blue types, because they are primary elements in the definition of the MSN Mobile *myContacts* schema. The *id* attribute is defined as a red element within the service. Such an element can be used in an XPath statement as a predicate. For example:

```
/myContacts/contact[id=’some_id’]
```

The above XPath would select a specific element of the myContacts schema that contained a child *id* attribute with the value of *some_id.*You can use XPath predicates against secondary (red) elements and attributes to further tailor your queries.

Inserting Elements

The insert request MSDL method is used to insert a new element into the user's service document.

ms:insertRequest

The *ms:insertRequest* method inserts a valid XML fragment into the selected node of a document. This is a sample request message.

```
<ms:insertRequest select="…"
  xmlns:ms="http://schemas.microsoft.com/ms/2001/1/core">
  {objects}
</ms:insertRequest>
```

The meaning of the attributes and elements shown in the preceding sample document fragment are listed in the following table.

Attribute	**Description**
select	The select attribute is an XPath expression that specifies the node to insert the objects. At this time, it is only possible to specify node by the absolute path to the node (for example, "/m:myWidgets/m:widget") or by the attribute's *id* element (for example, "/m:myWidgets/m:widget[@id='2']").

ms:insertResponse

An insert response message is returned on completion of the *ms:insertRequest* method. The following document fragment illustrates this message.

```
<ms:insertResponse selectedNodeCount="…" status="…"
  xmlns:ms="http://schemas.microsoft.com/ms/2001/1/core">
  <ms:newBlue id="…" changeNumber="…"> </ms:newBlue>
</ms:insertResponse>
```

The meaning of the attributes and elements shown in the preceding sample document fragment are listed in the following table.

Attribute	**Description**
selectedNodeCount	The selectedNodeCount attribute contains the number of selected nodes. It is an unsigned variable length integer.
status	The status attribute indicates the status of the method. See "Return Values" below.
id	The id attribute contains the ID assigned to the xdb:blue object. It is an unsigned variable length integer. It is generated by the MSN Mobile service and is unique within the document.
changeNumber	The changeNumber attribute contains the change number for the new or modified sdb:blue object. It is an unsigned variable length integer. For more information about change numbers, see "Change Numbers" on page 38.

Examples

This is a sample document before the method is invoked.

```
<m:myWidgets changeNumber="1">
</m:myWidgets>
```

This is a sample request message.

```
<ms:insertRequest select="/">
  <m:widget>
      <m:name xml:lang="en">My first widget</m:name>
      <m:unitPrice currency="USD">65.00</m:unitPrice>
  </m:widget>
  <m:widget>
      <m:name xml:lang="en">My second widget</m:name>
      <m:unitPrice currency="USD">25.00</m:unitPrice>
  </m:widget>
</ms:insertRequest>
```

This is a sample response.

```
<ms:insertResponse selectedNodeCount="2" status="success">
  <ms:newBlue id="1" changeNumber="2"/>
  <ms:newBlue id="2" changeNumber="2"/>
</ms:insertRequest>
```

This is a sample document after the method is invoked.

```
<m:myWidgets changeNumber="2">
  <m:widget id="1" changeNumber="2">
      <m:name xml:lang="en">My first widget</m:name>
      <m:unitPrice currency="USD">65.00</m:unitPrice>
  </m:widget>
  <m:widget id="2" changeNumber="2">
      <m:name xml:lang="en">My second widget</m:name>
      <m:unitPrice currency="USD">25.00</m:unitPrice>
  </m:widget>
</m:myWidgets>
```

Return Values

These are the values that may be returned by the service for this method.

Value	Response
status=success	Indicates success. Note that empty selections result in successful responses.
selectedNodeCount=0, status=failure	Indicates a missing primary (blue) element. This only occurs if the client tries to delete an object that doesn't exist any more. In the case of delete, the error can be safely ignored. For changes, the developer may want to provide some sort of error handling code (for example, the user could be notified that the object no longer exists and asked if the user wants to add it).
selectedNodeCount=1, status=failure	Indicates an internal error condition. The operation cannot be performed. In this case, the developer may want to display an error message on the client such as "service temporarily unavailable".

selectedNodeCount=0 or empty set, status=synchronize	Indicates that the change number is out of an acceptable range. In this case, the service cannot complete the method until the client application synchronizes to the service. The client must use an xpQuery to synchronize to the service.
selectedNodeCount=1, status=duplicateBlue	Indicates that the operation cannot be performed because of a duplicate primary (blue) element. The developer may want to provide some sort of duplicate resolution code (for example, the user could be shown both objects and asked which one they want to save). Note that a SQL-based data store with a unique index may generate a duplicateBlue status.

Querying for Nodes

The query request method allows for basic data retrieval functionality. An XPath statement resides in a select attribute telling the MSN Mobile service which XML nodes within the service document you want to query.

ms:queryRequest

The query request method retrieves the properties of objects from a document. Multiple queries can be made in a single *ms:queryRequest* request, if desired.

This is a sample request message.

```
<ms:queryRequest
  xmlns:ms="http://schemas.microsoft.com/ms/2001/1/core">
  <ms:xpQuery select="…"> </ms:xpQuery>
  <ms:changeQuery select="…" baseChangeNumber="…"> </ms:changeQuery>
</ms:queryRequest>
```

The meaning of the attributes and elements shown in the preceding sample document fragment are listed in the following table.

Attribute	**Description**
select	The select attribute is an XPath expression that specifies the node to query the objects. At this time, it is only possible to specify node by the absolute path to the node (for example, "/m:myWidgets/m:widget") or by the attribute's *id* element (for example, "/m:myWidgets/m:widget[@id='2']").
baseChangeNumber	The baseChangeNumber attribute is the value of the current changeNumber that the server has for the selected node. It is an unsigned variable length integer. For more information about change numbers, see "Change Numbers" on page 38.

ms:queryResponse

A query response message is returned on completion of the *ms:queryRequest* method.

```
<ms:queryResponse
  xmlns:ms="http://schemas.microsoft.com/ms/2001/1/core">
  <ms:xpQueryResponse status="…">
      {objects}
  </ms:xpQueryResponse>
  <ms:changeQueryResponse baseChangeNumber="…" status="…">
      {objects}
      <ms:deletedBlue id="…"> </ms:deletedBlue>
  </ms:changeQueryResponse>
</ms:queryResponse>
```

The meaning of the attributes and elements shown in the preceding sample document fragment are listed in the following table.

Attribute	Description
status	The status attribute indicates the status of the method. See “Return Values” below.
baseChangeNumber	The baseChangeNumber attribute is the value of the current changeNumber that the server has for the selected node. It is an unsigned variable length integer. For more information about change numbers, see “Change Numbers” on page 38.
id	The id attribute contains the ID assigned to the xdb:blue object. It is an unsigned variable length integer. It is generated by the MSN Mobile service and is unique within the document.

Examples

This is a sample request message.

```
<ms:queryRequest>
  <ms:xpQuery select="/"/>
</ms:queryRequest>
```

This is a sample response.

```
<ms:queryResponse>
  <ms:xpQueryResponse status="success">
      <m:myWidgets changeNumber="2">
          <m:widget id="1" changeNumber="2">
              <m:name xml:lang="en">My first widget</m:name>
              <m:unitPrice currency="USD">65.00</m:unitPrice>
          </m:widget>
          <m:widget id="2" changeNumber="2">
              <m:name xml:lang="en">My second widget</m:name>
              <m:unitPrice currency="USD">25.00</m:unitPrice>
          </m:widget>
      </m:myWidgets>
  <\ms:xpQueryResponse>
<\ms:queryResponse>
```

Note that a query can use XPath predicates to specify attributes or elements. This is a sample request message that uses XPath notation.

```
<ms:queryRequest>
  <ms:xpQuery select="/m:myWidgets/m:widget[@id='1']"/>
</ms:queryRequest>
```

This is a sample response that uses XPath notation.

```
<ms:queryResponse>
  <ms:xpQueryResponse status="success">
      <m:myWidgets changeNumber="2">
          <m:widget id="1" changeNumber="2">
              <m:name xml:lang="en">My first widget</m:name>
              <m:unitPrice currency="USD">65.00</m:unitPrice>
          </m:widget>
      </m:myWidgets>
  <\ms:xpQueryResponse>
<\ms:queryResponse>
```

This is a sample request message.

```
<ms:queryRequest>
  <ms:xpQuery select="="/m:myWidgets /m:widget[@id='3']"/>
</ms:queryRequest>
```

This is a sample response.

```
<ms:queryResponse>
  <ms:xpQueryResponse status="success"/>
<\ms:queryResponse>
```

Return Values

These are the values that may be returned by the service for this method.

Value	Response
status=success	Indicates success. Note that empty selections result in successful responses.
selectedNodeCount=0, status=failure	Indicates a missing primary (blue) element. This will only occur if the client tries to delete an object that doesn't exist any more. In the case of delete, the error can be safely ignored. For changes, the developer may want to provide some sort of error handling code (for example, the user could be notified that the object no longer exists and asked if the user wants to add it).
selectedNodeCount=1, status=failure	Indicates an internal error condition. The operation cannot be performed. In this case, the developer may want to display an error message on the client such as "service temporarily unavailable".

Deleting Nodes

The delete request method is a simple element with a select attribute indicating with XPath the targeted element to delete from the service document.

ms:deleteRequest

The *ms:deleteRequest* method deletes the selected note and all child nodes of a document.

```
<ms:deleteRequest select=”…”
  xmlns:ms=”http://schemas.microsoft.com/ms/2001/1/core”>
</ms:deleteRequest>
```

The meaning of the attributes and elements shown in the preceding sample document fragment are listed in the following table.

Attribute	Description
select	The select attribute is an XPath expression that specifies the node to delete the objects. At this time, it is only possible to specify node by the absolute path to the node (for example, “/m:myWidgets/m:widget”) or by the attribute’s *id* element (for example, “/m:myWidgets/m:widget[@id=’2’]”).

ms:deleteResponse

A delete response message is returned on completion of the *ms:deleteRequest* method.

```
<ms:deleteResponse selectedNodeCount=”…” status=”…”
  xmlns:ms=”http://schemas.microsoft.com/ms/2001/1/code”>
  <ms:changedBlue id=”…” changeNumber=”…”> </ms:changedBlue>
  <ms:deletedBlue id=”…” changeNumber=”…”> </ms:deletedBlue>
</ms:deleteResponse>
```

The meaning of the attributes and elements shown in the preceding sample document fragment are listed in the following table.

Attribute	Description
selectedNodeCount	The selectedNodeCount attribute contains the number of selected nodes. It is an unsigned variable length integer.
status	The status attribute indicates the status of the method. See “Return Values” below.
id	The id attribute contains the ID assigned to the xdb:blue object. It is an unsigned variable length integer. It is generated by the MSN Mobile service and is unique within the document.
changeNumber	The changeNumber attribute contains the change number for the new or modified sdb:blue object. It is an unsigned variable length integer. For more information about change numbers, see “Change Numbers” on page 38.

Examples

This is a sample request message.

```
<ms:deleteRequest select="/myWidgets/m:widget[m:unitPrice =
'65.00']"/>
```

This is a sample response.

```
<ms:deleteResponse selectedNodeCount="1" status="success">
  <hs:deletedBlue id="1" changeNumber="3"/>
</ms:deleteResponse>
```

This is a sample document after the method is invoked.

```
<m:myWidgets changeNumber="3">
  <m:widget id="2" changeNumber="2">
      <m:name xml:lang="en">My second widget</m:name>
      <m:unitPrice currency="USD">25.00</m:unitPrice>
  </m:widget>
</m:myWidgets>
```

Return Values

These are the values that may be returned by the service for this method.

Value	Response
status=success	Indicates success. Note that empty selections result in successful responses.
selectedNodeCount=0, status=failure	Indicates a missing primary (blue) element. This will only occur if the client tries to delete an object that doesn't exist any more. In the case of delete, the error can be safetly ignored. For changes, the developer may want to provide some sort of error handling code (for example, the user could be notified that the object no longer exists and asked if the user wants to add it).
selectedNodeCount=1, status=failure	Indicates an internal error condition. The operation cannot be performed. In this case, the developer may want to display an error message on the client such as "service temporarily unavailable".
selectedNodeCount=0 or empty set, status=synchronize	Indicates that the change number is out of an acceptable range. In this case, the service cannot complete the method until the client application synchronizes to the service. The client must use an xpQuery to synchronize to the service.

Replacing Elements

The replace message method allows the client to target a specific primary (blue) element within the service document and replace its contents with the contents supplied in the current replace request. The targeted element is replaced in its entirety.

ms:replaceRequest

The *ms:replaceRequest* method replaces the object at the selected node in the document with a new object.

```
<ms:replaceRequest select="…"
  xmlns:ms="http://schemas.microsoft.com/ms/2001/1/core">
  {objects}
</ms:replaceRequest>
```

The meaning of the attributes and elements shown in the preceding sample document fragment are listed in the following table.

Attribute	Description
select	The select attribute is an XPath expression that specifies the node to insert the objects. At this time, it is only possible to specify node by the absolute path to the node (for example, “/m:myWidgets/m:widget”) or by the attribute’s *id* element (for example, “/m:myWidgets/m:widget[@id=’2’]”).

ms:replaceResponse

A *ms:replaceResponse* message is returned on completion of a replace request method.

```
<ms:replaceResponse selectedNodeCount=”…” status=”…”
  xmlns:ms=”http://schemas.microsoft.com/ms/2001/1/core”>
  <ms:newBlue id=”…” changeNumber=”…”> </ms:newBlue>
  <ms:changedBlue id=”…” changeNumber=”…”> </ms:changedBlue>
  <ms:deletedBlue id=”…” changeNumber=”…”> </ms:deletedBlue>
</ms:replaceResponse>
```

The meaning of the attributes and elements shown in the preceding sample document fragment are listed in the following table.

Attribute	Description
selectedNodeCount	The selectedNodeCount attribute contains the number of selected nodes. It is an unsigned variable length integer.
status	The status attribute indicates the status of the method. See “Return Values” below.
id	The id attribute contains the ID assigned to the xdb:blue object. It is an unsigned variable length integer. It is generated by the MSN Mobile service and is unique within the document.
changeNumber	The changeNumber attribute contains the change number for the new or modified sdb:blue object. It is an unsigned variable length integer. For more information about change numbers, see “Change Numbers” on page 38.

Examples

This is a sample request message.

```
<ms:replaceRequest select="/m:myWidgets/m:widget[@id='2']">
  <m:widget>
      <m:name xml:lang="en">My new second object</m:name>
      <m:unitPrice currency="GBP">8.99</m:unitPrice>
  </m:widget>
</ms:replaceRequest>
```

This is a sample response.

```
<ms:replaceResponse selectedNodeCount="1" status="success">
  <ms:changedBlue id="2" changeNumber="4"/>
</ms:replaceResponse>
```

This is a sample document after the method is invoked.

```
<m:myWidgets changeNumber="4">
  <m:widget id="2" changeNumber="4">
      <m:name xml:lang="en">My new second object</m:name>
      <m:unitPrice currency="GBP">8.99</m:unitPrice>
  </m:widget>
</m:myWidgets>
```

Return Values

These are the values that may be returned by the service for this method.

Value	Response
status=success	Indicates success. Note that empty selections result in successful responses.
selectedNodeCount=0, status=failure	Indicates a missing primary (blue) element. This will only occur if the client tries to delete an object that doesn't exist any more. In the case of delete, the error can be safely ignored. For changes, the developer may want to provide some sort of error handling code (for example, the user could be notified that the object no longer exists and asked if the user wants to add it).
selectedNodeCount=1, status=failure	Indicates an internal error condition. The operation cannot be performed. In this case, the developer may want to display an error message on the client such as "service temporarily unavailable".
selectedNodeCount=0 or empty set, status=synchronize	Indicates that the change number is out of an acceptable range. In this case, the service cannot complete the method until the client application synchronizes to the service. The client must use an xpQuery to synchronize to the service.

Updating Elements

ms:updateRequest

The update request method groups multiple insert, delete, and replace operations into a single message.

```
<ms:updateRequest
  xlmns:ms="http://schemas.microsoft.com/ms/2001/1/core">
  <ms:updateBlock select="…">
      <ms:insertRequest select="…">
          {objects}
      </ms:insertRequest>
      <ms:deleteRequest select="…">
      </ms:deleteRequest>
      <ms:replaceRequest select="…">
          {objects}
      </ms:replaceRequest>
  </ms:updateBlock>
</ms:updateRequest>
```

The meaning of the attributes and elements shown in the preceding sample document fragment are listed in the following table.

Attribute	Description
select	The select attribute in the update block sets the global context of the update block in the document.

ms:updateResponse

The update response message is returned on completion of an *ms:updateRequest* method.

```
<ms:updateResponse
  xmlns:ms="http://schemas.microsoft.com/ms/2001/1/core">
  <ms:updateBlockStatus>
      <ms:insertResponse selectedNodeCount="…" status="…">
          <ms:newBlue id="…" changeNumber="…"> </ms:newBlue>
      </ms:insertResponse>
      <ms:deleteResponse selectedNodeCount="…" status="…">
          <ms:changedBlue id="…" changeNumber="…"> </ms:changedBlue>
          <ms:deletedBlue id="…" changeNumber="…"> </ms:deletedBlue>
      </ms:deleteResponse>
      <ms:replaceResponse selectedNodeCount="…" status="…">
          <ms:newBlue id="…" changeNumber="…"> </ms:newBlue>
          <ms:changedBlue id="…" changeNumber="…"> </ms:changedBlue>
          <ms:deletedBlue id="…" changeNumber="…"> </ms:deletedBlue>
      </ms:replaceResponse>
  </ms:updateBlockStatus>
</ms:updateResponse>
```

The meaning of the attributes and elements shown in the preceding sample document fragment are listed in the following table.

Attribute	Description
selectedNodeCount	The selectedNodeCount attribute contains the number of selected nodes. It is an unsigned variable length integer.
status	The status attribute indicates the status of the method. See “Return Values” below.
id	The id attribute contains the ID assigned to the xdb:blue object. It is an unsigned variable length integer. It is generated by the MSN Mobile service and is unique within the document.
changeNumber	The changeNumber attribute contains the change number for the new or modified sdb:blue object. It is an unsigned variable length integer. For more information about change numbers, see “Change Numbers” on page 38.

Examples

This is a sample request message.

```
<ms:updateRequest>
  <ms:updateBlock select="/myWidgets">
      <ms:insertRequest select=".">
          <m:widget>
              <m:name xml:lang="en">My third widget</m:name>
              <m:unitPrice currency="USD">15.00</m:unitPrice>
          </m:widget>
      </ms:insertRequest>
      <ms:replaceRequest select="./widget[@id='2']">
          <m:widget>
              <m:name xml:lang="en">My new new second object</m:name>
              <m:unitPrice currency="GBP">9.99</m:unitPrice>
          </m:widget>
      </ms:replaceRequest>
  </ms:updateBlock>
</ms:updateRequest>
```

This is a sample response.

```
<ms:updateResponse>
  <ms:updateBlockStatus>
      <ms:insertResponse selectedNodeCount="1" status="success">
          <ms:newBlue id="3" changeNumber="5"/>
      </ms:insertRequest>
      <ms:replaceResponse selectedNodeCount="1" status="success">
          <ms:changedBlue id="2" changeNumber="5"/>
      </ms:replaceResponse>
  </ms:updateBlockStatus>
</ms:updateResponse>
```

This is a sample document after the method is invoked.

```
<m:myWidgets changeNumber="5">
  <m:widget id="2" changeNumber="5">
      <m:name xml:lang="en">My new new second object</m:name>
      <m:unitPrice currency="GBP">9.99</m:unitPrice>
  </m:widget>
  <m:widget id="3" changeNumber="5">
      <m:name xml:lang="en">My third widget</m:name>
      <m:unitPrice currency="USD">15.00</m:unitPrice>
  </m:widget>
</m:myWidgets>
```

Ping

Ping is used diagnostically to ensure that a host computer you are trying to reach is actually operating. It is possible for clients to ping the MSN Mobile service to check whether or not it is actually running. More commonly, pings are used by the MSN Mobile service to notify a client that they should reconnect to the service to retrieve a high priority item such as an e-mail or rich news report. The ping typically contains the first characters of the item and provides a link to allow the user to reconnect to the service to retrieve the entire item. If a client is available to receive the ping, then the user can choose whether or not to reconnect to the service. If a client is not available to receive the ping (say, because the user is operating a mobile phone and drives through a tunnel), then the ping drops.

ms:pingRequest

The ping request method pings the client or the service. The method may contain information on the queued MSP messages.

MSP is transport independent. MSP sessions may exist over multiple transports and connections (for example, SMS and mobile IP connections). Currently mobile data infrastructure supports MO IP connections only. A MSP-based application may use the *ms:pingRequest* method over a MT connection (for example, by sending an SMS message that notifies the client that they have a message waiting and providing a link for re-establishing the IP connection).

```
<ms:pingRequest
    xmlns:ms=”http://schemas.microsoft.com/ms/2001/1/core”>
    <ms:messagePreview id=”…” objectID=”…”>0..N
        <ms:request service=”…” method=”...”>0..1</ms:request>
        <ms:response service=”…”>0..1</ms:response>
        <ms:emailAddress>0..1</ms:emailAddress>
        <ms:plainBody>0..1</ms:plainBody>
    </ms:messagePreview>
    {any}
</ms:pingRequest>
```

The meaning of the attributes and elements shown in the preceding sample document fragment are listed in the following table.

Attribute	Description
id	The id attribute contains the id of the queued MSP message. It is an unsigned variable length integer.
objectID	The objectID atrribute contains the ID of an object affected by the queued MSP message. **Note:** The messaging service passes the callID in the objectID. The myInbox service passes the ID of the e-mail.
ms:request	The request element contains the request element of the queued MSP message.
ms:response	The response element contains the response element of the queued MSP message.
ms:emailAddress	The emailAddress element contains either the screen name of email address of the originator of the queued MSP message. It is truncated to 15 characters. **Note:** Not all MSP messages have an originator. This element typically contains the sender of an instant message or e-mail.
ms:plainBody	The plainBody element contains a description of the queued MSP message (for example, “news alert”, “stock quote” or the subject of an e-mail message). It is truncated to 20 characters.

Example

A sample ping for a queued e-mail alert.

```
<s:Envelope>
  <s:Header>
      <srp:path>
          <srp:to>”somebody@msn.com”</srp:to>
          <srp:from>”http://myservices.mobile.msn.com”</srp:from>
          <srp:id>245</srp:id>
      </srp:path>
      <ms:request service=”mobileMyServices” method=”ping”
genResponse=”always”/>
  </s:Header>
  <s:Body>
      <ms:pingRequest>
          <ms:message id=”244”>
              <ms:request service=”myInbox” method=”notification”/>
              <ms:emailAddress>Darren Apfel</ms:emailAddress>
              <ms:plainBody>Re: Offline alerts</ms:plainBody>
          </ms:message>
      </ms:pingRequest>
  </s:Body>
</s:Envelope>
```

ms:pingResponse

The *ms:pingResponse* message is returned on receipt of a ping request.

```
<ms:pingResponse
  xmlns:ms=”http://schemas.microsoft.com/ms/2001/1/core”>
  {any}
</ms:pingResponse>
```

Interpreting Fault Responses

SOAP returns fault responses for errors in executing Authenticate and Request primitives.

This is a sample response.

```
<s:Fault>
    <s:faultcode>1..1</s:faultcode>
    <s:faultstring>0..1</s:faultstring>
    {any}
</s:Fault>
```

The meaning of the attributes and elements shown in the preceding sample document fragment are listed in the following table.

Attribute	Description
s:faultcode	The faultcode element must be present in a SOAP fault response. Valid values are: • VersionMismatch – The version number of the protocol is not supported. • ClientMismatch – The message was incorrectly formed or did not contain the required information (for example, there was no body tag). • Server – The message was correctly formed, but the MSN Mobile service did not respond.

	• LicenseExpired – The MSP session has expired. It is necessary for the client to re-authenticate.
s:faultstring	The faultstring attribute contains an explanation of the fault. The fault description is intended to be readable, not processed algorithmically. It is optional.

Synchronizing Data

Clients are responsible for handling synchronization of data with the MSN Mobile service. They use change numbers to keep track of changes in the user's data since the user's last use of the device, and then retrieve only the data that has changed. Clients must also have the capacity to receive large messages in partial amounts, called chunks. By receiving chunks of messages and displaying their content to a user, the end-user experience is improved because the user does not have to wait for large messages to download in their entirety before beginning to access their content.

Change Numbers

A client's MSN Mobile data may change as a result of many possible factors. For example, the client may have changed contacts information or e-mail settings on either the PC-based or mobile versions of Hotmail or Messenger, or a user's contact may have updated its properties. Each change information results in an update to a change number (that is, the changeNumer attribute of the xdb:blue object). Note that for MSN Messenger, changes in presence do not result in a change number update.

The changeNumber attribute is incremented whenever the xdb:blue object is changed. The changeNumber is propagated up to all xdb:blue ancestor nodes. A client that caches mobile service data uses the changeNumber attribute in conjunction with a change query to synchronize the cache. The service returns a response with status="synchronize" if it cannot synchronize the cache. In that case, the client should send an xpQuery to restore the cache.

Note: Setting the includeObjectChangeNumbers device capability to “no” suppresses object change numbers.

Usage Scenario

Joe User uses MSN Messenger on both his PC and his phone. He logs into Messenger on his phone and has an instant messaging session with a buddy. The MSN Mobile service tracks the changeNumber as 10. A while later, Joe's account receives notification that one of Joe's buddies, somebody@hotmail.com has changed its friendly name to “Some Body”. This silently increments the change number from 10 to 11. Joe's phone is set to receive real time property updates, so it receives a push notification that the friendly name for one of his contacts has changed. The push notification includes the change number 11. Joe's client processes the change number and decides not to re-synchronize with the service at this time, but to silently drop the change until Joe logs on again. The next time Joe logs on to Messenger using his mobile phone, the client retrieves just the changes that have taken place since he last logged on to the phone.

Examples

This is an example of a message before a request occurs.

```
<m:myWidgets changeNumber="11">
  <m:widget id="4" changeNumber="8">
      <m:name xml:lang="en">Widget 4</m:name>
      <m:unitPrice currency="USD">8.99</m:unitPrice>
  </m:widget>
  <m:widget id="5" changeNumber="10">
      <m:name xml:lang="en">Widget 5</m:name>
      <m:unitPrice currency="USD">128.00</m:unitPrice>
  </m:widget>
</m:myWidgets>
```

This is a sample request message.

```
<ms:queryRequest>
  <ms:changeQuery select="/" baseChangeNumber="8"/>
</ms:queryRequest>
```

This is a sample response.

```
<ms:queryResponse>
  <ms:changeQueryResponse baseChangeNumber="11"
      status="success">
      <ms:deletedBlue id="9" changeNumber="9"/>
      <m:widget id="5" changeNumber="10">
          <m:name xml:lang="en">Widget 5</m:name>
          <m:unitPrice currency="USD">128.00</m:unitPrice>
      </m:widget>
          <ms:deletedBlue id="10" changeNumber="11"/>
  </ms:changeQueryResponse>
</ms:queryResponse>
```

Segmentation

Some service responses may be too large for the mobile device to process in a single message. Examples are large xpQuery and changeQuery responses. When this occurs, the service may segment large responses based on the maxPDUSize device capability. The sections are atomic MSP messages. The breaks in the message are determined by the object boundaries.

Clients can keep track of a message that has been broken into sections by tracking the unique ID and relatesTo ID numbers. Each of the MSP message sections has a unique ID. Additionally, the MSP messages have the same relatesTo ID. And objects will have unique change numbers. It should be possible for the client application to reassemble segmented messages using the ID numbers regardless of the order in which they were received from the service. In the event that one or more of the message sections are missing, then the client application should identify the missing sections by using the previousChangeNumber element in the response header. In this way, the client should recover from a missing message by issuing a new changeQuery to the service which contains a baseChangeNumber for the last message received in sequence.

Note: Multiple MSP messages may be sent in a single PDU. The PDU must still be within the maxPDUSize for the device. This is a sample request message.

Examples

This is a sample original response.

```
<s:Envelope>
  <s:Header>
      <srp:path>
          <srp:from>"http://mywidgets.mobile.msn.com"</srp:from>
          <srp:id>23</srp:id>
          <srp:relatesTo>202</srp:relatesTo>
      </srp:path>
      <ss:license>12</ms:license>
      <ms:response service="myWidgets" previousChangeNumber="8"
          changeNumber="11"/>
  </s:Header>
  <s:Body>
      <ms:queryResponse>
          <ms:changeQueryResponse baseChangeNumber="11"
              status="success">
              <ms:deletedBlue id="9" changeNumber="9"/>
              <m:widget id="5" changeNumber="10">
                  <m:name xml:lang="en">Widget 5</m:name>
                  <m:unitPrice currency="USD">128.00</m:unitPrice>
              </m:widget>
                      <ms:deletedBlue id="10" changeNumber="11"/>
          </ms:changeQueryResponse>
      </ms:queryResponse>
  </s:Body>
</s:Envelope>
```

This is a sample segmented response.