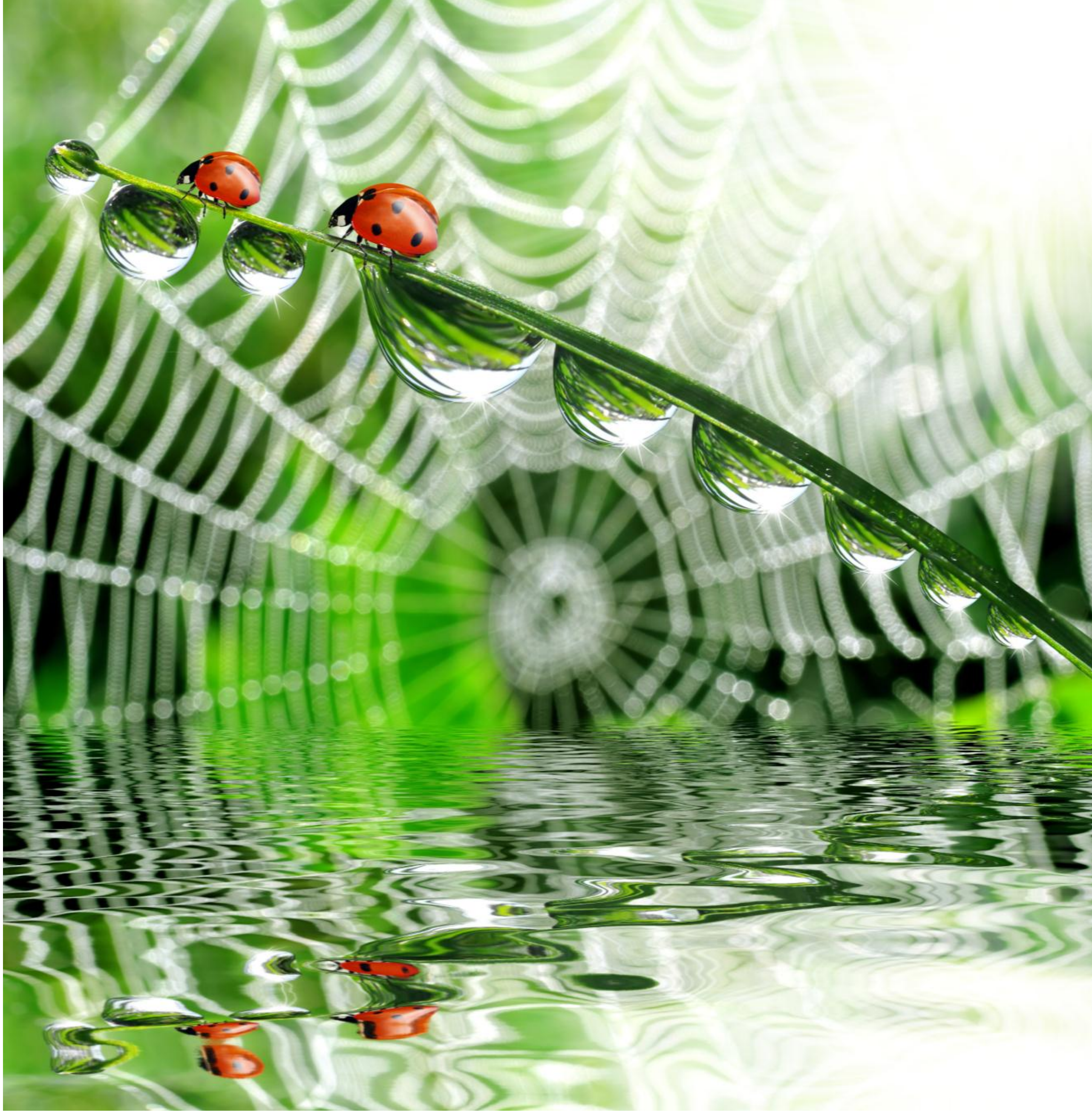
TWO-WAY EMAIL & SMS MESSAGING

m:science
every message counts



SMS WEB SERVICE

Product White Paper

Website: www.m-science.com

Telephone: 01202 241120

Email: enquiries@m-science.com

Contents

Introduction

The M:Science SMS Web Service is a comprehensive two-way SMS messaging service that provides a website, web service or application developer with the ability to incorporate methods to send, receive and track SMS messages via an internet link, without the need to install any additional client applications or components.

The service supports full discovery protocols and hence can simply be added as a Web Reference to any Visual Studio 2003 project.

The service supports the industry standard protocols:

- SOAP V1.1 as specified at http://schemas.xmlsoap.org/soap/encoding/
- HTTP V1.1 as specified in RFC 2068

To use the service the user must first create an M:Science internet account through the company web site at www.m-science.com and then enable the account to use the SMS Web Service by contacting our sales staff.

Blocks of messages may then be purchased in advance through the internet account.

Product Components

Web Service

Hosted at M:Science, this .NET web service allows external users to connect via HTTP or SOAP, send, receive and monitor messages or configure asynchronous inbound routing of messages.

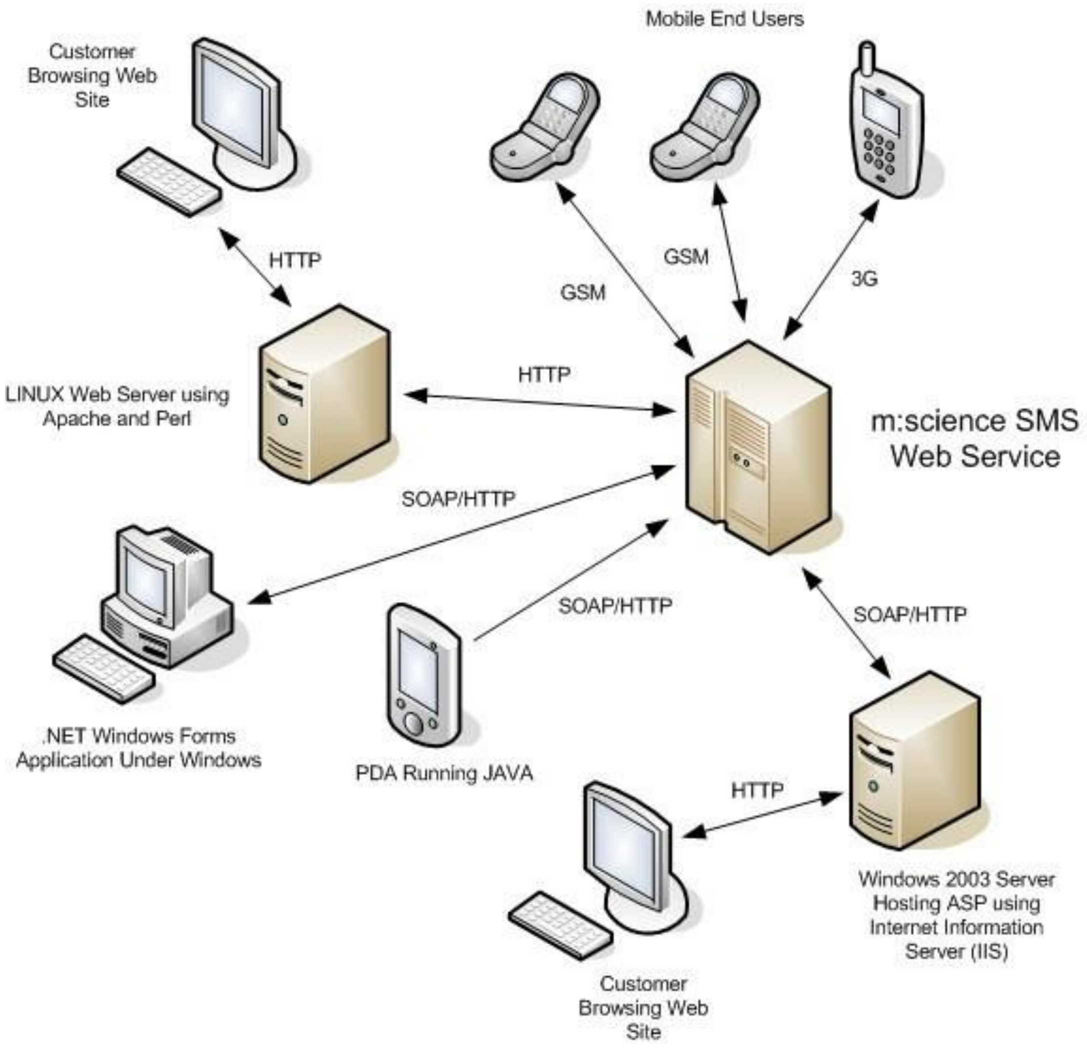
Each method provides basic security for encrypting account identifiers and passwords, but to achieve total security a secure sockets layer (SSL) may be utilised by prefixing the web service address with 'https://' instead of 'http://'.

Web Push Service

Asynchronously routes incoming messages via HTTP PUSH to one or more registered URLs.

Architecture

The M:Science SMS Web Service is an application programming interface (API) accessible using standard internet protocols which are completely vendor, platform and language independent. This enables multiple methods of connection, as shown in the figure below.


Customer Browsing Web Site
Mobile End Users
HTTP
GSM
GSM
3G
LINUX Web Server using Apache and Perl
HTTP
m:science SMS Web Service
SOAP/HTTP
SOAP/HTTP
SOAP/HTTP
.NET Windows Forms Application Under Windows
PDA Running JAVA
HTTP
Windows 2003 Server Hosting ASP using Internet Information Server (IIS)
Customer Browsing Web Site

Outbound Messaging

Sending Messages

To send a message, the SendMessage method of the Web Service can be used. The following information is required to use this method:

- **Encryption** - True if the account and password are encrypted.
- **Account ID** - Internet account ID.
- **Password** - Password as specified on the website
- **Source Tag** - Text string to identify the sending application.
- **SourceAddress** - Set to the inbound number if you wish the end mobile user to be able to reply.
- **DeliveryReceipt** - Set to 1 if a delivery receipt is required.
- **ProductMessageID** - A user defined identifier which can be used to tie a delivery receipt to a sending message.
- **Destination** - Destination phone number in standard international format e.g. '+447712345678'
- **Message** - Text of the message to send

All of the fields after the Account ID and Password are passed as arrays of values. This allows the user to specify up to five messages in one command. Messages over 160 characters are split into multiple messages automatically. Note that an Account ID and Password are always required but if available, the service uses cookies to improve speed by preventing recurrent logging in.

Example (using Python):

```
import urllib

f = urllib.urlopen("http://www.m-science.com/
MScienceSMSWebService/MScienceSMSWebService.asmx/SendMessages?encrypt=True
&accoun
tID=1&password=ghtyuik&sourceTag=Test&sourceAddress=%2b447712345678&deliver
yReceip
t=0&productMessageID=0&destination=%2b447787654321&message=TestMessage")
```

Note: %2b being the hexadecimal value for '+' as Python cannot use '+' preceding the mobile number.

print f.read()

An identifier for the message is sent back as a part of the response. For Example: OK-224355,299,5,0

Which translates to:

Status-MessageIdentifier, MessageBalance, PendingMessages, SurchargeBalance

Monitoring Messages

The identifier sent back from the SendMessage command can be used to track the status of the message by calling GetMessageStatus.

The following fields are required:

- **Encryption** - True if the account and password are encrypted.
- **Account ID** - Internet account ID.
- **Password** - Password as specified on the website.
- **MessageID** - The identifier of the message to check. This is an array parameter which can be used to check up to ten messages in one call.

Example (using Python):

```
import urllib

f = urllib.urlopen("http://www.m-science.com/MScienceSMSWebService.asmx/GetMessageStatus?encrypt=True&accountID=1&pa ssword=fghtygh&MessageID=224355")

print f.read()
```

An example response would be: OK-PENDING

Which translates to: Function call successful, message send is pending.

Inbound Messaging

Messages can either be polled and retrieved from the server by the client or pushed to the client using one or more registered URLs.

Pulling Messages

Receiving Messages

The client can check for new messages by calling GetNextInboundMessages on the SMS Web Service. This will send back up to five of the queued inbound messages. To retrieve more messages, these messages must first be acknowledged using the acknowledge procedure described below.

The following fields are required:

- **Encryption** - True if the account and password are encrypted.
- **Account ID** - Internet account ID.
- **Password** - Password as specified on the website.

Example (using Python):

```
import urllib

f = urllib.urlopen("http://www.m-
science.com/MScienceSMSWebService.asmx/GetNextInboundMessages?encrypt=True&
accountI D=1&password=fjgktiu")

print f.read()
```

An example response would be:

```
'OK-224355,+447877123456,+447977321654,15/11/2004 12:00:00, 1, 12, Test Message
```

Which translates to: MessageID of 224355, source and destination phone numbers, receipt date, user data, message length and message text.

Multiple responses are returned for more than one message.

Acknowledging Message Receipt

In order to retrieve the next batch of inbound messages, the receipt of the previous batch must first be acknowledged. This can be done using the AckMessages function.

The following fields are required:

- **Encryption** - True if the account and password are encrypted.
- **Account ID** - Internet account ID.
- **Password** - Password as specified on the Web Site
- **MessageID** - Identifier of the message to acknowledge. This parameter is an array which can specify up to ten message IDs if so desired.

Example (using Python):

```
import urllib

f = urllib.urlopen("http://www.m-science.com /
MScienceSMSWebService/MScienceSMSWebService.asmx/AckMessages?encrypt=True&
accountI D=1&password=fjgktiu&MessageID=224355")

print f.read()
```

An example response would be:

```
'OK-SUCCEEDED'
```

Which translates to method succeeded as did the message acknowledge.

Pushing Messages

Using the SMS Web Service it is possible to configure an account so that inbound messages are routed via HTTP PUSH to one or more URLs. This service must be activated for an account in advance stating the maximum number of URLs the user wishes to register. Each inbound number can also be configured to route either through the M:Science SMS Server product, to the Web Service (i.e. read using GetMessages) or to the asynchronous Web Push Service. To begin receiving asynchronous messages, the user first makes a call onto the Web Service to register the requested URL. This can be removed at a later date by another API call. It is also possible to monitor the state of a particular URL to determine if the messages are being dispatched successfully.

Writing a suitable URL

A client can be anything which accepts an HTTP PUSH. This might be for example an ActiveServerPage (ASP) script, a cgi (.cgi) script (using Perl or similar) or another .NET web service.

The fields passed with the call are as follows:

- **Destination** - destination phone number.
- **Source** - Source phone number.
- **ProductMessageID** - user data specified
- **ReceiveDate** - Date message was received at this end.
- **DeliveryReceipt** - If the message is a receipt for a previous send
- **ReceiptDate** - When the receipt was generated.
- **Message** - Message text
- **MessageID** - Identifer for the message.

Example (using C#):

```
[WebMethod]

public string NewMessage(string Destination, string Source, string ProductMessageID, string ReceiveDate, string DeliveryReceipt, string ReceiptDate, string Message, string MessageID)

{
```

```
StringBuilder sb = new StringBuilder(); sb.Append("Received message: ");
sb.AppendFormat("Destination={0}, ", Destination); sb.AppendFormat("Source={0}, ",
Source); sb.AppendFormat("ProductMessageID={0}, ", ProductMessageID);
sb.AppendFormat("ReceiveDate={0}, ", ReceiveDate);
sb.AppendFormat("DeliveryReceipt={0}, ", DeliveryReceipt);
sb.AppendFormat("ReceiptDate={0}, ", ReceiptDate); sb.AppendFormat("Message={0}, ",
Message); sb.AppendFormat("MessageID={0}", MessageID);
WriteToLog(LogLevel.information, sb.ToString());

return "OK-RECEIVED";

}
```

The server expects a response of ‘OK-RECEIVED’ before moving onto the next message in the queue. If this is not received, the server will retry sending the message.

Registering a URL

In order to start receiving messages, the client URL (or URLs) must be registered with the M:Science SMS Web Service using the RegisterForInbound method. The account must be activated for message pushing for this to succeed.

The following fields are required:

- **Encryption** - True if the account and password are encrypted.
- **Account ID** - Internet account ID.
- **Password** - Password as specified on the website.
- **Address** - URL to call when a message arrives. This is an array and can accept multiple URLs if so desired.

Example (using Python):

```
import urllib

f = urllib.urlopen(http://www.m-
science.com/MScienceSMSWebService/MScienceSMSWebService.asmx/RegisterForInbo
und?e
ncrypt=True&accountID=1&password=fjgktiu&address=http://mywebsite.com/newmess
age.a
```

sp)

print f.read()

An example response would be:

‘OK-REGISTERED’

Which translates to method succeeded as did the URL registration.

Monitoring a URL

Monitoring URL status is possible using the GetURLStatus method. This will indicate if there are problems sending messages through to the client. In the event of a bad communications link, the server will retry automatically.

The fields required are the same as those for registering the message with the exception that only one URL can be specified.

An example response would be:

‘OK-READY’

Which translates to method succeeded and the URL status is ready to receive new messages.

Unregistering a URL

A URL may be un-registered using the UnregisterInboundMessage method. This will indicate if there are problems sending messages through to the client. In the event of a bad communications link, the server will retry automatically.

The fields required are the same as those for registering the message. An example response would be:

‘OK-UNREGISTERED’

Which translates to method succeeded as did URL unregistration.

There is also another method called ResetInboundCallbacks which removes all URL registrations for a selected account.

Development Kit

Documentation

A comprehensive manual is provided in PDF format which contains examples of how to write software to communicate with the M:Science SMS Web Service.

Sample Code

Sample code is provided for accessing the Web Service using the following technologies:

VB.NET

VB6

C++.NET C#

Perl ASP Python Java

HTML

Service Availability

To ensure optimum service availability we utilise a wide bandwidth leased line premium internet connection to and from the data centre.

In addition, continuity of service is maintained through multiple levels of hardware, software and communications redundancy, including:

- Multiple load balanced servers
- Uninterruptable power supplies, including standalone generator backup
- Dynamic resilient message delivery routeing, ensuring high service availability at all times