



**ProgettoSMS**

**Manuale Gateway Web Services**

# Indice

Indice.....	2
Introduzione .....	3
http Web Services .....	4
Risposta .....	4
Descrizione delle costanti .....	5
AnswerRecipientType.....	5
ErrorID .....	5
ProtocolType .....	5
Statement .....	5
Descrizione degli oggetti .....	6
Oggetto Answer .....	6
Oggetto Request .....	6
Oggetto Message .....	6
Oggetto AnswerRecipients .....	6
Oggetto AnswerRecipient .....	6
Oggetto Contacts .....	7
Oggetto Contact .....	7
Oggetto GetUserStatusAnswer .....	7
Oggetto GetUserStatusRequest .....	7
Oggetto GetIncomingMessagesAnswer .....	7
Oggetto GetIncomingMessagesRequest .....	7
Oggetto GetMessageStatusAnswer .....	8
Oggetto GetMessageStatusRequest .....	8
Oggetto SendMessageAnswer .....	8
Metodi: .....	8
SendMessage .....	10
Risposta positiva a SendMessage .....	10
Esempi SendMessage .....	10
Esempio 1:.....	10
Esempio 2:.....	11
GetMessageStatus .....	13
Esempi GetMessageStatus .....	13
Esempio 1:.....	13
GetUserStatus .....	14
Esempi GetUserStatus .....	14
Esempio 1:.....	14
GetIncomingMessages .....	15
Esempi GetIncomingMessages .....	15
Esempio 1:.....	15
Appendice A: possibili stati dei messaggi .....	16
Appendice B: possibili stati di consegna dei messaggi .....	17
Appendice C: codici di errore .....	18

## **Introduzione**

I seguenti manuali contengono le modalità di utilizzo dei protocolli per la connessione diretta tra un applicativo ed il gateway di ProgettoSMS srl per l'invio di messaggi.

Attualmente il sistema di inoltro messaggi prevede tre diversi sistemi di trasmissione, HTTP, FTP ed SMTP.

Attraverso il protocollo HTTP è possibile inviare messaggi direttamente come URL (metodo GET), oppure attraverso il metodo POST, oppure utilizzando lo standard definito per i Web Services (XML/SOAP).

Attraverso il protocollo FTP è possibile inviare messaggi trasferendo un file che può essere in formato testo oppure in formato XML.

Infine, attraverso il protocollo SMTP è possibile inviare messaggi specificando le informazioni direttamente nel body della mail.

Nel caso di invio attraverso HTTP il server ProgettoSMS provvede a processare immediatamente le informazioni e ad inviare il risultato attraverso HTTP. Nel caso di FTP ed SMTP è possibile ricevere la risposta attraverso differenti canali (mail e/o sms)

I file e le richieste devono essere codificate con il sistema UTF-8.

## *http Web Services*

Tramite il protocollo HTTP formattato tramite la sintassi SOAP è possibile inviare messaggi ed interrogare lo stato del server e dell'account.

In generale qualunque comando invocato deve contenere i campi per l'identificazione e l'autenticazione del cliente che sta eseguendo la richiesta (User e Password).

Il resto delle informazioni possono variare a seconda della richiesta.

E' possibile richiamare la pagina di descrizione del servizio mediante l'indirizzo:

[http://<ip\\_and\\_dirbase>/WSGateway/Gateway.asmx](http://<ip_and_dirbase>/WSGateway/Gateway.asmx)

E' possibile scaricare il file .wsdl contenente il contratto all'indirizzo:

[http://<ip\\_and\\_dirbase>/WSGateway/Gateway.asmx?wsdl](http://<ip_and_dirbase>/WSGateway/Gateway.asmx?wsdl)

In tale file sono descritti tutti i tipi di oggetti utilizzati/utilizzabili dai comandi.

I comandi invocabili sono i seguenti:

Comando	Descrizione
SendMessage	Consente l'invio di un messaggio ad uno o più destinatari
GetMessageStatus	Consente di recuperare lo stato di un messaggio
GetUserStatus	Consente di recuperare le informazioni sull'utente (crediti)
GetIncomingMessages	Consente di recuperare gli sms ricevuti sulla piattaforma

## **Risposta**

La richiesta viene elaborata dal web service il quale eventualmente ritorna come risultato un oggetto complesso la cui struttura dipende dal comando invocato.

## Descrizione delle costanti

### AnswerRecipientType

AnswerRecipientType.None = 0  
AnswerRecipientType.Email = 1

### ErrorID

ErrorID.None = 0  
ErrorID.SyntaxError = 100  
ErrorID.ParameterSyntaxError = 110  
ErrorID.ParameterValueNotValid = 112  
ErrorID.MissingParameter = 114  
ErrorID.MissingStatement = 120  
ErrorID.StatementNotRecognized = 125  
ErrorID.ErrorInFixedList = 130  
ErrorID.ErrorInList = 135  
ErrorID.DisabledFunctionForReseller = 200  
ErrorID.LoginError = 300  
ErrorID.MessageNotFound = 500  
ErrorID.MessageNotFoundForUser = 510

### ProtocolType

ProtocolType.None = 0  
ProtocolType.http = 1  
ProtocolType.ftpText = 2  
ProtocolType.ftpXML = 3  
ProtocolType.smtp = 4

### Statement

Statement.None = 0  
Statement.SendMessage = 1  
Statement.GetMessageStatus = 2  
Statement.GetUserStatus = 3  
Statement.GetIncomingMessages = 4

## Descrizione degli oggetti

### Oggetto Answer

Rappresenta un risposta generica, questo oggetto viene utilizzato come base per la definizione delle risposte più complesso.

#### Proprietà:

**ErrorDescription:** stringa contenente la descrizione dell'errore.

**ErrorID:** codice numerico identificativo dell'errore, di tipo ErrorID.

**ReqID:** stringa contenente l'identificativo della richiesta.

### Oggetto Request

Rappresenta una richiesta generica, questo oggetto viene utilizzato come base per la definizione di richieste più complesse.

#### Proprietà:

**AnswerRecipients:** array di oggetti AnswerRecipient.

**User:** stringa contenente l'utente da specificare per ottenere la corrispondente risposta.

**Password:** stringa contenente la password da specificare per ottenere la corrispondente risposta.

**ReqID:** stringa contenente l'identificativo della richiesta.

**Statement:** descrittore della tipologia della richiesta, di tipo Statement.

### Oggetto Message

Rappresenta un messaggio da spedito con tutte le informazioni che lo riguardano.

#### Proprietà:

**AdC:** rappresenta il numero del destinatario del messaggio (Address Code) è un oggetto di tipo AdC.

**ID:** codice numerico rappresentante l'identificativo univoco del messaggio.

**IDContact:** numero intero rappresentante l'identificativo del contatto a cui è stato inviato.

**Result:** stringa che rappresenta il risultato dell'invio del messaggio.

### Oggetto AnswerRecipients

Insieme di oggetti AnswerRecipient. Rappresenta una lista di mezzi e parametri per inviare le risposte oltre che con il web-service.

### Oggetto AnswerRecipient

#### Proprietà:

**Type:** è un valore costante di tipo AnswerRecipientType che indica la tipologia del Parametro dell'oggetto AnswerRecipient. Può assumere ad esempio il valore `Email` per indicare di spedire la risposta del servizio anche ad un indirizzo e-mail.

**Parameter:** è un valore di tipo stringa che indica l'indirizzo del mezzo del tipo specificato con il parametro type. Ad esempio potrebbe essere `r.biagio@target.it`.

## Oggetto Contacts

Insieme di oggetti Contact. Rappresenta l'insieme di numeri dei destinatari del messaggio.

## Oggetto Contact

### Proprietà:

**PortablePhone:** valore stringa rappresentante il numero di telefono del contatto (es. +393483168407).

## Oggetto GetUserStatusAnswer

E' un oggetto che rappresenta una risposta, deriva da un oggetto Answer quindi ne eredita tutte le caratteristiche.

### Proprietà:

**User:** stringa contenente l'utente utilizzato per l'invio dei messaggi.

**Credits:** elenco di nazioni con il relativo credito residuo dopo l'invio di messaggi.

## Oggetto GetUserStatusRequest

E' un oggetto che rappresenta una richiesta, deriva da un oggetto Request quindi ne eredita tutte le caratteristiche.

## Oggetto GetIncomingMessagesAnswer

E' un oggetto che rappresenta una risposta, deriva da un oggetto Answer quindi ne eredita tutte le caratteristiche.

### Proprietà:

**Messages:** elenco di incoming message.

## Oggetto GetIncomingMessagesRequest

E' un oggetto che rappresenta una richiesta, deriva da un oggetto Request quindi ne eredita tutte le caratteristiche.

## Oggetto GetMessageStatusAnswer

E' un oggetto che rappresenta una risposta, deriva da un oggetto Answer quindi ne eredita tutte le caratteristiche.

### Proprietà:

**ID:** numero intero che rappresenta l'identificativo del messaggio.

**OAdC:** stringa rappresentante il mittente.

**AdC:** oggetto di tipo Adc che rappresenta il numero del destinatario del messaggio (Address Code).

**DT:** data e ora di invio del messaggio nel caso in cui sia già stato inviato (o sia avvenuto un tentativo di invio).

**DDT:** data e ora di invio del messaggio nel caso sia richiesto un invio differito.

**Status:** stringa contenente lo stato del messaggio.

**StatusDescription:** stringa contenente la descrizione dello stato del messaggio.

**Reason:** stringa contenente un ulteriore dettaglio nel caso in cui il messaggio non sia stato inviato (esempio la descrizione di un errore interno o la descrizione del messaggio del gateway).

**DeliveryReport:** True o False a seconda che per il messaggio sia stato richiesto lo stato di consegna

**DeliveryReportStatus:** lo stato di consegna del messaggio

**DeliveryReportStatusDescription:** la descrizione dello stato di consegna del messaggio

**DeliveryReportDateTime:** l'istante in cui è cambiato per l'ultima volta lo stato del delivery report

## Oggetto GetMessageStatusRequest

E' un oggetto che rappresenta una richiesta, deriva da un oggetto Request quindi ne eredita tutte le caratteristiche.

### Proprietà:

**MessageID:** numero intero che rappresenta l'identificativo del messaggio di cui si desidera recuperare lo stato.

## Oggetto SendMessageAnswer

### Proprietà:

**Sent:** numero intero rappresentante il numero di messaggi spediti.

**Errors:** numero intero rappresentante l'eventuale errore.

**Credits:** elenco di nazioni con il relativo credito residuo dopo l'invio di messaggi.

**MessageSents:** array di oggetti Msg.

### Metodi:

**GetIncomingMessagesAnswer** GetIncomingMessages (**string** userName , **string** password, **DateTime?** startDate, **DateTime?** endDate, **string** sender)

**GetIncomingMessagesAnswer** GetIncomingMessages1 (**String** userName , **String** password, **DateTime?** startDate, **DateTime?** endDate, **string** sender , **AnswerRecipient[]**)

**GetUserStatusAnswer** GetUserStatus (**string** userName , **string** password)

**GetUserStatusAnswer** GetUserStatus1 (**string** userName , **string** password , **AnswerRecipient[]** answerRecipients)



---

**GetUserStatusAnswer** GetUserStatus2 (**GetUserStatusRequest** *getUserStatusRequest*)

**GetMessageStatusAnswer** GetMessageStatus (**string** *userName* , **string** *password* , **Int32** *messageID*)

**GetMessageStatusAnswer** GetMessageStatus1 (**string** *userName* , **string** *password* , **Int32** *messageID* ,  
**AnswerRecipient[]** *answerRecipients*)

**GetMessageStatusAnswer** GetMessageStatus10 (**string** *userName* , **string** *password* , **Int32[]** *messagesIDs*)

**GetMessageStatusAnswer** GetMessageStatus11 (**string** *userName* , **string** *password* , **Int32[]** *messagesIDs* ,  
**AnswerRecipient[]** *answerRecipients*)

**SendMessageAnswer** SendMessage (**string** *userName* , **string** *password* , **string** *OAdC* , **Contact** *contact* , **string**  
*message*)

**SendMessageAnswer** SendMessage1 (**string** *userName* , **string** *password* , **string** *OAdC* , **Contact[]** *contacts* ,  
**String** *message*)

**SendMessageAnswer** SendMessage2 (**string** *userName* , **string** *password* , **Contact** *contact* , **string** *message*)

**SendMessageAnswer** SendMessage3 (**string** *userName* , **string** *password* , **Contact[]** *contacts* , **string** *message*)

**SendMessageAnswer** SendMessage10 (**string** *userName* , **string** *password* , **string** *OAdC* , **Contact[]** *contacts* ,  
**string** *message* , **AnswerRecipient[]** *answerRecipients* , **DateTime** *DDT*)

## SendMessage

Consente l'invio di un messaggio ad uno o più destinatari.

Il comando può essere invocato mediante le seguenti chiamate:

```
SendMessageAnswer SendMessage(string userName, string password, string OAdC, Contact contact, string message)
```

```
SendMessageAnswer SendMessage1(string userName, string password, string OAdC, Contact[] Contacts, string message)
```

```
SendMessageAnswer SendMessage2(string userName, string password, Contact contact, string message)
```

```
SendMessageAnswer SendMessage3(string userName, string password, Contact[] contacts, string message)
```

```
SendMessageAnswer SendMessage10(string userName, string password, string OAdC, Contact[] contacts, string message, AnswerRecipient[] answerRecipients, DateTime DDT)
```

I parametri utilizzabili durante l'invio di un messaggio sono i seguenti:

Parametro	Opz	Descrizione
User	Obb	Nome dell'utente
Password	Obb	Password dell'utente
AnswerRecipients	Opz	Lista di mezzi e parametri per inviare le risposte in aggiunta al servizio.
OAdC	Opz	Originator Address Code. Mittente. A seconda del gateway di invio e del contratto stipulato dal cliente potrebbe essere ignorato. Il formato del campo (alfanumerico o numerico) può variare a seconda del contratto stipulato e del gateway di invio.
Contacts	Obb	Numero o lista di numeri destinatari. Il formato del campo deve essere +<prefisso_internazionale><prefisso_nazionale><numero> oppure <prefisso_nazionale><numero>
Message	Obb	Messaggio. Può contenere al più 160 caratteri.
DDT	Opz	Deferred Delivery Time. Ora di invio differita nel formato DateTime. Se non specificato i messaggi vengono inviati appena possibile.

**Nota Bene:** nel caso in cui AdCs contenga una lista di destinatari, il controllo dei crediti avviene ad ogni singolo invio del messaggio. Una volta che i crediti sono esauriti il resto dei messaggi non vengono inviati.

## Risposta positiva a SendMessage

Se la chiamata al comando non ha successo, si verificherà una eccezione intercettabile da codice. Se il comando chiamato ha avuto successo, verrà restituito un oggetto `SendMessageAnswer` (vedi descrizione nel file `.wsdl`).

## Esempi SendMessage

### Esempio 1:

Invio del messaggio 'Chiamami' al numero +393333333333 con mittente 'Io':

```
Using ProjectTest.WSGateway;

namespace ProjectTest
{
    class Class1
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main(string[] args)
        {
            Gateway gw = new Gateway();

            Contact contact = new Contact();
            contact.PortablePhone = "+393333333333";

            SendMessageAnswer answer = gw.SendMessage("mioNomeUtente", "miaPassword", "Io",
            contact, "Chiamami");

            Console.WriteLine("{0}", answer.Sent);

            foreach (ProgettoSMS.CustomerCredit cc in answer.Credits)
                Console.WriteLine("{0}: {1}", cc.Country, cc.Credits);

            Console.ReadLine();
        }
    }
}
```

## Esempio 2:

Invio del messaggio 'Chiamami' ai numeri +393477777777 e +393333333333 con mittente 'Io'

```
Using ProjectTest.WSGateway;

namespace ProjectTest
{
    class Class1
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main(string[] args)
        {
            Gateway gw = new Gateway();

            Contact[] contacts = new Contact[2];
            contacts[0] = new Contact();
            contacts[0].PortablePhone = "+393477777777";

            contacts[1] = new Contact();
            contacts[1].PortablePhone = "+393333333333";

            SendMessageAnswer SendMessagel("mioNomeUtente", "miaPassword", "Io", contacts,
            "Chiamami");

            Console.WriteLine("{0}", answer.Sent);
        }
    }
}
```



---

```
foreach (ProgettoSMS.CustomerCredit cc in answer.Credits)
    Console.WriteLine("{0}: {1}", cc.Country, cc.Credits);
Console.ReadLine();
}
```

## GetMessageStatus

Consente di recuperare lo stato di un messaggio.

Il comando può essere invocato mediante le seguenti chiamate:

```
GetMessageStatusAnswer GetMessageStatus(string userName, string password, int messageID)
```

```
GetMessageStatusAnswer GetMessageStatus1(string userName, string password, int messageID,  
AnswerRecipient[] answerRecipients)
```

```
GetMessageStatusAnswer GetMessageStatus10(string userName, string password, int[]  
messagesIDs)
```

```
GetMessageStatusAnswer GetMessageStatus11(string userName, string password, int[]  
messagesIDs, AnswerRecipient[] answerRecipients)
```

## Esempi GetMessageStatus

### Esempio 1:

Richiesta di informazioni sul messaggio 263154172 da parte dell'utente mioNomeUtente con password miaPassword

```
Using ProjectTest.WSGateway;
```

```
namespace ProjectTest  
{  
    class Class1  
    {  
        /// <summary>  
        /// The main entry point for the application.  
        /// </summary>  
        [STAThread]  
        static void Main(string[] args)  
        {  
            Gateway gw = new Gateway();  
  
            AnswerRecipient[] answerRecipients = new AnswerRecipient[1];  
            answerRecipients[0] = new AnswerRecipient();  
            answerRecipients[0].Type = AnswerRecipientType.Email;  
            answerRecipients[0].Parameter = "me@mydom.com";  
  
            GetMessageStatusAnswer answer = gw.GetMessageStatus1("mioNomeUtente ",  
            "miaPassword", 263154172, answerRecipients);  
  
            Console.WriteLine("{0} {1}", answer.MessagesStatus[0].ID,  
            answer.MessagesStatus[0].Adc.Number);  
  
            Console.ReadLine();  
        }  
    }  
}
```

## GetUserStatus

Consente di recuperare le informazioni sull'utente (crediti).

Il comando può essere invocato mediante le seguenti chiamate:

```
GetUserStatusAnswer GetUserStatus(string userName, string password)
```

```
GetUserStatusAnswer GetUserStatus1(string userName, string password, AnswerRecipient[] answerRecipients)
```

## Esempi GetUserStatus

### Esempio 1:

Richiesta delle proprie informazioni da parte dell'utente mioNomeUtente con password miaPassword.

```
Using ProjectTest.WSGateway;

namespace ProjectTest
{
    class Class1
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main(string[] args)
        {
            Gateway gw = new Gateway();

            AnswerRecipient[] answerRecipients = new AnswerRecipient[1];
            answerRecipients[0] = new AnswerRecipient();
            answerRecipients[0].Type = AnswerRecipientType.Email;
            answerRecipients[0].Parameter = "me@mydom.com";

            GetUserStatusAnswer answer = gw.GetUserStatus1("mioNomeUtente", "miaPassword",
            answerRecipients);

            Console.WriteLine("{0}", answer.User);

            foreach (ProgettoSMS.CustomerCredit cc in answer.Credits)
                Console.WriteLine("{0}: {1}", cc.Country, cc.Credits);

            Console.ReadLine();
        }
    }
}
```

## GetIncomingMessages

Consente di recuperare le informazioni relativamente agli sms ricevuti sulla piattaforma.

Il comando può essere invocato mediante le seguenti chiamate:

```
GetIncomingMessagesAnswer GetIncomingMessages(string userName, string password, DateTime?
startDate, DateTime? endDate, string sender)
```

```
GetIncomingMessagesAnswer GetIncomingMessages1(string userName, string password,
DateTime? startDate, DateTime? endDate, string sender, AnswerRecipient[]
answerRecipients)
```

## Esempi GetIncomingMessages

### Esempio 1:

Richiesta degli sms ricevuti dalla data 03/01/2011 ad oggi per il mittente +393334578123

```
Using ProjectTest.WSGateway;
```

```
namespace ProjectTest
{
    class Class1
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main(string[] args)
        {
            Gateway gw = new Gateway();

            AnswerRecipient[] answerRecipients = new AnswerRecipient[1];
            answerRecipients[0] = new AnswerRecipient();
            answerRecipients[0].Type = AnswerRecipientType.Email;
            answerRecipients[0].Parameter = "me@mydom.com";

            GetIncomingMessagesAnswer answer = gw.GetIncomingMessages1("mioNomeUtente",
            "miaPassword", new DateTime(2011, 01, 03), null, "+393334578123",
            answerRecipients);

            Console.WriteLine("{0}", answer.User);

            foreach (ProgettoSMS.IncomingMessage im in answer.Messages)
                Console.WriteLine("{0}: {1} - {2}", im.Snr, im.receivedDate.ToString(),
            im.Text);

            Console.ReadLine();
        }
    }
}
```

## ***Appendice A: possibili stati dei messaggi***

<b>Codice</b>	<b>Descrizione</b>	<b>Note</b>
N	Messaggio nuovo	Il messaggio è stato generato, ma non ancora inviato
S	Messaggio inviato	Il messaggio è stato generato e correttamente inviato
E	Errore in invio	Il messaggio è stato generato, ma durante l'invio si è verificato un errore. Non sarà più inviato
A	Messaggio abortito (di sistema)	Il messaggio è stato generato, ma successivamente ne è stato abortito l'invio. Non sarà più inviato

## ***Appendice B: possibili stati di consegna dei messaggi***

<b>Codice</b>	<b>Descrizione</b>	<b>Note</b>
U	Stato sconosciuto	Il messaggio è stato consegnato al gateway inviante, ma il suo stato non è ancora conosciuto
F	Invio fallito	Non è stato possibile inviare il messaggio
D	Consegnato	Il messaggio è stato consegnato al terminale finale
W	In attesa	Il messaggio è in attesa di essere consegnato al terminale finale

## Appendice C: codici di errore

Codice	Descrizione	Note
0	None	Quando specificato in un qualche messaggio indica che l'operazione è andata a buon fine
100	Syntax error	La request contiene degli errori
110	Parameter syntax error	Un parametro specificato nella request contiene degli errori
112	Parameter value not valid	Il valore assegnato al parametro non è valido
114	Missing parameter	Nella request manca un parametro
120	Missing statement	La request è stata inoltrata senza il parametro statement
125	Statement not recognized	Lo statement indicato nella request non è stato riconosciuto
130	Error in fixed list	Errore in un parametro a lista in cui il valore ha un numero fisso di elementi
135	Error in list	Errore in un parametro a lista
200	Disabled function for reseller	La funzionalità NON risulta essere abilitata per gli utenti reseller
300	Login error	Lo user o la password specificati sono errati
400	Insufficient credit	Il credito del cliente è terminato
410	Insufficient credit	Il credito del reseller è terminato
500	Message not found	Il messaggio non è stato trovato (ad esempio l'ID del messaggio non è corretto)
510	Message not found	Il messaggio non è di proprietà dell'utente che ne ha richiesto lo stato.
999	Internal error	Solitamente nella descrizione è possibile trovare qualche dettaglio ulteriore sull'errore.