Specifiche API

Provisioning Cliente

Indice

1 Introduzione Tecnica

Le API sono accessibili tramite HTTPS con i protocolli:

– HTTP-GET/POST
– XML-RPC
– SOAP

L'encoding di base per l'accesso alla piattaforma è ISO-8859-1 (Latin1), tutti i caratteri vanno quindi indicati con questo charset e anche nell'eventuale encoding in base64 vanno usati caratteri Latin1.

Tutti i nomi dei campi e variabili indicati nel presente documento sono "case sensitive".

1.1 API HTTP GET/POST

Le API HTTP-GET/POST permettono di effettuare richieste in maniera molto semplice fornendo parametri standard alla richiesta HTTP. Per tutte le chiamate è indifferente utilizzare il metodo POST o GET, si raccomanda di eseguire la codifica in URL prima di passare i valori alle API in modo che vengano interpretati bene dai nostri sistemi.

Nel caso di errori evidenziati nelle varie fasi di richiesta, il sistema restituisce un codice di errore e una descrizione.

(vedi "Codici di Errore", paragrafo 10).

Gli errori vengono ritornati con una stringa, a titolo esemplificativo:

-105-sender contains invalid characters or is too long

Per tutte le chiamate HTTP GET/POST è necessario specificare il login e la password di accesso, i nomi dei campi per l'autenticazione sono riportati nella Tabella 1.

Le URL di accesso alle API HTTP GET/POST sono specificate nei paragrafi successivi.

1.2 API XML-RPC

Ogni metodo XML-RPC accetta un solo parametro di tipo *struct* contenente l'elenco dei parametri accettati da ogni singola funzione specifica. Quindi per chiamare un qualsiasi metodo qui documentato vanno specificati i parametri di autenticazione e quelli specifici di ogni metodo.

L'autenticazione viene eseguita con i seguenti parametri obbligatori:

Nome Campo	Descrizione	XML Type
`authlogin`	Login di autenticazione alle API che nel livello di accesso corrente (BCP) corrisponde al login dell'Account Cliente	string
`authpasswd`	Password di autenticazione alle API che, come sopra, corrisponde alla password dell'Account Cliente	string

Tabella 1: Autenticazione

In caso di errore fatale l'API ritorna un errore XML tramite un oggetto ***Fault***, con codice di errore e descrizione del relativo codice di errore (vedi "Codici di Errore", paragrafo 10).

L'URL completo della chiamata XML-RPC è:

https://secure.apisms.it/xmlrpc/BCP/provisioning.py

1.3 API SOAP

Le API SOAP rispettano lo standard WSDL e quindi è possibile implementare il WebService nella propria applicazione anche tramite tool automatici come Microsoft Visual Studio.

Il nome dei campi delle richieste è il medesimo delle API XML-RPC, quindi per la documentazione dei rispettivi campi si rimanda ai paragrafi delle API XML-RPC.

L'URL completo della chiamata SOAP è:

https://secure.apisms.it/soap/web_service

2 Verifica credito residuo

La funzione ritorna il credito residuo del cliente.

API disponibile in : ***HTTP-POST, XML-RPC, SOAP***

2.1 HTTP-GET/POST

L'URL base di accesso è:

https://secure.apisms.it/http/get_credit

Nessun parametro aggiuntivo da specificare, vanno specificati solo i parametri di autenticazione (Tabella 1)
A questa richiesta viene ritornato il valore del credito residuo.

2.2 XML-RPC

Metodo: ***get_credit***
Return: il credito residuo (*long*)
Nessun parametro aggiuntivo da specificare, vanno specificati solo i parametri di autenticazione (Tabella 1)

2.3 SOAP

Metodo: ***getCredit***

Nessun parametro aggiuntivo da specificare, vanno specificati solo i parametri di autenticazione (Tabella 1)
Per maggiori dettagli tecnici vedi WSDL.

3 Invio SMS

Questa funzione permette di eseguire una richiesta di invio SMS.

API disponibile in : ***HTTP-POST, XML-RPC, SOAP***

3.1 HTTP GET/POST

L'URL base di accesso è:

https://secure.apisms.it/http/send_sms

La chiamata ha gli stessi parametri della richiesta XML-RPC, vedi Tabella 2. Con questa API è possibile inviare un solo SMS per ogni richiesta HTTP. Per ogni SMS viene ritornato il relativo codice di invio (ID Messaggio).

A titolo esemplicativo, il risultato positivo dopo l'invio di un SMS può essere:

+01 SMS Queued - ID: 180507124540133599

3.2 XML-RPC

Metodo: ***send_sms***

La richiesta di invio SMS va specificata previa autenticazione in una struct con chiave "sms" con valore un altro array.
Una singola richiesta può contenere quindi più SMS.

Nome Campo	Descrizione	XML Type
`sender`	Sender/mittente dell'SMS **campo da codificare in base64**	string
`body`	Body/Testo dell'SMS **campo da codificare in base64**	string
`destination`	MSISDN / Numero di telefono del destinatario a cui inviare l'SMS	string
`id_api`	ID_API da utilizzare per spedire l'SMS.	integer
`deliver_timestamp`	*Timestamp/Data Ora in cui va programmato l'invio del sms*	*string*
`report_type`	*Tipologia di report da avere per il messaggio*	*string*

Tabella 2: Campi Invio SMS. I campi in corsivo sono opzionali

Ritorno XML-RPC

La chiamata restituirà una lista di struct contenente tante struct ordinate per quante richieste SMS sono state fatte.

Se per esempio sono stati specificati due SMS nella stessa richiesta, e il secondo contiene un errore, si avrà una struct di ritorno simile a questa:

{ 'id_sms' : '18012549746465' } # struct di ritorno relativo al primo SMS

{'error_code' : 100, 'error_message' : 'Invalid destination' } # struct di ritorno relativo al secondo SMS

3.3 SOAP

Metodo: ***sendSms***

Questo metodo accetta come parametri gli stessi della chiamata XML-RPC (vedi Tabella 2). Anche con questa API è possibile inviare più SMS nella stessa richiesta SOAP.

La chiamata ritornerà un oggetto di risposta contenente l'esito dell'accodamento.

Per maggiori dettagli tecnici vedi WSDL.

3.4 Specifiche parametri di Invio

Di seguito verranno definite le specifiche dei parametri indicati nella Tabella 2.

3.4.1 Parametro Sender

Il campo "**sender**" rappresenta il mittente con cui l'SMS sarà spedito. Questo può essere Numerico o Alfanumerico.
Nel caso di "**sender**" numerico, il valore deve essere composto dal numero di telefono comprensivo di "+" e prefisso internazionale, per esempio: +393351234567. In questo caso il formato ammesso sarà una stringa composta dal "+" iniziale seguito da un massimo di 20 cifre.
Nel caso di mittente Alfanumerico, il valore deve essere alfanumerico (Vedi "Codifica dei Messaggi", paragrafo 4) e deve misurare al massimo 11 caratteri. Si sconsiglia comunque l'uso di caratteri speciali nel Sender in quanto non tutti i terminali GSM sono in grado di interpretarli correttamente.
Il valore va codificato in base64 prima di essere inserito nella richiesta.

3.4.2 Parametro Body

Nel campo "**body**" va indicato il testo dell'SMS da spedire.
Il testo deve rispettare quanto specificato nel paragrafo "Codifica dei Messaggi" (paragrafo 4) e non deve superare i 612 caratteri. A seconda del tipo di SMS o di cellulare, l'SMS può arrivare diviso in più parti o visualizzato concatenato in un singolo messaggio.
Il valore va codificato in base64 prima di essere inserito nella richiesta.

3.4.3 Parametro Destination

Nel campo "**destination**" va indicato il numero di cellulare a cui spedire l'SMS.
Il numero va inserito nel formato internazionale ***senza*** il più e con il prefisso internazionale

Se per esempio l'SMS è da spedire in Italia al numero 335 1234567, andrà inserito: "393351234567".

3.4.4 Parametro ID_API

L'ID_API è il numero identificativo della rotta/tipo sms con cui inviare l'SMS.
Questo codice si può trovare nell'area riservata della piattaforma cliccando sul nome del listino nella parte bassa dell'home page.

3.4.5 Parametro Deliver_Timestamp

Il campo "deliver_timestamp" è opzionale e va indicato nel caso si desideri programmare la spedizione di un SMS.
Il formato del valore da inserire è: "GG/MM/AAAA HH:MM:SS", quindi se si vuole spedire un sms il giorno 05/06/2015 alle ore 16:23, bisognerà indicare nel campo: "05/06/2015 16:23:00".

3.4.6 Parametro Report Type

Il campo "report_type" è opzionale e indica il modo con cui si desidera accedere alla reportistica.
I valori ammessi sono:

"P" : Modalità Pull Report
"C" : Modalità Call Back Report

Le modalità sono descritte nel paragrafo "Reportistica" (paragrafo5).

4 Codifica dei messaggi

Le transazioni HTTP saranno negoziate nel charset ISO-8859-1 (Latin 1), tutte le comunicazioni dovranno rispettare questo charset.

Il Body del messaggio SMS deve rispettare il seguente documento di specifica.

Il carattere Euro (€) in quanto non rappresentabile in ISO-8859-1 sarà rappresentato secondo lo standard Windows-1252 ovvero 0x80.

Sia nel Sender che nel Body viene effettuato un controllo di validazione dei caratteri in quanto devono essere presenti nel charset GSM 03.38 (vedi tabella a seguito).

Nella conversione da ISO-8859-1 sono presenti alcuni caratteri, come appunto l'Euro e le parentesi graffe, che in GSM occupano la dimensione di due caratteri. Questi caratteri che valgono "doppio" sono evidenziati in blu nella Tabella 3.

I caratteri che non hanno corrispondenza nella colonna 'ISO-8859-1 DEC', nella seguente tabella saranno considerati non validi.

Hex	Dec	Character name	Character	ISO-8859-1 DEC
0x00	0	COMMERCIAL AT	@	64
0x01	1	POUND SIGN	£	163
0x02	2	DOLLAR SIGN	$	36
0x03	3	YEN SIGN	¥	165
0x04	4	LATIN SMALL LETTER E WITH GRAVE	è	232
0x05	5	LATIN SMALL LETTER E WITH ACUTE	é	233
0x06	6	LATIN SMALL LETTER U WITH GRAVE	ù	249
0x07	7	LATIN SMALL LETTER I WITH GRAVE	ì	236
0x08	8	LATIN SMALL LETTER O WITH GRAVE	ò	242
0x09	9	LATIN CAPITAL LETTER C WITH CEDILLA	Ç	199
0x0A	10	LINE FEED		10
0x0B	11	LATIN CAPITAL LETTER O WITH STROKE	Ø	216
0x0C	12	LATIN SMALL LETTER O WITH STROKE	ø	248
0x0D	13	CARRIAGE RETURN		13
0x0E	14	LATIN CAPITAL LETTER A WITH RING ABOVE	Å	197
0x0F	15	LATIN SMALL LETTER A WITH RING ABOVE	å	229
0x10	16	GREEK CAPITAL LETTER DELTA	Δ	
0x11	17	LOW LINE	_	95
0x12	18	GREEK CAPITAL LETTER PHI	Φ	
0x13	19	GREEK CAPITAL LETTER GAMMA	Γ	
0x14	20	GREEK CAPITAL LETTER LAMBDA	Λ	
0x15	21	GREEK CAPITAL LETTER OMEGA	□	
0x16	22	GREEK CAPITAL LETTER PI	Π	
0x17	23	GREEK CAPITAL LETTER PSI	Ψ	
0x18	24	GREEK CAPITAL LETTER SIGMA	Σ	
0x19	25	GREEK CAPITAL LETTER THETA	Θ	
0x1A	26	GREEK CAPITAL LETTER XI	Ξ	
0x1B	27	ESCAPE TO EXTENSION TABLE		
0x1B0A	27 10	FORM FEED		12
0x1B14	27 20	CIRCUMFLEX ACCENT	^	94
0x1B28	27 40	LEFT CURLY BRACKET	{	123
0x1B29	27 41	RIGHT CURLY BRACKET	}	125
0x1B2F	27 47	REVERSE SOLIDUS (BACKSLASH)	\	92
0x1B3C	27 60	LEFT SQUARE BRACKET	[	91
0x1B3D	27 61	TILDE	~	126
0x1B3E	27 62	RIGHT SQUARE BRACKET	]	93
0x1B40	27 64	VERTICAL BAR	|	124
0x1B65	27 101	EURO SIGN	€	128 *(Windows-1252)*

0x1C	28	LATIN CAPITAL LETTER AE	Æ	198
0x1D	29	LATIN SMALL LETTER AE	æ	230
0x1E	30	LATIN SMALL LETTER SHARP S (German)	ß	223
0x1F	31	LATIN CAPITAL LETTER E WITH ACUTE	É	201
0x20	32	SPACE		32
0x21	33	EXCLAMATION MARK	!	33
0x22	34	QUOTATION MARK	"	34
0x23	35	NUMBER SIGN	#	35
0x25	37	PERCENT SIGN	%	37
0x26	38	AMPERSAND	&	38
0x27	39	APOSTROPHE	'	39
0x28	40	LEFT PARENTHESIS	(	40
0x29	41	RIGHT PARENTHESIS	)	41
0x2A	42	ASTERISK	*	42
0x2B	43	PLUS SIGN	+	43
0x2C	44	COMMA	,	44
0x2D	45	HYPHEN-MINUS	-	45
0x2E	46	FULL STOP	.	46
0x2F	47	SOLIDUS (SLASH)	/	47
0x30	48	DIGIT ZERO	0	48
0x31	49	DIGIT ONE	1	49
0x32	50	DIGIT TWO	2	50
0x33	51	DIGIT THREE	3	51
0x34	52	DIGIT FOUR	4	52
0x35	53	DIGIT FIVE	5	53
0x36	54	DIGIT SIX	6	54
0x37	55	DIGIT SEVEN	7	55
0x38	56	DIGIT EIGHT	8	56
0x39	57	DIGIT NINE	9	57
0x3A	58	COLON	:	58
0x3B	59	SEMICOLON	;	59
0x3C	60	LESS-THAN SIGN	<	60
0x3D	61	EQUALS SIGN	=	61
0x3E	62	GREATER-THAN SIGN	>	62
0x3F	63	QUESTION MARK	?	63
0x40	64	INVERTED EXCLAMATION MARK	¡	161
0x41	65	LATIN CAPITAL LETTER A	A	65
0x42	66	LATIN CAPITAL LETTER B	B	66
0x43	67	LATIN CAPITAL LETTER C	C	67
0x44	68	LATIN CAPITAL LETTER D	D	68
0x45	69	LATIN CAPITAL LETTER E	E	69
0x46	70	LATIN CAPITAL LETTER F	F	70
0x47	71	LATIN CAPITAL LETTER G	G	71
0x48	72	LATIN CAPITAL LETTER H	H	72
0x49	73	LATIN CAPITAL LETTER I	I	73
0x4A	74	LATIN CAPITAL LETTER J	J	74
0x4B	75	LATIN CAPITAL LETTER K	K	75
0x4C	76	LATIN CAPITAL LETTER L	L	76
0x4D	77	LATIN CAPITAL LETTER M	M	77
0x4E	78	LATIN CAPITAL LETTER N	N	78

0x4F	79	LATIN CAPITAL LETTER O	O	79
0x50	80	LATIN CAPITAL LETTER P	P	80
0x51	81	LATIN CAPITAL LETTER Q	Q	81
0x52	82	LATIN CAPITAL LETTER R	R	82
0x53	83	LATIN CAPITAL LETTER S	S	83
0x54	84	LATIN CAPITAL LETTER T	T	84
0x55	85	LATIN CAPITAL LETTER U	U	85
0x56	86	LATIN CAPITAL LETTER V	V	86
0x57	87	LATIN CAPITAL LETTER W	W	87
0x58	88	LATIN CAPITAL LETTER X	X	88
0x59	89	LATIN CAPITAL LETTER Y	Y	89
0x5A	90	LATIN CAPITAL LETTER Z	Z	90
0x5B	91	LATIN CAPITAL LETTER A WITH DIAERESIS	Ä	196
0x5C	92	LATIN CAPITAL LETTER O WITH DIAERESIS	Ö	214
0x5D	93	LATIN CAPITAL LETTER N WITH TILDE	Ñ	209
0x5E	94	LATIN CAPITAL LETTER U WITH DIAERESIS	Ü	220
0x5F	95	SECTION SIGN	§	167
0x60	96	INVERTED QUESTION MARK	¿	191
0x61	97	LATIN SMALL LETTER A	a	97
0x62	98	LATIN SMALL LETTER B	b	98
0x63	99	LATIN SMALL LETTER C	c	99
0x64	100	LATIN SMALL LETTER D	d	100
0x65	101	LATIN SMALL LETTER E	e	101
0x66	102	LATIN SMALL LETTER F	f	102
0x67	103	LATIN SMALL LETTER G	g	103
0x68	104	LATIN SMALL LETTER H	h	104
0x69	105	LATIN SMALL LETTER I	i	105
0x6A	106	LATIN SMALL LETTER J	j	106
0x6B	107	LATIN SMALL LETTER K	k	107
0x6C	108	LATIN SMALL LETTER L	l	108
0x6D	109	LATIN SMALL LETTER M	m	109
0x6E	110	LATIN SMALL LETTER N	n	110
0x6F	111	LATIN SMALL LETTER O	o	111
0x70	112	LATIN SMALL LETTER P	p	112
0x71	113	LATIN SMALL LETTER Q	q	113
0x72	114	LATIN SMALL LETTER R	r	114
0x73	115	LATIN SMALL LETTER S	s	115
0x74	116	LATIN SMALL LETTER T	t	116
0x75	117	LATIN SMALL LETTER U	u	117
0x76	118	LATIN SMALL LETTER V	v	118
0x77	119	LATIN SMALL LETTER W	w	119
0x78	120	LATIN SMALL LETTER X	x	120
0x79	121	LATIN SMALL LETTER Y	y	121
0x7A	122	LATIN SMALL LETTER Z	z	122
0x7B	123	LATIN SMALL LETTER A WITH DIAERESIS	ä	228
0x7C	124	LATIN SMALL LETTER O WITH DIAERESIS	ö	246
0x7D	125	LATIN SMALL LETTER N WITH TILDE	ñ	241
0x7E	126	LATIN SMALL LETTER U WITH DIAERESIS	ü	252
0x7F	127	LATIN SMALL LETTER A WITH GRAVE	à	224

Tabella 3: Caratteri GSM

5 Reportistica

È possibile accedere tramite API alla reportistica degli SMS inviati in due modalità:

– Pull (disponibile in HTTP POST, XML-RPC e SOAP)
– Call Back (API HTTP POST)

La scelta della modalità per accedere al Report va specificata per ogni singola richiesta di invio SMS
(Vedi "Invio SMS" parametro report_type, paragrafo 3.4.6).

Nella modalità "**Pull**", sarà una procedura del Cliente ad eseguire un'interrogazione periodica alle nostre API per verificare l'esistenza e la disponibilità di nuovi Report, scaricandoli.
Una volta scaricati i report saranno cancellati dal nostro database e quindi alla successiva interrogazione non saranno scaricati nuovamente.

Nella modalità "**Call Back**" è invece una nostra procedura che invia i Report eseguendo chiamate HTTP/HTTPS ad un URL fornito dal cliente appena questi sono disponibili sui nostri sistemi.

Per utilizzare la modalità Call Back Report è necessario specificare una URL nell'area riservata "Extra → API In HTTP e XML SOAP" su cui verranno eseguite le POST.

La chiamata POST di Call Back avverrà con i seguenti parametri:

id_sms : ID SMS, codice dell'invio rilasciato in fase di invio

id_dlr : codice della parte del mesaggio, per l'invio via API sarà sempre valorizzato a 1

destination: il numero a cui è stato spedito l'SMS

timestamp_report: timestamp del report nel formato GG/MM/AAAA HH:MM:SS

status: stato del messaggio, (S → spedito, R → ricevuto, K → numero non esistente, rifiutato dalla rete o scaduto)

5.1 Pull Report HTTP GET/POST

La richiesta Pull Report ha come scopo quello di "scaricare" tutti i report generati per gli SMS precedentemente inviati con modalità "Pull Report".
L'URL base di accesso è:

https://secure.apisms.it/http/pull_report

Gli unici parametri richiesti sono l'autenticazione con i parametri a Tabella 1.

La richiesta ritorna un CSV composto dai seguenti elementi:
- id_dlr : codice della parte del mesaggio, per l'invio via API sarà sempre valorizzato a 1
- id_sms : ID SMS, codice dell'invio rilasciato in fase di invio
- destination: il numero a cui è stato spedito l'SMS
- timestamp_report: timestamp del report nel formato GG/MM/AAAA HH:MM:SS
- status: stato del messaggio, (S → spedito, R → ricevuto, K → numero inesistente, rifiutato dalla rete o scaduto)

I nomi dei campi sono "case sensitive".

5.2 Pull Report XML-RPC

La richiesta Pull Report ha come scopo quello di "scaricare" tutti i report generati per gli SMS precedentemente inviati con modalità "Pull Report".
Gli unici parametri richiesti sono l'autenticazione con i parametri della Tabella 1.

Metodo: ***pull_report***
Return: struct

La richiesta ritorna una struct composta dai seguenti elementi:
- id_sms : ID SMS, codice dell'invio rilasciato in fase di invio
- id_dlr : codice della parte del mesaggio, per l'invio via API sarà sempre valorizzato a 1
- destination: MSISDN del numero a cui è stato spedito l'SMS
- timestamp_report: timestamp del report nel formato GG/MM/AAAA HH:MM:SS
- status: stato del messaggio, (S → spedito, R → ricevuto, K → numero inesistente, rifiutato dalla rete o scaduto)

I nomi dei campi sono "case sensitive".

5.3 Pull Report SOAP

Metodo: ***pullReport***

La richiesta Pull Report ha come scopo quello di "scaricare" tutti i report generati per gli SMS precedentemente inviati con modalità "Pull Report".
Gli unici parametri richiesti sono l'autenticazione con i parametri a Tabella 1. La chiamata ritorna un oggetto contenente gli stessi campi ritornati dal metodo XML-RPC (paragrafo 5.2).
Per maggiori dettagli tecnici vedi WSDL.

6 Ricezione SMS

Il servizio Ricezione SMS è un servizio attivabile su richiesta che permette la ricezione di SMS via E-Mail o via API HTTP/HTTPS.

Al cliente viene fornito una numerazione GSM e tutti gli SMS ricevuti su questo numero verranno inoltrati via E-Mail o all'URL indicata dal cliente in fase di attivazione del servizio.

Nel caso di inoltro via API HTTP/HTTPS il cliente deve predisporre un'URL dove riceve per ogni SMS ricevuto una POST con i seguenti parametri:

Sender : numero di telefono del cellulare che ha spedito l'SMS

Destination : numero di telefono del cellulare assegnato al servizio SMS Ricezione

Timestamp : timestamp del messaggio nel formato GG/MM/AAAA HH:MM:SS

Body : testo dell'SMS ricevuto

7 Aggiunta Email Autorizzata per Invio SMS

La richiesta Create Email Out Authorized permette di aggiungere un Email autorizzata a spedire SMS. Con questa chiamata è possibile associare all'email il mittente, la relativa rotta con cui inviare l'SMS (ID_API) e se può accettare i messaggi long o meno.

API disponibile in : ***XML-RPC***

Metodo: ***create_sms_email_out_authorized***

Nel caso venga specificato il campo **sender**, e quel sender non sia già stato inserito, verrà automaticamente inserito e sarà visibile nell'area "Extra → Gestione Mittenti". Questo sarà quindi disponibile anche per eseguire invii via Web.
Return: ritorna "OK" in caso di esito positivo, oggetto Fault nel caso di errori.

I parametri accettati da questo metodo sono:

Nome Campo	Descrizione	XML Type
id_api	ID_API da utilizzare per spedire l'SMS.	string
email	Indirizzo email da cui sarà possibile spedire SMS.	string
sender	*Mittente con cui l'SMS sarà spedito (vedi specifiche Sender par. 3.4.1)* ***campo da codificare in base64***	*string*
sender_description	*Descrizione mnemonica nel caso di mittente numerico (che compare su piattaforma via web)* ***campo da codificare in base64***	*string*
sms_long	*Valore che indica se può essere spedito un messaggio più lungo di 160 caratteri. Nel caso sms_long sia True, il messagggio viene accettato e gestito, in caso contrario se il messsaggio inoltrato via email è più lungo di 160 caratteri verrà rifiutato.*	*boolean*

I campi in corsivo sono opzionali

8 Gestione Rubrica / Liste

La gestione della Rubrica e delle Liste è possibile sia sulla piattaforma stessa che via API, è quindi possibile mantenere sincronizzati il database sulla piattaforma e altro database gestionale.

API disponibile in : ***XML-RPC***

Metodo: ***contacts_management***

I parametri accettati da questo metodo sono:

Nome Campo	Descrizione	XML Type
contacts	*La struttura contiene i contatti su cui fare l'elaborazione richiesta*	*struct*
id_lista	*Lista/Gruppo su cui eseguire l'elaborazione richiesta*	*string*
action	*Azione richiesta da eseguire*	*string*

I campi id_lista e contacts sono opzionali a seconda del tipo di "action" che viene richiesta.

Il parametro *contacts* è una struct contenente array associativi contenenti a loro volta i contatti da fornire al metodo per effettuare l'elaborazione richiesta specificata dall'*action*.
I campi degli array della struttura *contacts* sono:

Nome Campo	Descrizione	XML Type
nome	*Nome del contatto (testo variabile)* ***campo da codificare in base64***	*string*
cognome	*Cognome del contatto (testo variabile)* ***campo da codificare in base64***	*string*
cellulare	Numero di cellulare del contatto, chiave primaria del contatto. (Per il formato vedi "Invio SMS" parametro destination, paragrafo 3.4.3).	string
sesso	*Sesso del contatto, valori ammessi "M" o "F"*	*string*
indirizzo	*Indirizzo del contatto (testo variabile)* ***campo da codificare in base64***	*string*
citta	*Città del contatto (testo variabile)* ***campo da codificare in base64***	*string*
provincia	*Sigla di due lettere della provincia italiana, Es. "BO" per Bologna*	*string*
provincia_testo	*Nome della provincia nel caso di provincia estera.* ***campo da codificare in base64***	*string*
datanascita	*Data di nascita nel formato DD/MM/YYYY (es. 17/05/1976)*	*string*
telefono	*Numero di telefono del contatto, valore libero* ***campo da codificare in base64***	*string*
email	*Email del contatto* ***campo da codificare in base64***	*string*
note	*Eventuali note sul contatto* ***campo da codificare in base64***	*string*
smscompleanno	*Attiva o Disattiva la funzione di invio dell'SMS di Compleanno*	*boolean*

Tutti i campi sono facoltativi eccetto il campo "cellulare".

Il parametro *action* definisce l'azione da compiere sui contatti contenuti nel parametro *contacts* e/o sulla lista definita dal parametro *id_lista*.

Comportamento del metodo in funzione dei parametri forniti:

Valore del parametro action	id_lista valorizzato	Comportamento
INSUPD	no	Inserisce o aggiorna in rubrica i contatti contenuti nel parametro *contacts*.
	sì	Inserisce o aggiorna in rubrica i contatti contenuti nel parametro *contacts* e li associa alla lista specificata.
DELETE	no	Elimina dalla rubrica i contatti contenuti nel parametro *contacts*.

	sì	Disassocia i contatti contenuti nel parametro *contacts* dalla lista specificata e non li elimina dalla rubrica.
`DELETE ALL`	no	Elimina dalla rubrica tutti i contatti.
	sì	Disassocia tutti i contatti della lista dalla lista stessa e non li elimina dalla rubrica.
`EXPORT`	no	Ritorna le informazioni su tutti i contatti della rubrica.
	sì	Ritorna le informazioni sui contatti contenuti nella lista specificata.

Ritorno XML-RPC

Valore del parametro action	Ritorno
`INSUPD`	Ritorna una struct come segue: {'added_contacts_count' : integer, 'updated_contacts_count' : integer, 'invalid_contacts_count' : integer, 'invalid_contacts' : [“393351234567”, ”393357654321”, ...] *'list_added_contacts_count' : integer*} added_contacts_count è il numero di contatti aggiunti, updated_contacts_count è il numero di contatti aggiornati, invalid_contacts_count è il numero di contatti non validi forniti al metodo (zero nel caso in cui siano tutti corretti), invalid_contacts è l'array contenente i numeri di cellullare dei contatti non validi, list_added_contacts_count è il numero dei contatti aggiunti alla lista (se nella chiamata al metodo id_lista era valorizzato).
`DELETE`	OK
`DELETE ALL`	OK
`EXPORT`	Ritorna una lista di array associativi così composti: {'Nome' : string, 'Cognome' : string, 'Cellulare' : string, 'Sesso' : string, 'Indirizzo' : string, 'Citta' : string, 'Provincia' : string, 'DataNascita' : string, 'Telefono' : string, 'Email' : string, 'Note' : string, 'SMSCompleanno' : boolean}

Metodo: ***get_liste***

Elenco delle liste. Nessun parametro aggiuntivo richiesto.

Return: un lista di array contenenti la seguente struttura:

Ritorno XML-RPC

Chiave	Descrizione	XML Type
nomelista	Nome della lista.	string
id	Identificativo della lista	integer
attributo		string
iscrizione_via_web		boolean

Metodo: ***add_lista***

Crea una lista.

Parametri:

Nome Campo	Descrizione	XML Type
nomelista	Nome della lista.	string

Return: l'identificativo della lista creata (integer).

Metodo: ***change_lista***

Cambia il nome di una lista.

Parametri:

Nome Campo	Descrizione	XML Type
id	Identificativo della lista.	integer
nomelista	Nuovo nome della lista.	string

Return: il nome precedente della lista (string) .

Metodo: ***search_contact_lista***

Cerca in quali liste è prensente un contatto della Rubrica.

Parametri:

Nome Campo	Descrizione	XML Type
msisdn	Numero di cellulare del contatto in rubrica	string

Return: una lista di identificativi (integer) di liste in cui è presente il contatto, se il contatto non viene trovato o non è presente in nessuna lista, ritorna una lista vuota.

Metodo: ***del_lista***

Rimuove una lista, la lista deve essere vuota.

Parametri:

Nome Campo	Descrizione	XML Type
`id_lista`	Identificativo della lista da eliminare.	

Return: (boolean) True in caso di successo, False altrimenti.

9 Verifica Numeri

Il servizio di Verifica Numeri permette di avere un riscontro circa la validità di un numero di telefonia mobile, e di conoscere l'effettivo operatore di appartenenza anche se è avvenuta una Number Portability.
La verifica è possibile grazie all'interrogazione di sistemi della rete GSM interconnessa con tutti gli operatori.

La verifica è disponibile in due modalità:
– Network Query
Questa verifica viene eseguita tramite interrogazione HLR, un sistema di rete degli operatori mobili aggiornato in tempo reale in grado di fornire sia la validità o meno del numero di telefono sia il reale operatore su cui è registrata la SIM.
– National Operator Query
Questa verifica è in grado di rilevare solo l'operatore di appartenenza di un numero di telefono mobile e non la relativa validità.
La verifica è eseguita tramite interrogazione dei sistemi di rete interconnessi direttamente alle reti Nazionali degli operatori aggiornate al giorno lavorativo precedente.

9.1 Network Query

API disponibile in : ***XML-RPC***

Metodo: ***network_query***

I parametri accettati da questo metodo sono:

Nome Campo	Descrizione	XML Type
`numbers`	*La struttura contiene una lista di numeri di telefonia mobile, compresi di prefisso internazionale, su cui effettuare la verifica dell'operatore di appartenenza.* *Il numero deve contenere solo caratteri numerici*	*struct*

Ritorno XML-RPC

Il metodo ritorna un valore di tipo *string* contenente un identificativo univoco delle richiesta effettuata, che conferma l'avvenuta ricezione. La richiesta viene elaborata immediatamente, ed il risultato viene consegnato all'URL specificato in fase di attivazione del servizio, attraverso upload di un file **csv** tramite POST HTTP (vedi esempio paragrafo 11.5).

Il file continene i seguenti campi:

Nome Campo	Descrizione
`ID richiesta`	*Identificativo univoco della richiesta inviata ai sistemi tramite API*
`msisdn`	*Numero di telefonia mobile verificato*
`stato`	*Stato della verifica*
`mcc`	*Codice identificativo della nazione di appartenenza dell'operatore*
`mnc`	*Codice identificativo dell'operatore di appartenenza del numero*
`operatore`	*Nome dell'operatore di appartenenza del numero*

Stato può assumere I seguenti valori:

Stato	Descrizione
O	*Il numero è valido*
K	*Il numero non è valido o non esistente*
E	*Errore della rete GSM in fase di verifica del numero*

9.2 National Operator Query

API disponibile in: ***XML-RPC***

Metodo: ***national_operator_query***

I parametri accettati da questo metodo sono:

Nome Campo	Descrizione	XML Type
numbers	*La struttura contiene una lista di numeri di telefonia mobile, comprensivi di prefisso nazionale, su cui effettuare la verifica dell'operatore di appartenenza.* *Il numero deve contenere solo caratteri numerici*	*struct*

Ritorno XML-RPC

Il metodo ritorna una struttura dati *XML* di tipo *struct* contenente i dati dell'elaborazione della richiesta.

Nome Campo	Descrizione
msisdn	*Numero di telefonia mobile verificato*
stato	*Stato della verifica*
mcc	*Codice identificativo della nazione di appartenenza dell'operatore*
mnc	*Codice identificativo dell'operatore di appartenenza del numero*
operatore	*Nome dell'operatore di appartenenza del numero*

Stato può assumere i seguenti valori:

Stato	Descrizione
Valid	*Il numero è valido in seguito a verifiche formali e sintattiche*
Invalid	*Il numero non è valido in seguito a verifiche formali e sintattiche*
Not Covered	*Il numero non appartiene ad una nazione coperta dal servizio*

10 Codici di Errore

Tutti gli oggetti *Fault* restituiti dalle API HTTP-GET/POST, XML-RPC e SOAP sono elencati di seguito:

Error Code	Description	Causa
3	Access denied	La login/password per l'accesso non è corretta o il tipo di risorsa richiesta non è accessibile dall'account con cui si è tentato di eseguire l'accesso.
5	Parameter %s not specified	Il parametro fornito al metodo XML-RPC non è stato specificato
6	Date '%s' incorrect	La data specificata non è valida
12	Invalid id_cliente	Si è tentato di operare su un cliente non esistente o non si dispongono dei permessi necessari per l'operazione
13	%s invalid parameter type	Il tipo del parametro inserito non è corretto. (Es: il codice doveva essere Integer e invece è stato passato come Stringa)
30	method %s not supported	Il metodo XML-RPC chiamato non esiste
31	%s arguments expected	Il numero di argomenti non è conforme alla richiesta.
32	Internal server error, try again later	La richiesta inserita è stata annullata in quanto si sono verificati problemi tecnici, l'errore è tracciato e va verificato con il reparto tecnico.
33	Wrong xml request: %s	La richiesta xml non è valida, viene riportato il messaggio di errore del parser.
34	Malformed request	La richiesta al metodo non era composta da un'array associativo(struct).
100	Invalid destination	La destination specificata non è formalmente valida
101	Destination not allowed	La destination specificata non è consentita per l'ID API specificata o comunque non valida
102	Body contains invalid characters or is too long (max 612 chars)	Il Body specificato è troppo lungo o contiene caratteri non ammessi
103	Not enough credit	Il credito residuo non è sufficiente per consentire l'invio
104	Invalid id_api	L'ID_API specificato non è valido
105	Sender contains invalid characters or is too long (max 11 chars)	Il campo Sender contiene caratteri non validi, o è troppo lungo.
35	email already exist	L'email che si è cercato di inserire è già presente come email autorizzata all'invio.
4	Selected list does not exist	La lista selezionata per la gestione dei contatti non esiste.
16	Unknown action	L'azione richiesta per la gestione dei contatti non esiste.
17	Service not provisioned	Listino non definito per utilizzare i servizi network_query e national_operator_query

11 Esempi

Verranno illustrati degli esempi di chiamata alle funzioni in vari linguaggi di programmazione. Ovviamente in quanto le combinazioni di esempio sarebbero decine e decine abbiamo considerato solo quelle di API/Linguaggio più comunemente utilizzate dai nostri clienti per inviare SMS.

11.1 Esempio PHP XML-RPC

La libreria su cui consigliamo di lavorare in PHP è scaricabile all'indirizzo: http://phpxmlrpc.sourceforge.net/
La versione su cui sono stati eseguiti i test è la 2.2.

L'esempio che segue prende in considerazione l'invio di un SMS

```
<?

include("xmlrpc.inc");
$xmlrpc_client = new xmlrpc_client('https://secure.apisms.it/xmlrpc/BCP/provisioning.py');

# parametri obbligatori (vedi paragrafo 3)
$parametri = array(
   'authlogin' => 'login',
   'authpasswd' => 'password',
   'sms' => array( array( 'sender' => base64_encode(“HELLO”),
                          'body'   => base64_encode(“Test message”),
                          'destination' => '393351234567',
                          'id_api' => 99999 ))
   );
$xmlrpc_msg = new xmlrpcmsg('send_sms', array(php_xmlrpc_encode($parametri)));
$xmlrpc_resp = $xmlrpc_client->send($xmlrpc_msg);
if($xmlrpc_resp->errno != 0) {
  # Qui si può gestire un controllo di errore
   die(sprintf("ERRORE %s",$xmlrpc_resp->errstr));
}
$decode = php_xmlrpc_decode($xmlrpc_resp->value());
# come output la struttura dati ritornata che poi si può integrare nei propri sistemi
print_r($decode);
?>
```

11.2 Esempio PHP HTTP GET/POST

L'esempio che segue prende in considerazione l'invio di un SMS. Questo esempio richiede che sul server sia abilitata la libreria CURL di PHP.

```
<?
  $buffer = array(“authlogin” => “login”,
                  “authpasswd” => “password”,
                  “sender” => base64_encode(“mittente”),
                  “body” => base64_encode(“testo sms”),
                  “destination” => “393351234567”,
                  “id_api” => 999);

  $ch = curl_init();
  curl_setopt($ch, CURLOPT_URL, “https://secure.apisms.it/http/send_sms”);
  curl_setopt($ch, CURLOPT_HEADER, 0);
  curl_setopt($ch, CURLOPT_POSTFIELDS, $buffer);
  curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
  $ret = curl_exec($ch);
  curl_close($ch);
  # ritorno dalle api
  print_r($ret);
?>
```

11.3 Esempio .NET XML-RPC

Di seguito riportiamo il medesimo esempio riportato nel paragrafo precedente in PHP equivalente in VB .NET implementando l'API in XML-RPC

La libreria che consigliamo è scaricabile dall'indirizzo: http://www.xml-rpc.net/

```
Imports CookComputing.XmlRpc
Imports System.Text
Public Class SMSConnector
    Private _authlogin As String = "login"
    Private _authpasswd As String = "password"
    Private _proxyURL As String = "https://secure.apisms.it/xmlrpc/BCP/provisioning.py"
    Private _proxy As ISMSproxy
    Public Sub New()
        _proxy = XmlRpcProxyGen.Create(GetType(ISMSproxy))
    End Sub
    Public Function SendSMS(ByVal sender As String, ByVal body As String, ByVal destination As String, ByVal
id_api As Integer)
        Dim params As New ISMSproxy.sendParam
        params.authlogin = _authlogin
        params.authpasswd = _authpasswd
        Dim sms1 As New ISMSproxy.SMSData
        sms1.sender = convertToBase64String(sender)
        sms1.body = convertToBase64String(body)
        sms1.destination = destination
        sms1.id_api = id_api
        Dim smsCollection As ISMSproxy.SMSData() = {sms1}
        params.sms = smsCollection
        Try
            Return _proxy.send_sms(params)
        Catch ex As Exception
            Throw ex
        End Try
    End Function
    Private Function convertToBase64String(ByVal value As String) As String
        Return System.Convert.ToBase64String(Encoding.Default.GetBytes(value))
    End Function
End Class

<XmlRpcUrl("https://secure.apisms.it/xmlrpc/BCP/provisioning.py")> _
Public Interface ISMSproxy
    <XmlRpcMethod("send_sms")> _
    Function send_sms(ByVal params As sendParam) As Object()
    Class baseParams
        Public authlogin As String
        Public authpasswd As String
    End Class
    Class sendParam
        Inherits baseParams
        Public sms As SMSData()
    End Class
    Structure SMSData
        Public sender As String
        Public body As String
        Public destination As String
        Public id_api As Integer
        'Public deliver_timestamp As String
        ' Public report_type As String
    End Structure
End Interface
```

11.4 Esempio .NET SOAP

Se si usa Microsoft Visual Studio è sufficiente aggiungere nel progetto tramite "Add Web Reference" il web service.

A questo punto nel codice sarà possibile utilizzare, ad esempio, l'api di invio nel seguente modo:

```
        Dim SMSconn As New gwsms()
        Dim request As New Richiesta
        request.authlogin = "login"
        request.authpassd = "password"

        Dim msg1 As New SMS()
        msg1.body = convertToBase64String("Prova di invio SMS" )
        msg1.destination = "393351234567"
        msg1.id_api = 999

        ReDim request.Msg(0)
        request.Msg(0) = msg1


        Dim resp As SMSresp() = SMSconn.sendSms(request)

        For Each r As SMSresp In resp

            Me.TextBox1.Text &= "ID: " & r.ID_Spedizione & "      Codice: " & r.Codice & "      Descr: " &
r.Descrizione & vbCrLf

        Next
        Private Function convertToBase64String(ByVal value As String) As String
            Return System.Convert.ToBase64String(Encoding.Default.GetBytes(value))
        End Function
```

11.5 Esempio ricezione dati da CallBack Network Query

In questo esempio salviamo il contenuto del file ricevuto in upload dalla chiamata di callback del servizio network_query. L'esempio è puramente a scopo dimostrativo in quanto sarà poi a cura del programmatore l'analisi dei dati ricevuti.

In questo esempio i dati ricevuti dalla funzione di callback vengono salvati in un file di log "`/tmp/test_callback_hlr_report.log`".

```
<?
  $flog = fopen("/tmp/test_callback_hlr_report.log", 'a');
  fwrite($flog, sprintf("%s %s\n", date("d/m/Y H:i:s"), print_r($_FILES, true)));
  $fupload = fopen($_FILES['file']['tmp_name'], 'r');
  fwrite($flog, fread($fu, $_FILES['file']['size']));
  fwrite($flog, ">>>>>>> END <<<<<<<\n");
  fclose($flog);
  fclose($fupload);
?>
```

12 Changes

09/05/2008 – Versione 1

19/05/2008 – Aggiunto paragrafo 7

15/07/2008 – Corretto errore sintassi esempio paragrafo 9.1, corretti codici di errore (senza il -) paragrafo 8

13/11/2008 – Aggiunto paragrafo 8 relativo a "Gestione Rubrica / Liste"

04/08/2009 – Aggiunto paragrafo 9 relativo a "Verifica Numeri"

04/06/2013 – Integrato metodi per gestione liste

Classificazione documento

Revisione	R 197 - 04/06/13 17.22
Versione	1

Classificazione di sicurezza

Protocollo trasmissione Porta TCP/IP	HTTPS
	443
Protocollo applicativo	XML-RPC
URL base di accesso	https://secure.apisms.it/
Livello di accesso	BCP – Provisioning Cliente
Procedure autorizzate	XML-RPC: provisioning.py [send_sms(), get_credit(), pull_report(), create_sms_email_authorized(), contacts_management(), network_query(), national_operator_query(), get_liste(), add_lista(), change_lista(), search_contact_lista(), del_lista()] SOAP: sendSms, getCredit, pullReport HTTP-POST/GET: get_credit, send_sms, pull_report