**Integrazione SMS VB.net SDK**

Indice

[1 INTRODUZIONE 4](#__RefHeading__44166_1366233465)

[2 CONFIGURAZIONE DELLA LIBRERIA 4](#__RefHeading__44168_1366233465)

[3 CREAZIONE DI UNA CONNESSIONE 5](#__RefHeading__44170_1366233465)

[4 INVIO DI SMS 6](#__RefHeading__44172_1366233465)

[5 STATO DEI MESSAGGI 7](#__RefHeading__44174_1366233465)

[6 CANCELLAZIONE INVII POSTICIPATI 8](#__RefHeading__44176_1366233465)

[7 STORICO DEI MESSAGGI 8](#__RefHeading__44178_1366233465)

[8 CREDITO SMS DISPONIBILE 9](#__RefHeading__44180_1366233465)

[9 LETTURA DEI MESSAGGI RICEVUTI TRAMITE IL SERVIZIO DI RICEZIONE SMS 10](#__RefHeading__44182_1366233465)

[10 GESTIONE SUBACCOUNTS 11](#__RefHeading__44184_1366233465)

[11 GESTIONE CREDITI SUBACCOUNTS 12](#__RefHeading__44186_1366233465)

# INTRODUZIONE

[**[ INDICE ]**](#INDICE)

I server Aruba mettono a disposizione degli sviluppatori alcune interfacce di comunicazione HTTP per l'invio e la ricezione dei messaggi, la verifica di validità dei numeri di telefono, e permettono inoltre di recuperare tutti i dati riguardanti lo stato dei messaggi e lo storico.

# CONFIGURAZIONE DELLA LIBRERIA

[**[ INDICE ]**](#INDICE)

Nell'archivio si trovano i seguenti file:

* questo documento
* archivio contenente i file sorgente dell'SDK

Per iniziare a lavorare con l'sdk è necessario inserire la propria login e password all'interno della classe Config (file Config.cs)

Public Class Config

 Public Const HOSTNAME As String = "http://adminsms.aruba.it

 Public Const USERNAME As String = "your\_Aruba\_login"

 Public Const PASSWORD As String = "your\_Aruba\_password"

 Public Const DEFAULT\_PORT As Integer = 80

 Public Const PROXY As String = ""

 Public Const PROXY\_PORT As Integer = 8080

End Class

Se si utilizza un proxy, vanno configurate nella stessa classe le voci PROXY e PROXY\_PORT.
Se, diversamente, si utilizzano più account o si utilizzano le funzionalità di subaccount, è possibile specificare login e password all'apertura della connessione (vedi paragrafo successivo).

# CREAZIONE DI UNA CONNESSIONE

[**[ INDICE ]**](#INDICE)

Tutte le operazioni (invio sms, richiesta stato messaggi, ecc.) avvengono attraverso l'interfaccia SMSCConnection. E' possibile ottenere un oggetto SMSCConnection tramite le chiamate:

* new SMSCHTTPConnection()
* new SMSCHTTPConnection(login As String, password as String)
che permette inoltre di specificare uno username e una password specifici per la connessione.

Per chiudere una connessione si utilizza il metodo logout() dell'interfaccia SMSCConnection.

Per quanto non sia necessario, se si utilizza un solo account è consigliabile creare un oggetto SMSConnection all'avvio dell'applicazione e di chiuderlo all'uscita dell'applicazione.

Esempio:

Dim connection As SMSCConnection
Try
 connection = New SMSCHTTPConnection()
Catch smsce As SMSCException
 System.Console.WriteLine("exception: " + smsce.Message)
End Try
' operazioni sulla connessione
' ...
smsc\_connection.logout() 'nel caso di HTTP il metodo non lancia mai eccezioni

# INVIO DI SMS

[**[ INDICE ]**](#INDICE)

La classe SMS contiene tutti i dati necessari per inviare un messaggio SMS attraverso il gateway Aruba.

Creato un oggetto di tipo SMS, è necessario impostare:

* tipo di messaggio: ALTA o (attributo TypeOfSMS);
* destinatari: numero di telefono dei destinatari del messaggio in formato internazionale; per aggiungere un numero alla lista dei destinatari, utilizzare i metodi addSmsRecipient(String) oppure addSmsRecipient(SMSRecipient);
* messaggio: attributo Message, imposta il testo del messaggio da inviare, massimo 160 caratteri per un'unico SMS, massimo 1000 caratteri per un SMS multiplo (concatenato). Attenzione: i caratteri ^ { } \ [ ~ ] | € vengono calcolati come 2 caratteri; è possibile utilizzare l'attributo CountSMS per conoscere il numero di SMS necessari per effettuare l'invio;
* mittente: nel caso di SMS con mittente personalizzato, massimo 16 caratteri numerici per un numero di telefono in formato internazionale oppure 11 caratteri alfanumerici per una stringa di testo (Es: "+123456789012345" oppure "Azienda Spa"), attributo SMSSender .

Opzionalmente, si possono definire:

* data e ora per l'invio di SMS posticipato (attributo ScheduledDelivery);
* una stringa alfanumerica da associare all'invio, indispensabile per ottenere successivamente le informazioni riguardanti lo stato dei messaggi (attributo OrderId). La lunghezza massima è di 32 caratteri, tutti i caratteri eccedenti non saranno considerati.

Se l'OrderId non viene specificato, il server ne genererà uno garantito univoco.

L'OrderId potrà essere utilizzato per richiedere lo stato del messaggio per i 30 giorni successivi all'invio.

Esempio:

try

 Dim sms As New SMS()

 sms.addSMSRecipient("+393471234567")

 sms.addSMSRecipient("+393471234566")

 sms.Message = "hello world!"

 sms.SMSSender = "test"

 sms.setImmediate()

 sms.OrderId = "FFFAAABBBC"
 Dim result as SendResult = smsc\_connection.sendSMS(sms)
Catch smscre As SMSCRemoteException
 System.Console.WriteLine("message: " + smscre.Message)
Catch smsce As SMSCException
 System.Console.WriteLine("exception: " + smsce.Message)
End Try

# STATO DEI MESSAGGI

[**[ INDICE ]**](#INDICE)

Successivamente all'invio dei messaggi è possibile richiederne lo stato, utilizzando il metodo getMessageStatus(String) dell'oggetto SMSCConnection. Specificando l'id dell'SMS, si richiede al server l'esito dell'invio e viene restituita una List<MessageStatus>; ogni oggetto di tipo MessageStatus contiene:

* il numero di telefono del destinatario (attributo Recipient);
* lo stato di invio (attributo Status);
* data e ora di effettiva ricezione del messaggio, se l'sms è stato correttamente ricevuto dal destinatario (attributo DeliveryDate).

L'enum SMSStatus può assumere i seguenti valori:

SCHEDULED // posticipato, non ancora inviato

SENT // inviato, non attende delivery

DLVRD // l'SMS è stato correttamente ricevuto

ERROR // errore in invio dell'SMS

TIMEOUT // l'operatore non ha fornito informazioni sull'SMS entro le 48 ore

TOOM4NUM // troppi SMS per lo stesso destinatario nelle ultime 24 ore

TOOM4USER // troppi SMS inviati dall'utente nelle ultime 24 ore

UNKNPFX // prefisso SMS non valido o sconosciuto

UNKNRCPT // numero di telefono del destinatario non valido o sconosciuto

WAIT4DLVR // messaggio inviato, in attesa di delivery

WAITING // in attesa, non ancora inviato

UNKNOWN // stato sconosciuto

Se non si vuole implementare una gestione particolareggiata degli eventuali errori d'invio, la classe MessageStatus fornisce un attributo bool IsError che vale true se il messaggio è in errore.

Esempio:

try

 Dim smsstatus As List(Of MessageStatus) = connection.getMessageStatus("1234")

 For Each message\_status As MessageStatus In smsstatus

 System.Console.WriteLine("destinatario :" + message\_status.Recipient & ": ")

 If message\_status.IsError Then

 System.Console.WriteLine("errore!")

 Else

 If message\_status.Status = MessageStatus.SMSStatus.DLVRD Then

 System.Console.WriteLine("consegnato.")

 Else

 System.Console.WriteLine("in attesa...")

 End If

 End If

 Next

catch smscre As SMSCRemoteException
 System.Console.WriteLine("message: " + smscre.Message)
End Try

# CANCELLAZIONE INVII POSTICIPATI

[**[ INDICE ]**](#INDICE)

Il metodo removeScheduledSend(String) dell'oggetto SMSCConnection richiede al server Aruba di annullare un invio precedentemente programmato impostando il parametro scheduled\_delivery al momento dell'invio dei messaggi.

Esempio:

try

 connection.removeScheduledSend("12345")

catch smscre As SMSCRemoteException
 System.Console.WriteLine("message: " + smscre.Message)
End Try

# STORICO DEI MESSAGGI

[**[ INDICE ]**](#INDICE)

Il metodo getSMSHistory() dell'oggetto SMSCConnection richiede al server Aruba la lista degli SMS inviati nell'intervallo fra due date specifiche; restituisce una List<SentSMS>.

Esempio:

Try

 Dim sent\_smss As List(Of SentSMS) = connection.getSMSHistory(New DateTime(2009, 1, 1), DateTime.Now)

 For Each sent\_sms As SentSMS In sent\_smss

 System.Console.WriteLine((("SMS " + sent\_sms.OrderId & ", tipo SMS" & sent\_sms.TypeOfSMS.ToString() & " creato il ") + sent\_sms.CreateTime & " inviato a ") + sent\_sms.RecipientsCount & " destinatari")

 Next

Catch smscre As SMSCRemoteException

 System.Console.WriteLine("Exception from Aruba: " & smscre.getMessage())

End Try

# CREDITO SMS DISPONIBILE

[**[ INDICE ]**](#INDICE)

Il metodo getCredits() dell'oggetto SMSCConnection richiede al server Aruba la propria disponibilità di credito; restituisce una List<Credit>. Ogni oggetto Credit contiene: il tipo, la nazione (null se credito internazionale) e la quantità di credito disponibile.

Esempio:

Try

 Dim credits As List(Of Credit) = connection.getCredits()

 For Each credit As Credit In credits

 If credit.Nation IsNot Nothing AndAlso credit.Nation = Nations.ITALY Then

 If credit.CreditType = CreditType.ALTA Then

 System.Console.WriteLine("Posso inviare " + credit.Count & " SMS con mittente personalizzato in Italia.")

 Exit For

 End If

 End If

 Next
Catch smscre As SMSCRemoteException
 System.Console.WriteLine("Exception from Aruba: " & smscre.getMessage())
End Try

# LETTURA DEI MESSAGGI RICEVUTI TRAMITE IL SERVIZIO DI RICEZIONE SMS

[**[ INDICE ]**](#INDICE)

L'utente proprietario di uno o più servizi di ricezione, dedicati o condivisi, che decida di richiedere i messaggi ai server Aruba, può utilizzare indifferentemente uno dei tre differenti metodi messi a disposizione dall'oggetto SMSConnection, ognuno dei quali restituisce una List<SMS\_MO>:

* getNewSMS\_MOs(), che richiede al server tutti i messaggi “nuovi”, ovvero, tutti quelli ricevuti dall'ultima volta che è stata chiamata questa procedura (attenzione! questo metodo richiede attivazione da parte di Aruba, se vuoi richiedere i messaggi tramite questo metodo, contattaci). Per l'integrazione all'interno di un applicativo suggeriamo l'utilizzo di questo metodo.
* getSMS\_MOHistory(Date,Date) richiede al server tutti gli SMS ricevuti in un particolare intervallo di date
* getSMS\_MOById(long) richiede tutti gli SMS ricevuti che abbiano un identificativo univoco (campo id\_message della classe SMS\_MO) superiore a quello passato come parametro; tutti gli identificativi sono interi maggiori di zero, passando zero come parametro, quindi, si otterranno in risposta tutti gli SMS ricevuti dall'utente.

Esempio:

Try

 Dim smss\_mo As List(Of SMS\_MO) = connection.getNewSMS\_MOs()

 If smss\_mo.Count = 0 Then

 System.Console.WriteLine("nessun nuovo SMS ricevuto")

 Else

 For Each sms\_mo As SMS\_MO In smss\_mo

 System.Console.WriteLine("Nuovo SMS id " + sms\_mo.IdMessage & ":")

 System.Console.WriteLine(" - numero tel.: " + sms\_mo.Recipient)

 System.Console.WriteLine(" - mittente: " + sms\_mo.Sender)

 System.Console.WriteLine(" - testo SMS: " + sms\_mo.Message)

 System.Console.WriteLine(" - inviato il " + sms\_mo.SendDate)

 Next

 End If

Catch smscre As SMSCRemoteException

 System.Console.WriteLine("Exception from Aruba: " & smscre.getMessage())

End Try

# GESTIONE SUBACCOUNTS

[**[ INDICE ]**](#INDICE)

Un utente superaccount può utilizzare alcuni metodi dell'oggetto SMSConnection per verificare e gestire i propri subaccounts. Nello specifico

- getSubaccounts(), restituisce una List<Subaccount> contenente tutti i subaccount dell'utente corrente

- createSubaccount(Subaccount) crea e restituisce un nuovo subaccount; login e password sono generati da Aruba, e devono essere letti dall'oggetto Subaccount restituito
- lockSubaccount(Subaccount) e unlockSubaccount(Subaccount), permettono di attivare/disattivare un subaccount; in entrambi i casi viene restituito un Subaccount

Esempio:

Try

 Dim subs As List(Of Subaccount) = connection.getSubaccounts()

 If subs.Count = 0 Then

 System.Console.WriteLine("nessun subaccount configurato")

 Else

 For Each subacc As Subaccount In subs

 System.Console.WriteLine("Subaccount:")

 System.Console.WriteLine("Login " + subacc.Login)

 System.Console.WriteLine("Password " + subacc.Password)

 System.Console.WriteLine("Locked " & (If(subacc.Active, "no", "yes")))

 Next

 End If

Catch smscre As SMSCRemoteException

 System.Console.WriteLine("Exception from Aruba: " & smscre.getMessage())

End Try

# GESTIONE CREDITI SUBACCOUNTS

[**[ INDICE ]**](#INDICE)

Un utente superaccount può dare/togliere crediti ai propri subaccounts.

In base alla tipologia di credito del subaccount, sono disponibili metodi differenti.

- Ai subaccount con tipo credito *USE\_SUPER\_CREDIT* non è possibile dare/togliere crediti.

- Ai subaccount con tipo credito *HAS\_CREDIT* è possibile dare/togliere crediti tramite il metodo moveCredits(CreditMovement). Per visualizzare i crediti del subaccount si utilizza il metodo getSubaccountCredits(Subaccount), che restituisce una lista di Credit.

- Ai subaccount con tipo credito *USE\_BOTH\_CREDITS* è possibile dare crediti con il metodo createPurchase(CreditMovement) e togliere credito con il metodo deletePurchase(CreditMovement). Per visualizzare gli acquisti del subaccount, si utilizza il metodo getPurchases(Subaccount), che restituisce una List<CreditMovement>.

Esempio:

Try

 Dim sub1 As Subaccount = connection.getSubaccounts()(0)

 If sub1.Credit\_mode = Subaccount.HAS\_CREDIT Then

 'No movements allowed here!

 ElseIf sub1.Credit\_mode = Subaccount.USE\_SUPER\_CREDIT Then

 'Getting available credits

 Dim subCredits As List(Of Credit) = connection.getSubaccountCredits(sub1)

 If subCredits.Count > 0 Then

 System.Console.WriteLine((("Nation " + subCredits(0).Nation.Iso3166 & " (") + subCredits(0).CreditType & "): ") + subCredits(0).Count)

 End If

 'Giving some credit...

 Dim newmovement As New CreditMovement()

 newmovement.Subaccount\_login = sub1.Login

 newmovement.Super\_to\_sub = True

 newmovement.Sms\_type = SMSType.ALTA

 newmovement.Amount = 100

 connection.moveCredits(newmovement)

 ElseIf sub1.Credit\_mode = Subaccount.USE\_BOTH\_CREDITS Then

 'Getting purchases list

 Dim purchases As List(Of CreditMovement) = connection.getPurchases(sub1)

 'Deleting first purchase

 If purchases.Count > 0 Then

 connection.deletePurchase(purchases(0))

 End If

 'Creating new purchase

 Dim newpurch As New CreditMovement()

newpurch.Subaccount\_login = sub1.Login

newpurch.Sms\_types = New SMSType() {SMSType.ALTA}

newpurch.PricePerMessage = New Double() {0.01}

newpurch.Price = 50.0

connection.createPurchase(newpurch)

 End If

Catch smscre As SMSCRemoteException

 System.Console.WriteLine("Exception from Aruba: " & smscre.getMessage())

End Tr