**Integrazione SMS C# SDK**

Indice

[1 INTRODUZIONE 4](#__RefHeading__11357_1366233465)

[2 CONFIGURAZIONE DELLA LIBRERIA 4](#__RefHeading__11359_1366233465)

[3 CREAZIONE DI UNA CONNESSIONE 5](#__RefHeading__11361_1366233465)

[4 INVIO DI SMS 6](#__RefHeading__11363_1366233465)

[5 STATO DEI MESSAGGI 7](#__RefHeading__11365_1366233465)

[6 CANCELLAZIONE INVII POSTICIPATI 8](#__RefHeading__11367_1366233465)

[7 STORICO DEI MESSAGGI 8](#__RefHeading__11369_1366233465)

[8 CREDITO SMS DISPONIBILE 9](#__RefHeading__11371_1366233465)

[9 LETTURA DEI MESSAGGI RICEVUTI TRAMITE IL SERVIZIO DI RICEZIONE SMS 10](#__RefHeading__11373_1366233465)

[10 GESTIONE SUBACCOUNTS 11](#__RefHeading__11375_1366233465)

[11 GESTIONE CREDITI SUBACCOUNTS 12](#__RefHeading__11377_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)

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 string HOSTNAME = "http://adminsms.aruba.it";

 public const string USERNAME = "your\_aruba\_login";

 public const string PASSWORD = "your\_aruba\_password";

 public const int DEFAULT\_PORT = 80;

 public const string PROXY = "";

 public const int PROXY\_PORT = 80;

 }

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(String login, String password)
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:

SMSCConnection smsc\_connection = null;

try {

 smsc\_connection = new SMSCHTTPConnection();

} catch (SMSCException smsc\_ex) {

 System.err.println(“errore in connessione a Aruba: “+

 smsc\_ex.getMessage());

 return;

}

// 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 (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 {

 SMS sms = new SMS();

 sms.TypeOfSMS = SMSType.ALTA;

 sms.addSMSRecipient("+393491234567");

 sms.addSMSRecipient("+393499876543");

 sms.Message = "hello world!";

 sms.SMSSender = "Mittente";

 sms.setImmediate(); // oppure sms.setScheduled\_delivery(java.util.Date)

 sms.OrderId = "12345";

 SendResult result = smsc\_connection.sendSMS(sms);

} catch (SMSCRemoteException smscre) {

 System.Console.WriteLine("Exception from Aruba: "+smscre.Message);

} catch (SMSCException smsce) {

 System.Console.WriteLine("Exception creating message: "+smsce.Message);

}

# 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 {

 List<MessageStatus> smsstatus = connection.getMessageStatus("1234");

 foreach (MessageStatus message\_status in smsstatus) {

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

 if (message\_status.IsError) {

 System.Console.WriteLine("errore!");

 } else {

 if (message\_status.Status == MessageStatus.SMSStatus.DLVRD) {

 System.Console.WriteLine("consegnato.");

 } else {

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

 }

 }

 }

} catch (SMSCRemoteException smscre) {

 System.Console.WriteLine("Exception from Aruba: "+smscre.Message);

}

# 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 (SMSCRemoteException smscre) {

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

}

# 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 {

 List<SentSMS> sent\_smss =

 connection.getSMSHistory(new DateTime(2009,1,1),DateTime.Now);

 foreach (SentSMS sent\_sms 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");

 }

} catch (SMSCRemoteException smscre) {

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

}

# 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 {

 List<Credit> credits = connection.getCredits();

 foreach (Credit credit in credits) {

 if (credit.Nation!=null && credit.Nation == Nations.ITALY) {

 if (credit.CreditType == CreditType.ALTA) {

 System.Console.WriteLine(

 "Posso inviare "+credit.Count+

 " SMS con mittente personalizzato in Italia.");

 break;

 }

 }

 }

} catch (SMSCRemoteException smscre) {

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

}

# 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 {

 List<SMS\_MO> smss\_mo = connection.getNewSMS\_MOs();

 if (smss\_mo.Count == 0) {

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

 } else {

 foreach (SMS\_MO 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);

 }

 }

} catch (SMSCRemoteException smscre) {

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

}

# 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 {

 List<Subaccount> subs = connection.getSubaccounts();

 if (subs.Count==0) {

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

 } else {

 foreach (Subaccount subacc in subs) {

 System.Console.WriteLine("Subaccount:");

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

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

 System.Console.WriteLine("Locked "+ (subacc.Active ? "no" : "yes" ));

 }

 }

} catch (SMSCRemoteException smscre) {

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

}

# 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 {

 Subaccount sub1 = connection.getSubaccounts()[0];

 if (sub1.Credit\_mode == Subaccount.HAS\_CREDIT) {

 //No movements allowed here!

 }

 else if (sub1.Credit\_mode == Subaccount.USE\_SUPER\_CREDIT) {

 //Getting available credits

 List<Credit> subCredits = connection.getSubaccountCredits(sub1);

 if (subCredits.Count>0) {

 System.Console.WriteLine("Nation " +

 subCredits[0].Nation.Iso3166 +

 " (" + subCredits[0].CreditType + "): " +

 subCredits[0].Count);

 }

 //Giving some credit...

 CreditMovement newmovement = new CreditMovement();

 newmovement.Subaccount\_login = sub1.Login;

 newmovement.Super\_to\_sub = true;

 newmovement.Sms\_type = SMSType.ALTA;

 newmovement.Amount = 100;

 connection.moveCredits(newmovement);

 }

 else if (sub1.Credit\_mode == Subaccount.USE\_BOTH\_CREDITS) {

 //Getting purchases list

 List<CreditMovement> purchases = connection.getPurchases(sub1);

 //Deleting first purchase

 if (purchases.Count>0) {

 connection.deletePurchase(purchases[0]);

 }

 //Creating new purchase

CreditMovement newpurch = new CreditMovement();

newpurch.Subaccount\_login = sub1.Login;

newpurch.Sms\_types = new SMSType[]{SMSType.ALTA};

newpurch.PricePerMessage = new double[]{0.010};

newpurch.Price =50.00;

connection.createPurchase(newpurch);

 }

} catch (SMSCRemoteException smscre) {

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