Web Service – Direct Billing 3.1

Charge users directly from apps and PC websites

Integration Guide

CONTENTS

Document change log

Date	Version	Change detail
December 2009	n/a	Pre-release version
February 2010	n/a	Added GetUserId to this Web Service Moved DoPaymentForCard into this Web Service
February 2010	0.9	Preview release
March 2010	n/a	Added IsDirectBillingSupported
March 2010	1.0	First public release
April 2010	1.0.2	Documentation improvements and clarifications
April 2010	1.0.3	Renamed paymentStatus output parameter to responseCode
April 2010	1.0.4	Updated introduction
July 2010	1.0.5	Reordered Web Methods
July 2010	1.0.6	New section on getting the mobile number on-handset
September 2010	1.0.7	New section on getting the mobile number from a PC web site
September 2010	2.0.1	DoPaymentForCardRepeat added and DoPaymentForCard updated to also return a repeatCardReference, rephrasing of credit card payments paragraph. General maintenance.
November 2010	2.0.2	Added clarification that Tracking User IDs cannot be used for Direct Billing
February 2011	3.0.1	Refactored API Added DoPaymentWithPrice Added support for filtering payment methods by type Added ValidateCardForUser for first-time card payments Removed DoPaymentForCard Removed DoPaymentForCardRepeat
March 2011	3.0.2	Added GetRemainingAccesses Added GetRemainingTime Added ConsumeOneAccess
August 2011	3.1.0	Added DoRefund method to allow refund of transactions to carrier bill (where supported) or back to Bango balance. Added GetRefundStatus method to allow validation of a refund request where a carrier does not support direct-to-bill refund capability Added support for external transaction IDs to be logged in the Bango system Added support for idempotency in method calls using above external transaction ID
August 2011	3.1.1	Added UAT test cases support and documentation of “magic numbers” Changed parameter refundTypeFilter to refundType in method DoRefund
September 2011	3.1.2	Added support for client authentication via certificates passed on requests Updated WebService end points when using certificates Added new response code for invalid certificate authentication failure Removed defunct IsSecureConnection validation
September 2011	3.1.3	Added detailed logging metrics to methods for authorisation, internal and external processes
September 2011	3.1.4	Add new method CreateSubscriptionWithPrice
October 2011	3.1.5	Added more detailed description of CreateSubscriptionWithPrice and updated the description for GetRemainingTime as the existing description was misleading.
November 2011	3.1.6	Added TransactionMethods to the DoPaymentResult, DoPaymentWithPriceResult, CreateSubscriptionResult and CreateSubscriptionWithPriceResult

Introduction

The Bango Direct Billing API allows you to initiate payments from within an on-handset application or a PC web site. Rich, real-time information on each billing attempt is available – was billing successful and if not, exactly why the payment failed (e.g. user out of credit; suspended/hotlined etc). You can use this real-time payment result data to optimize your user experience.

In a standard on-handset browser-based payment experience, the user is redirected to a Bango-hosted payment page and asked to accept the charge before being returned to you for content delivery. The Direct Billing API allows you to offer the same clear payment experience outside the browser, supporting scenarios like a PC web site needing to bill a user or an on-handset application.

The payment user experience is controlled by and presented by your application or web site – Bango has no direct interaction with the end user.

Before you start

If you plan to operate a subscription service using Bango, see the Safe Subscriptions documentation for "before you start" advice.

Your responsibility to your customers

The Direct Billing API bypasses the browser-based payment flow normally used in a mobile web site. Instead, you present the UI to the user and are responsible for communicating pricing terms to the user and obtaining their approval before sending a payment request to Bango. The Direct Billing API gives you direct access to the core Bango billing platform.

Your service must comply with local rules and regulations set by mobile operators and regulatory bodies such as the MMA, PhonePayPlus and Payforit etc.

Direct Billing vs. browser-based payments

The Direct Billing API is compatible with many operator billing systems. Some operators, however, require the user to visit a payment confirmation page hosted by the operator. This requires a web browser and a data connection using one of that operator's mobile gateways, so isn't compatible with Direct Billing from within your on-handset application. All operators in the USA and UK are compatible with Direct Billing.

Even where operator billing isn't compatible with Direct Billing, credit card is always available. If a user has paid with credit card before (either via their browser or Direct Billing), you can process a card payment with a simple API call. For a first-time card user, you can collect their card details in your own application or web site and pass them to the Direct Billing API to process a payment. The User Information Web Service lets you check whether card details are already stored for a user (documentation available via the Bango.com Support Center).

To check whether Direct Billing is available for a particular User ID / operator combination, call the IsDirectBillingSupported method in this API. This will let you know whether the user involved is able to make a payment via Direct Billing and also indicates what type of payment method will be used (e.g. operator bill, PSMS, credit/debit card etc). If the payment method is credit/debit card and the user doesn't have a card on-file with Bango, you need to collect their card details and forward them to Bango. A DATA_MISSING error will be returned if you try to bill a user without card details stored.

Where Direct Billing isn't supported, or perhaps the payment method available isn't acceptable to you, the fallback behaviour is for your application to open the on-device browser and direct the user through the standard browser-based payment flow. This ensures maximum billing coverage globally.

Integrating this API

This API can be integrated into your web site or on-handset application to place charges directly onto the phone bill of your customers. All you need to know is their phone number and the mobile operator they are using, and call the DoPayment or DoPaymentWithPrice method to request payment, with real-time status available of the billing outcome.

Mobile number or Bango User ID

Typically, Bango Web Services identify and reference your customers via their Bango User ID. With the Direct Billing API, when requesting payment is collected, you need to provide the Bango User ID that should be billed. Each User ID can have a mobile phone number, credit card, PayPal account etc associated with it. It's this rich information that lets the Direct Billing API decide how best to bill each user.

But your application or web site may not know the Bango User ID to bill. In this scenario, you simply use the GetUserId method to map a mobile phone number and operator ID pair into a Bango User ID. You can get the phone number and operator details from the on-handset runtime environment.

If the phone number and operator combination you provide have not previously accessed the Bango system, they won't already have a User ID assigned, but the GetUserId method will create a new User ID and tell you what it is. This creation process is identical to that used when a user browses to bango.net using their on-handset web browser, so gives you the same identity we will later give that user when they access bango.net via their web browser.

Once you have this User ID, you can call the DoPayment or DoPaymentWithPrice methods to request a payment.

Please note: the automatic creation of new User IDs is a feature of the GetUserId method in the Direct Billing Web Service only.

The GetUserId method in the User Information Web Service does not create new User IDs – if a new mobile number/operator pair is passed to the User Information GetUserId method, a response code of NOT_FOUND will be returned.

Getting the mobile number from a handset

Most billers require the user's mobile number to be able to charge content to the bill. When billing from an on-handset application, there are a number of ways to obtain the mobile number.

1. From the handset runtime environment
2. By sending a "Mobile Identity Code" text message from the handset
3. Using Bango Identifier via the web browser

The best user experience is to detect the mobile number from within the application. However, as mobile platforms vary in capabilities and security restrictions, this often isn't possible or reliable.

On some BlackBerry devices, the mobile number that can be read by an application is stored on the SIM card and is actually user-editable, so cannot be trusted for billing. Android devices make the mobile number available, but if the user has ported to another operator, the mobile number available via the Android SDK may not be the current number.

For optimum user experience, we recommend attempting the above 3 steps in order to get the mobile number.

From the handset runtime environment

The ability to read your mobile number from the on-handset environment varies by platform. In many cases, your mobile number isn't actually stored on the phone itself, although it may be possible to write your number to the SIM card. But take care when reading a mobile number from a SIM card, as there's no guarantee the number stored on the SIM is correct – it could be any number.

On Android, the following code will read the mobile number:

```
private Context context;
context = getApplicationContext();
TelephonyManager tm =
(TelephonyManager)context.getSystemService(Context.TELEPHONY_SERVICE);
String phoneId = tm.getLine1Number();
```

iPhone and Symbian S60 don't have APIs for reading the mobile number. Other mobile platforms may provide API calls to read the number – please check the appropriate SDK documentation.

By sending a "Mobile Identity Code" text message from the handset

Where the mobile number can't be read from the operating system, sending a text message to a shortcode is an alternative way of getting the phone number and mobile operator. However, this option is only available in countries where Bango shortcodes are available. Contact Bango customer services for coverage details.

Call the GetMobileIdentityCode method in the User Information API, passing the ISO-3166 code (GBR, USA etc) for the country to which the mobile operator belongs. This method will output a unique billing identity code, shortcode and text message content. Simply send a text message from the application using the

appropriate methods in the platform's SDK to the specified shortcode, setting the message content to be the message value returned by GetMobileIdentityCode.

There are 2 ways to retrieve the mobile number and operator:

1. Call the CheckMobileIdentityCode method in the User Information API, passing the unique identity code. The mobile number and operator will be returned to you.
2. Enable MOBILEIDENTITY events in the Event Notification system. A few seconds after the text message is received by Bango, your server will receive an Event Notification containing the mobile number and operator.

See the User Information Web Service and Event Notification integration guides for more details.

Using Bango Identifier via the web browser

Open the web browser and redirect the user to a Bango Identifier URL. Once you have read the User ID and mobile operator from the URL after being redirected back to your web site, call the User Identification Web Service to get the mobile number.

Remember to disable the "Send Tracking User IDs" feature of the Identifier service via the Bango.com Properties section. Only a standard 32-bit user ID can be used with the Direct Billing API. Tracking User IDs are 64-bit and are for temporary user tracking only – not for persistent identification, so cannot be used for billing.

Getting the mobile number from a PC web site

Where billing is initiated from a PC web site, there are 2 primary ways to get a validated mobile number for a user:

1. Invite the user to type their mobile number into the PC web site, send the user a text message containing a unique PIN code and ask them to enter that PIN code into the PC web site
2. Ask the user to send a text message containing a unique code from their handset

Manually type your mobile number

Present the user with a form inviting them to type their mobile number. Once entered, send a text message to the mobile number, containing a unique PIN code. Tell the user to read the text message and type the PIN code into the PC web site, confirming receipt of the message.

This approach confirms the mobile number, but doesn't reveal the mobile operator to which the mobile number belongs. You will need to use a mobile number lookup service (available from most SMS aggregators) to determine the mobile operator.

Ask the user to send a text message containing a unique code

Sending a text message to a shortcode is an alternative way of getting the phone number and mobile operator. However, this option is only available in countries where Bango shortcodes are available. Contact Bango customer services for coverage details.

From your web site, call the GetMobileIdentityCode method in the User Information API, passing the ISO-3166 code (GBR, USA etc) for the country in which the user's browser is connecting. This method will output a unique billing identity code, shortcode and text message content. Simply invite the user to send the specified text message to the appropriate shortcode.

There are 2 ways to retrieve the mobile number and operator:

1. Call the CheckMobileIdentityCode method in the User Information API, passing the unique identity code. The mobile number and operator will be returned to you.
2. Enable MOBILEIDENTITY events in the Event Notification system. A few seconds after the text message is received by Bango, your server will receive an Event Notification containing the mobile number and operator.

See the User Information Web Service and Event Notification integration guides for more details.

Caching the User Identity

However the user identity is obtained, take care if your application caches it. If the identity is cached for too long, it's possible for the SIM to be swapped in the device, without your application noticing the change in identity.

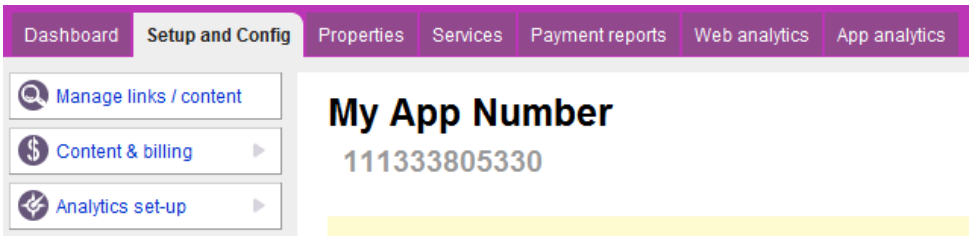
We recommend clearing any cached identity each time the device reboots.

Bango Numbers

To make payment requests using this API you must first set up a Bango Number in your package. Bango numbers are used to configure your pricing model and are also fed into the API.

To create a Bango number log into the Bango.com Management Tools and go to the **Setup and config** tab. Select **Content & billing** from the left hand menu and then select **Charge for content.**

Follow the on-screen instruction to set up a Bango number, the number will be generated and displayed at the end of the process and will begin with 111333. After you have set up your number(s) you can access them again by revisiting the **Setup and config** tab.



Network ID's

Some methods of this API use Network ID's. A list of these can be found here - http://bango.custhelp.com/app/answers/detail/a_id/842.

Requesting a Payment

There are several types of payment that can be collected using this API:

1. A one-time payment
2. A payment to allow access for a period of time
3. A payment to start a subscription
4. A subscription renewal payment

Types of Payment Method

For many users, a choice of payment methods is available when collecting payment. The possible types are:

- OPERATOR – on-bill payment
- PSMS – on-bill payment via premium text message
- CARD – credit/debit card
- INTERNET – other Internet payment types (PayPal etc)

In some situations, you might want to prevent certain types of payment method from appearing, or even force just a single type to be used (for example, to force a payment via credit card).

The DoPayment / DoPaymentWithPrice and CreateSubscription / CreateSubscriptionWithPrice methods used to request a payment allow an optional list of the above payment method types to be supplied, filtering the list of payment methods to only those matching the requested types.

If you want to process a payment or start a new subscription using credit card, check the UserInformation.HasCreditCard method to check the user has a card already on file. If they don't, your application needs to collect the card details and pass to ValidateCardForUser before attempting to collect a payment or start a subscription.

One-time payments

To make a one-time payment request, you first need to define a price point via the Bango.com Management Tools. Set up a Bango Number with the appropriate pricing configuration – e.g. how much to charge in each currency.

Then call the DoPayment method of this Web Service, passing the Bango User ID you want to collect payment from, and the Bango Number for which you've configured the amount to charge.

For a Relay-enabled Bango Number, the price can be changed in each request to DoPaymentWithPrice. The price can be specified in any number of supported currencies and the price in the currency appropriate for the user will be used.

Timed-access payments

To make a timed-access payment request, you first need to define a price point via the Bango.com Management Tools. Set up a Bango Number with the appropriate pricing configuration – e.g. how much to charge in each currency and how much access time a payment gives you.

Then call the DoPayment method of this Web Service, passing the Bango User ID you want to collect payment from, and the Bango Number for which you've configured the amount to charge.

For a Relay-enabled Bango Number, the price can be changed in each request to DoPaymentWithPrice. The price can be specified in any number of supported currencies and the price in the currency appropriate for the user will be used.

In the normal browser-based payment flow, Bango makes sure a user can only access your service if they are within the time period their payment covered. With the Direct Billing API, although you can tell the Bango system how much time a payment covers for reporting purposes, as you present the UI to the user, it's your responsibility to make sure only a user who has paid can access your service.

Payments to start a subscription (managed by Bango)

To start a time-based subscription, you first need to define a price point via the Bango.com Management Tools. Set up a Bango Number with the appropriate pricing configuration – e.g. how much to charge in each currency and the subscription time period (e.g. $9.99 per month).

Now call the CreateSubscription web method. This requires you provide the Bango User ID and Bango Number for which to start a subscription.

The first subscription payment will be collected and Bango will automatically manage subscription renewal payments and any associated billing retries.

The CreateSubscription method outputs the ID of the newly-created subscription, along with the transaction ID of any payment collected at the same time. This helps you maintain an audit trail in your system.

If you wish to dynamically override the pricing configuration at runtime, you can call the CreateSubscriptionWithPrice method, passing in the appropriate currency and amount values for the subscription renewal price and an optional initial price.

See the Safe Subscriptions integration guide for more details on the Bango managed subscriptions service.

Subscription renewal payments

Bango automatically collects subscription renewal payments when due. A real-time feed of notifications of renewal payment success/fail rates is sent to you via our Event Notification service. You receive full details of all renewal billing attempts and any subsequent retries.

It isn't necessary to manually ask Bango to collect renewal payments – the process is fully automated.

Credit Card Payments

Direct Billing Credit card payments are not enabled by default. To process credit card transactions contact support@bango.com to request this feature.

The payment flow displayed to the user must meet with the following requirements:

- You must state that the content is being purchased from Bango
- You must state that the following will appear on the users credit card statement:
  - '8779272036.com purchase' for R rated USA transactions
  - 'Bango Mobile' for any other transactions
- You must not display any Visa or MasterCard logos

You can find out if a user has a credit card stored before attempting a payment via the User Information Web Service (the documentation for this API can be downloaded from the bango.com Support Center). This Web Service also allows you to enquire about many other aspects of a user account.

The first time a credit card payment is processed for any user, their card details need to be collected and stored for future use. If no card details are stored for a user and you try to process a card payment, a DATA_MISSING status code will be returned.

First Card Payment

You can collect card details and use the ValidateCardForUser method to store the card against a user account:

- Card number (16-21 digits in length)
- Start date (MMYY) (only required for some card types)
- Expiry date (MMYY)
- CVV code (3 digits)
- Issue number (only required for some card types)
- ZIP code (if card issued by USA bank)
- User's email address **OPTIONAL**
- User's current IP address

It's recommended to collect the user's email address, as this allows Bango to send the user an email receipt for their payment. Giving the user this information reduces the chance of the user disputing the charge on their card statement later, resulting in a chargeback and a financial impact on you.

After validating the input parameters, card authorization is attempted with the card processor and, if successful, the card details are stored against the user account so future card payments can be processed with just 1 click.

Once a card has been validated for a user account, you can call DoPayment to collect an initial payment just as for repeat payments or CreateSubscription to start a new subscription.

Repeat Card Payments

You can use DoPayment specifying a type of just CARD to complete a card payment for a user who has card details saved from an initial payment.

It's not necessary to collect card details for each subsequent payment – only the first time a card is validated.

Real-Time Price Control

A call to one of the DoPayment methods will charge the user the price you have pre-configured for the specified Bango Number via the Bango Management Tools. You can override this price in real-time for a Relay-enabled Bango Number by instead using the DoPaymentWithPrice method.

This method accepts an array of amount/currency pairs. If a price is supplied in the currency the specified user account uses, that's the price the user will be billed. If you don't have a price specified in the user' account currency, the price in your package currency will be used and automatically converted to the user's currency.

The pre-configured prices in the Bango system are used as a fallback if no appropriate currency price is supplied to DoPaymentWithPrice.

Please note: due to operator billing restrictions, subscription price points must be pre-configured via the Bango.com Management Tools. Real-time price control via the Direct Billing API supports only non-subscription services.

Managing timed-access or number-of-accesses services

Payments can be processed for a number of accesses or for a period of time. Bango keeps track of how many paid-for accesses or how much time remains.

Use the GetRemainingAccesses and GetRemainingTime methods to find out how much access remains.

For numbered accesses, use the ConsumeOneAccess method each time the remaining access count should be decreased (e.g. each time access-controlled functionality is accessed). Once the number of accesses left has hit zero and can't be decreased any further, you need to invite the user to extend their access by making another payment.

Payment Status Codes

When a request is made to DoPayment, a payment status code is returned in the responseCode field of the web service response. A status code of “OK” indicates payment was successfully collected.

As the status codes used by each biller vary significantly, the Direct Billing API uses a single set of status codes as a superset of all possible payment status codes each biller could output. For example, Sprint has a “hotlined” status indicating a user is temporarily suspended from placing charges on their phone bill. As billers use their own different status codes for this same suspended condition, using a single standard set of status codes in this API means biller-specific behaviour is abstracted.

Any other status code indicates why payment could not be collected:

Status Code	Description
CONNECT_ERROR	Connection error submitting the payment request to the biller
CONNECT_TIMEOUT	Connection timeout submitting the payment request to the biller
DATA_INVALID	The biller has rejected the payment request due to invalid data (e.g. invalid phone number)
DATA_MISSING	The biller has rejected the payment request due to missing data (e.g. no phone number, no credit card details stored)
DECLINED	The biller declined the payment request (e.g. credit card declined)
ERROR	Unspecified error
NOT_AVAILABLE	No direct billing methods available
NOT_SUPPORTED	The user or operator (e.g. MVNO) is not supported by the biller
SPEED_LIMIT	The biller has rejected a payment request too soon after the previous one for this user
USER_BARRED	The user is not allowed to use this payment method
USER_INSUFFICIENT_CREDIT	The user does not have enough credit for this payment
USER_INVALID	The biller does not recognize this user
USER_NOT_ENABLED	The biller has not enabled this user for payments
USER_NOT_FOUND	The biller does not recognize this user
USER_SPEND_LIMIT	The user has hit a spend limit in the biller’s system
USER_SUSPENDED	The user is temporarily not allowed to use this payment method (e.g. Sprint “hotlined” status)
NEEDS_INTERACTION	The user needs be redirected to the standard browser-based payment flow
WAITNG	A payment collection attempt has been made, but the biller has not yet confirmed whether this was successful. Most likely to happen when billing via PSMS and waiting for the operator to send Bango a message delivery report, letting us know whether billing worked or not
ERROR	Any other type of billing error

The following status codes are specific to credit card payments:

Status Code	Description
COUNTRY_MISMATCH	The country to which the card is registered does not match the country of the user ID or current client IP connection
COUNTRY_NEEDED	This card was issued in the USA, so the user's country is needed

These real-time payment status codes allow you to know exactly why a specific payment request has failed. Typically, most failures will be due to a user either blocked from making payments by the biller (e.g. Sprint can "hotline" some customers) or because a user either has no credit available on their pre-pay account, or has hit a monthly spend limit (e.g. $100/month on AT&T).

Refunding

Any payment can be refunded by calling the DoRefund method, passing the Bango transaction ID output when the payment was originally collected.

You can either refund back to the user's carrier bill (where this functionality is supported by their carrier), or refund to their Bango account balance.

To refund back to the user's carrier bill, you should pass a refundType with a value of OPERATOR.

Where a carrier supports a direct refund API, we will immediately inform you of the success or failure of a refund request. Some carriers only support a delayed refund request; in this instance a response of PENDING will be returned. You can check the status of this refund request by calling the GetRefundStatus method which will indicate whether the refund request was successful or not.

If refunding to carrier bill is not supported, you can alternatively refund to the user's Bango account balance. The value of the original payment is refunded to the user's Bango stored value account, giving them a credit balance that can be spent on future purchases.

To refund back to the Bango account you should pass a refundType with a value of BANGO.

External Transaction IDs and Idempotency

All transactional methods now support an optionally supplied external transaction ID which can be generated by your system and added alongside payment and refund requests within the Bango system. This external transaction ID can then be used in reports generated by Bango for reconciliation purposes within your own systems.

If you wish to enforce idempotency on transaction requests, you must ensure an external transaction ID is supplied on each request.

Idempotency refers to a method whose response is always the same for given input parameters whilst those parameters remain the same. That is, if you are processing a credit card payment, subsequent calls to DoPayment with the same parameters would always return you the response which was given on the first call, however, subsequent calls can not effect a state change.

This can be useful to avoid scenarios such as double billing – if you were to call DoPayment twice, the first call may result in a successful purchase; the second call would not request a new payment to be processed, but would return the same result as the initial call, with exactly the same parameters such as response codes, transaction ID, etc.

Direct Billing idempotency is not enabled by default. To enforce this on all transactional methods contact support@bango.com to request this feature.

User Acceptance Testing Support

Test responses have now been added to certain methods to allow external UAT where known responses will be returned from method calls to allow you as the integrator to code against all response codes.

To enable UAT support, contact support@bango.com to request this feature. When UAT is enabled, any calls to methods will only respond with fixed response codes, and the actual request will not be processed. It is recommended that once your integration and UAT has been completed, this feature is then disabled.

To use this feature, you must still have valid Web Service access credentials, and a valid Bango Number to use, as all initial validation checks will be performed.

You can receive UAT responses by passing in a “magic number” for the specified parameters detailed below:

IsDirectBillingSupported

Parameter	Value	Response Code	Response Message
userId	-100001	OK	Success
userId	-100002	INVALID_BANGO	Invalid Bango number
userId	-100003	INVALID_USERID	Invalid User ID
userId	-100004	INVALID_ACCESSMODEL	Bango number does not have the correct access model
userId	-100005	INTERNAL_ERROR	A problem on the server meant that the request could not be processed

ValidateCardForUser

Parameter	Value	Response Code	Response Message
userId	-100001	OK	Success
userId	-100002	INVALID_BANGO	Invalid Bango number
userId	-100003	INVALID_USERID	Invalid User ID
userId	-100004	INVALID_ACCESSMODEL	Bango number does not have the correct access model
userId	-100005	INTERNAL_ERROR	A problem on the server meant that the request could not be processed
userId	-200001	INVALID_CARDNUMBER	Card number must be 16-21 digits
userId	-200002	INVALID_CARDSTART	Card expiry must be a valid date in the format MMYY
userId	-200003	INVALID_CARDSTART	Month (MM) must be between 1 and 12
userId	-200004	INVALID_CARDSTART	Card start cannot be a date in the future
userId	-200005	INVALID_CARDSTART	Card start is too old
userId	-200006	INVALID_CARDEXPIRY	Card expiry must be a valid date in the format MMYY
userId	-200007	INVALID_CARDEXPIRY	Month (MM) must be between 1 and 12
userId	-200008	INVALID_CARDEXPIRY	Card expiry cannot be a date in the past
userId	-200009	INVALID_CARDEXPIRY	Card expiry is too far in the future
userId	-200010	INVALID_CARDCVV	Card CVV must be 3 digits
userId	-200011	INVALID_CARDISSUE	Card expiry must be an integer value
userId	-200012	INVALID_CARDZIP	Card ZIP must be 5 digits
userId	-200013	INVALID_EMAIL	Invalid email address
userId	-200014	INVALID_IP	Invalid IP address

DoPayment / DoPaymentWithPrice / CreateSubscription / CreateSubscriptionWithPrice

Parameter	Value	Response Code	Response Message
userId	-100001	OK	Success
userId	-100002	INVALID_BANGO	Invalid Bango number
userId	-100003	INVALID_USERID	Invalid User ID
userId	-100004	INVALID_ACCESSMODEL	Bango number does not have the correct access model
userId	-100005	INTERNAL_ERROR	A problem on the server meant that the request could not be processed
userId	-100006	DATA_MISSING	The user ID does not have a mobile number or credit card stored, depending on how payment was attempted
userId	-200001	CONNECT_ERROR	Connection error submitting the payment request to the biller
userId	-200002	CONNECT_TIMEOUT	Connection timeout submitting the payment request to the biller
userId	-200003	DATA_INVALID	The biller has rejected the payment request due to invalid data (e.g. invalid phone number)

userId	-200004	DATA_MISSING	The biller has rejected the payment request due to missing data (e.g. no phone number, no credit card details stored)
userId	-200005	DECLINED	The biller declined the payment request (e.g. credit card declined)
userId	-200006	ERROR	Unspecified error
userId	-200007	NOT_AVAILABLE	No direct billing methods available
userId	-200008	NOT_SUPPORTED	The user or operator (e.g. MVNO) is not supported by the biller
userId	-200009	SPEED_LIMIT	The biller has rejected a payment request too soon after the previous one for this user
userId	-200010	USER_BARRED	The user is not allowed to use this payment method
userId	-200011	USER_INSUFFICIENT_CREDIT	The user does not have enough credit for this payment
userId	-200012	USER_INVALID	The biller does not recognize this user
userId	-200013	USER_NOT_ENABLED	The biller has not enabled this user for payments
userId	-200014	USER_NOT_FOUND	The biller does not recognize this user
userId	-200015	USER_SPEND_LIMIT	The user has hit a spend limit in the biller's system
userId	-200016	USER_SUSPENDED	The user is temporarily not allowed to use this payment method (e.g. Sprint "hotlined" status)
userId	-200017	NEEDS_INTERACTION	The user needs be redirected to the standard browser-based payment flow
userId	-200018	WAITING	A payment collection attempt has been made, but the biller has not yet confirmed whether this was successful. Most likely to happen when billing via PSMS and waiting for the operator to send Bango a message delivery report, letting us know whether billing worked or not
userId	-300001	COUNTRY_MISMATCH	The country to which the card is registered does not match the country of the user ID or current client IP connection
userId	-300002	COUNTRY_NEEDED	This card was issued in the USA, so the user's country is needed
userId	-400001	SUBSCRIPTION_EXISTS	User already subscribed to Bango number

DoRefund

Parameter	Value	Response Code	Response Message
transactionId	-100001	OK	Success
transactionId	-100002	INVALID_BANGO	Invalid Bango number
transactionId	-100003	INVALID_USERID	Invalid User ID
transactionId	-100004	INVALID_ACCESSMODEL	Bango number does not have the correct access model
transactionId	-100005	INVALID_REFUND_TYPE	The specified refund type is invalid
transactionId	-100006	INVALID_TRANSACTIONID	The specified transactionId is invalid
transactionId	-100007	INTERNAL_ERROR	A problem on the server meant that the request could not be processed
transactionId	-200001	PENDING	The refund has been requested but not yet actioned
transactionId	-200002	CANT_REFUND	It's not possible to refund this transaction
transactionId	-200003	ALREADY_REFUNDED	The transaction has already been refunded
transactionId	-200004	NOT_SUPPORTED	Refunding this transaction is not supported

GetRefundStatus

Parameter	Value	Response Code	Response Message
refundTransactionId	-100001	OK	Success
refundTransactionId	-100002	INVALID_BANGO	Invalid Bango number
refundTransactionId	-100003	INVALID_REFUND_ TRANSACTIONID	The specified refundTransactionId is invalid
refundTransactionId	-100004	INTERNAL_ERROR	A problem on the server meant that the request could not be processed
refundTransactionId	-200001	PENDING	The refund has been requested but not yet actioned
refundTransactionId	-200002	CANT_REFUND	It's not possible to refund this transaction
refundTransactionId	-200003	ALREADY_REFUNDED	The transaction has already been refunded
refundTransactionId	-200004	NOT_SUPPORTED	Refunding this transaction is not supported

GetRemainingTime / GetRemainingAccesses

Parameter	Value	Response Code	Response Message
userId	-100001	OK	Success
userId	-100002	INVALID_BANGO	Invalid Bango number
userId	-100003	INVALID_USERID	Invalid User ID
userId	-100004	INVALID_ACCESSMODEL	Bango number does not have the correct access model
userId	-100005	NO_PAYMENT	No existing payment could be found
userId	-100006	INTERNAL_ERROR	A problem on the server meant that the request could not be processed

ConsumeOneAccess

Parameter	Value	Response Code	Response Message
userId	-100001	OK	Success
userId	-100002	INVALID_BANGO	Invalid Bango number
userId	-100003	INVALID_USERID	Invalid User ID
userId	-100004	INVALID_ACCESSMODEL	Bango number does not have the correct access model
userId	-100005	ACCESS_EXPIRED	No accesses remain
userId	-100006	INTERNAL_ERROR	A problem on the server meant that the request could not be processed

Web Services API

The Web Services API is a secure server-to-server API (SSL only), offering full control over managing payments for your direct billing integration.

The API contains methods to query information for a particular user and the available payment methods they may have available as well as methods to perform both Payments and Refund direct to a user's carrier bill, and creating of subscriptions.

Security

The Bango Web Services API is protected to ensure that only authorized clients use it. There are three primary levels of security:

The Web Services API allows your server to send requests to the Bango server. The Bango server will only accept requests from authorized IP addresses. Requests directly from an on-handset application are not allowed – the application must communicate with your server, which must then communicate with the Bango Web Services API.

- API username/password and third-party account authentication
- Client IP address validation
- Secure Sockets Layer (SSL) data transport

A failure at any one of these security levels denies access to the Bango Web Services API.

Please note: the certificate used by the Bango Web Services API is issued by Thawte. Please make sure Thawte is configured on your servers as a Trusted Root Certification Authority (CA).

There is also an optional level of security utilizing Digital ID certificates. Certificates can be attached to each request, and Bango will validate your certificate using the Serial Number, Thumbnail and Expiry Date.

If you wish to use this additional security, please note the port number for connectivity should be changed from the default 443 to **8443**. For details on how to supply your certificate details, please see overleaf.

WSDL Location

WSDL can be located at the following URL.

https://webservices.bango.com/directbilling_v3_1/?wsdl

For those wishing to use Digital ID certificates, the WSDL can be located at:

https://webservices.bango.com:8443/directbilling_v3_1/?wsdl

Requesting Web Services access

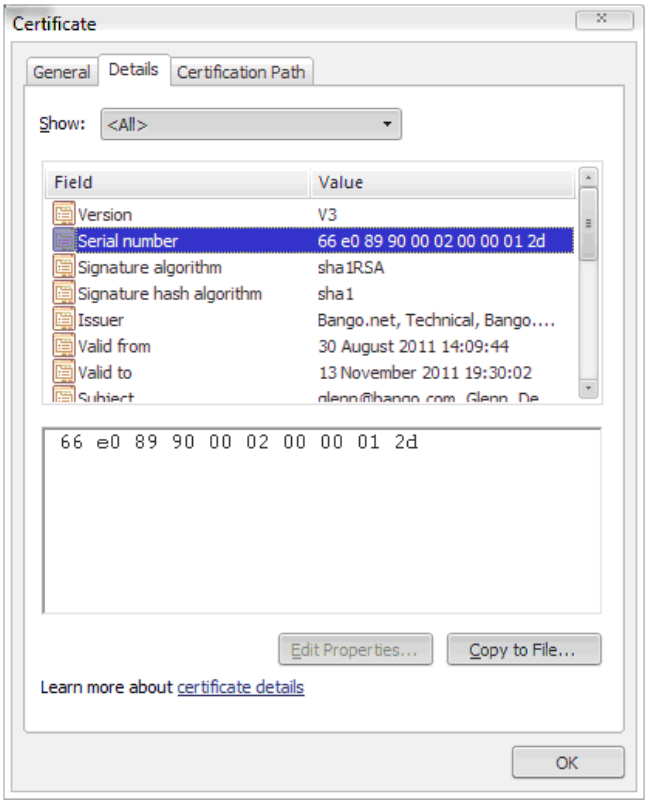
Before you can use the Web Services API, you need to first request access credentials. Login to the Management tools on bango.com and go to the Setup and Config tab, then select 'Web Service API's' on the left.

Requesting Client Certificate Security

To begin, you will need to purchase a suitable client certificate – Bango recommend a Verisign Digital ID certificate.

Once you have purchased and configured your certificate, you will need to contact Bango customer services and supply them the **Serial Number**, **Thumbnail** and **Expiry Date** for the certificate you wish to use.

An example of how to find these details using Microsoft Windows is shown below:



These details will be added to your WebServices credentials and configured to be required on every request.

Once this is done, you should ensure your integration end point uses the new 8443 port when connecting, and that the certificate is attached to each request. Once enabled, failure to provide a valid certificate will deny access to the Bango Web Services API.

Method: **GetUserId**

Description: Returns the User Id for a mobile phone number on a specific operator. If the user is already known to the Bango system, their User ID will be returned.

Please note: the mobile phone number must be in international format, including country code (e.g. 15551234567 for a USA mobile number and 447710123456 for a UK mobile number).

If the user is not known, but is on an operator which passes real-time MSISDN's to Bango, a new user account is automatically created and the User ID returned. When the user does access the Bango system and their MSISDN is passed to Bango by their operator, this is the User ID they will be assigned.

Inputs:

username	String(20)	Your username
password	String(20)	Your password
mobileNumber	String(20)	The mobile number to search for – must be international format i.e. 16141234567 or 447787123456
networkId	String(50)	Bango operator id (see Relay documentation for list)
bango	String(50)	A Bango Number accessible by these credentials

Outputs:

responseCode	String(50)	Status of request.
responseMessage	String(255)	Textual description of request status
userId	Int32	User ID for this mobile number

Response Codes:

OK	Success
ACCESS_DENIED	Invalid username/password
ACCESS_DENIED	Invalid IP address
ACCESS_DENIED	Invalid certificate
ACCESS_DENIED	Invalid access to specified Bango number
INVALID_BANGO	Invalid Bango number
INTERNAL_ERROR	A problem on the server meant that the request could not be processed
NOT_FOUND	No user ID for mobile number

Method: **IsDirectBillingSupported**

Description: Checks whether Direct Billing is supported for a specific User ID, outputting a list of the payment methods indicating how the user will pay if a Direct Billing payment is requested.

Some operators require the user visits an operator-hosted payment confirmation page – it's not possible to use the Direct Billing API to place a charge directly onto the phone bill on these operators, as the user is not involved in the transaction, so can't view the operator-hosted page.

Regardless of whether an operator supports Direct Billing, a credit card Direct Billing payment can be collected for any user.

The "type" output parameter indicates what type of payment method will be used and will be one of these values:

OPERATOR	Operator billing (on-bill charge)
PSMS	Premium SMS
CARD	Credit or debit card
INTERNET	Internet billing method (e.g. PayPal, Google Checkout etc)

Inputs:

username	Your username
password	Your password
bango	Bango number
userId	Bango User ID
typeFilter	Array of allowed payment method types
priceList	Array of price entities, each comprising the following: amount, int, no nulls currency, char(3), no nulls

Outputs:

responseCode	Status of service call
responseMessage	Textual description of service status
supported	"Y" if this operator supports Direct Billing
defaultType	Type of first (default) payment method available
paymentMethods	List of payment methods available: methodId, String(50) type, String(20)

Response Codes:

OK	Success
ACCESS_DENIED	Invalid username/password
ACCESS_DENIED	Invalid client IP address
ACCESS_DENIED	Invalid certificate
ACCESS_DENIED	Invalid access to specified Bango number
INVALID_BANGO	Invalid Bango number
INVALID_USERID	Invalid User ID
INTERNAL_ERROR	A problem on the server meant that the request could not be processed

Method: **ValidateCardForUser**

Description: Attempts to validate card details with the card provider, verifying the card can be used to process payments.

Once validated, the card is registered against the supplied user ID, enabling it for payments to be processed.

If any of the supplied card details fail validation (e.g. incorrect card number, ZIP code doesn't match country of card issuing bank etc), an appropriate status code will be returned, along with a textual description of the failure reason.

A ZIP code is required for all cards issued in the USA. Email address is optional and allows an email receipt to be sent to the user.

Inputs:

username	Your username
password	Your password
bango	Bango number
userId	User ID
cardNumber	16-21 digits of card number
cardStart	Optional Card start date in MMYY format
cardExpiry	Card expiry date in MMYY format
cardCVV	Card CVV (3 digits)
cardIssue	Optional Card issue number
zip	5-digit zip code (required for US cards)
email	Optional email address to which an email receipt should be sent
clientIP	IP address of the client connection seen by your server

Outputs:

responseCode	Result of the payment request (see below)
responseMessage	Any additional information to clarify the payment status code

Response Codes:

OK	Success
ACCESS_DENIED	Invalid username/password
ACCESS_DENIED	Invalid client IP address
ACCESS_DENIED	Invalid certificate
ACCESS_DENIED	Invalid access to specified Bango number
INVALID_BANGO	Invalid Bango number
INVALID_USERID	Invalid User ID
INVALID_ACCESSMODEL	Bango number does not have the correct access model
INVALID_CARDNUMBER	Card number must be 16-21 digits
INVALID_CARDSTART	Card expiry must be a valid date in the format MMYY
INVALID_CARDSTART	Month (MM) must be between 1 and 12
INVALID_CARDSTART	Card start cannot be a date in the future
INVALID_CARDSTART	Card start is too old
INVALID_CARDEXPIRY	Card expiry must be a valid date in the format MMYY
INVALID_CARDEXPIRY	Month (MM) must be between 1 and 12
INVALID_CARDEXPIRY	Card expiry cannot be a date in the past
INVALID_CARDEXPIRY	Card expiry is too far in the future
INVALID_CARDCVV	Card CVV must be 3 digits
INVALID_CARDISSUE	Card expiry must be an integer value
INVALID_CARDZIP	Card ZIP must be 5 digits
INVALID_EMAIL	Invalid email address

INVALID_IP	Invalid IP address
INTERNAL_ERROR	A problem on the server meant that the request could not be processed
NOT_CARD_OWNER	Cannot bill user - this credit card is locked to another user

Method: **DoPayment**

Description: Makes a payment request for the specified User ID and Bango Number. The Bango Number must be pre-configured via the Bango.com Management Tools with the appropriate end-user pricing.

The responseCode output parameter indicates the payment status result. A value of “OK” indicates a successful payment.

Inputs:

username	Your username
password	Your password
bango	Bango number
userId	User ID
typeFilter	Array of allowed payment method types
externalTransactionId	An optional transaction ID generated by your system which can be logged against a payment request within the Bango system.

Outputs:

responseCode	Result of the payment request (see below)
responseMessage	Any additional information to clarify the payment status code
transactionId	Payment ID of successful transaction
transactionMethods	An array of TransactionMethod, comprised of: • networkId • networkDescription • paymentMethodId • paymentMethodDescription

Response Codes:

OK	Success
ACCESS_DENIED	Invalid username/password
ACCESS_DENIED	Invalid client IP address
ACCESS_DENIED	Invalid certificate
ACCESS_DENIED	Invalid access to specified Bango number
INVALID_BANGO	Invalid Bango number
INVALID_USERID	Invalid User ID
INVALID_ACCESSMODEL	Bango number does not have the correct access model
INTERNAL_ERROR	A problem on the server meant that the request could not be processed
DATA_MISSING	The user ID does not have a mobile number or credit card stored, depending on how payment was attempted
<other>	Payment specific error code (see Payment Status Codes table)

Method: **DoPaymentWithPrice**

Description: Makes a payment request for the specified User ID and Bango Number. The Bango Number must be pre-configured via the Bango.com Management Tools with the appropriate access model and default prices. The prices list supplied is used to override the pre-configured prices in the database.

The responseCode output parameter indicates the payment status result. A value of "OK" indicates a successful payment.

Inputs:

username	Your username
password	Your password
bango	Bango number
userId	User ID
typeFilter	String array of allowed payment method types
priceList	Array of price entities, each comprising the following: amount, int, no nulls currency, char(3), no nulls
externalTransactionId	An optional transaction ID generated by your system which can be logged against a payment request within the Bango system.

Outputs:

responseCode	Result of the payment request (see below)
responseMessage	Any additional information to clarify the payment status code
transactionId	Payment ID of successful transaction
transactionMethods	An array of TransactionMethod, comprised of: • networkId • networkDescription • paymentMethodId • paymentMethodDescription

Response Codes:

OK	Success
ACCESS_DENIED	Invalid username/password
ACCESS_DENIED	Invalid client IP address
ACCESS_DENIED	Invalid certificate
ACCESS_DENIED	Invalid access to specified Bango number
INVALID_BANGO	Invalid Bango number
INVALID_USERID	Invalid User ID
INVALID_ACCESSMODEL	Bango number does not have the correct access model
INTERNAL_ERROR	A problem on the server meant that the request could not be processed
DATA_MISSING	The user ID does not have a mobile number or credit card stored, depending on how payment was attempted
<other>	Payment specific error code (see Payment Status Codes table)

Method: **CreateSubscription**

Description: Creates a new time-based subscription for the specified User ID and Bango Number.

If an active subscription already exists for the User ID and Bango Number, a status code of SUBSCRIPTION_EXISTS is returned.

If the subscription requires an initial payment (e.g. it doesn't have an initial free period), if collecting that payment fails, no subscription will be created and the payment status code describing the payment failure will be returned as the status code from this method.

Inputs:

username	Your username
password	Your password
bango	Bango number
userId	User ID
typeFilter	Array of allowed payment method types
externalTransactionId	An optional transaction ID generated by your system which can be logged against a payment request within the Bango system.

Outputs:

responseCode	Result of the subscription creation request (see below)
responseMessage	Any additional information to clarify the payment status code
subscriptionId	Subscription ID
transactionId	Payment ID
transactionMethods	An array of TransactionMethod, comprised of: • networkId • networkDescription • paymentMethodId • paymentMethodDescription

Response Codes:

OK	Success
ACCESS_DENIED	Invalid username/password
ACCESS_DENIED	Invalid client IP address
ACCESS_DENIED	Invalid certificate
ACCESS_DENIED	Invalid access to specified Bango number
INVALID_BANGO	Invalid Bango number
INVALID_USERID	Invalid User ID
INVALID_ACCESSMODEL	Bango number does not have the correct access model
SUBSCRIPTION_EXISTS	User already subscribed to Bango number
INTERNAL_ERROR	A problem on the server meant that the request could not be processed
<other>	Payment specific error code (see Payment Status Codes table)

Method: **CreateSubscriptionWithPrice**

Description: Creates a new time-based subscription for the specified User ID and Bango Number. The Bango Number must be pre-configured via the Bango.com Management Tools with the appropriate access model and default prices. The prices list supplied is used to override the pre-configured prices in the database.

If an active subscription already exists for the User ID and Bango Number, a status code of SUBSCRIPTION_EXISTS is returned.

If the subscription requires an initial payment (e.g. it doesn't have an initial free period), if collecting that payment fails, no subscription will be created and the payment status code describing the payment failure will be returned as the status code from this method.

If an initial price is configured when the subscription is created and no initial price list is supplied when CreateSubscriptionWithPrice is called, the configured initial price will be charged for the initial period.

If any or both of the price lists contain only invalid currencies (must be either the user's currency or the default currency setup for the subscription upon creation) the error INVALID is returned.

If the initial period is free, no payment methods will be returned in the transactionMethods.

Inputs:

username	Your username
password	Your password
bango	Bango number
userId	User ID
typeFilter	Array of allowed payment method types
initialPriceList	Array of price entities, each comprising the following: amount, int, no nulls currency, char(3), no nulls
priceList	Array of price entities, each comprising the following: amount, int, no nulls currency, char(3), no nulls
externalTransactionId	An optional transaction ID generated by your system which can be logged against a payment request within the Bango system.

Outputs:

responseCode	Result of the subscription creation request (see below)
responseMessage	Any additional information to clarify the payment status code
subscriptionId	Subscription ID
transactionId	Payment ID
transactionMethods	An array of TransactionMethod, comprised of: • networkId • networkDescription • paymentMethodId • paymentMethodDescription

Response Codes:

OK	Success

ACCESS_DENIED	Invalid username/password
ACCESS_DENIED	Invalid client IP address
ACCESS_DENIED	Invalid certificate
ACCESS_DENIED	Invalid access to specified Bango number
INVALID_BANGO	Invalid Bango number
INVALID_USERID	Invalid User ID
INVALID_ACCESSMODEL	Bango number does not have the correct access model
SUBSCRIPTION_EXISTS	User already subscribed to Bango number
INTERNAL_ERROR	A problem on the server meant that the request could not be processed
<other>	Payment specific error code (see Payment Status Codes table)

Method: **DoRefund**

Description: Refunds a previous payment, either refunding a transaction back to the user's carrier bill (where supported) or crediting the user's Bango account with the value of the original payment, ready to be spent on future content purchases dependent on the requested type filter.

Inputs:

username	Your username
password	Your password
bango	Bango number
transactionId	Payment transaction ID to refund
refundType	Specifies whether a refund should be attempted to the user's carrier bill or back to Bango stored value account. You should specify **OPERATOR** to attempt a carrier bill refund, or **BANGO** for refund back to the user's Bango account.
externalTransactionId	An optional transaction ID generated by your system which can be logged against a refund request within the Bango system.

Outputs:

responseCode	Result of the refund request (see below)
responseMessage	Any additional information to clarify the refund status code
refundTransactionId	Transaction ID of refund

Response Codes:

OK	Success
ACCESS_DENIED	Invalid username/password
ACCESS_DENIED	Invalid client IP address
ACCESS_DENIED	Invalid certificate
ACCESS_DENIED	Invalid access to specified Bango number
INVALID_BANGO	Invalid Bango number
INVALID_ACCESSMODEL	Bango number does not have the correct access model
INVALID_REFUND_TYPE	The specified refund type is invalid
INVALID_TRANSACTIONID	The specified transactionId is invalid
INTERNAL_ERROR	A problem on the server meant that the request could not be processed
PENDING	The refund has been requested but not yet actioned
CANT_REFUND	It's not possible to refund this transaction
ALREADY_REFUNDED	The transaction has already been refunded
NOT_SUPPORTED	Refunding this transaction is not supported

Method:	**GetRefundStatus**
Description:	Gets the status of a refund which has previously been marked as pending following a DoRefund request

Inputs:

username	Your username
password	Your password
bango	Bango number
refundTransactionId	Transaction ID of refund

Outputs:

responseCode	Result of the refund request (see below)
responseMessage	Any additional information to clarify the refund status code

Response Codes:

OK	Success
ACCESS_DENIED	Invalid username/password
ACCESS_DENIED	Invalid client IP address
ACCESS_DENIED	Invalid certificate
ACCESS_DENIED	Invalid access to specified Bango number
INVALID_BANGO	Invalid Bango number
INVALID_REFUND_TRANSACTIONID	The specified refundTransactionId is invalid
INTERNAL_ERROR	A problem on the server meant that the request could not be processed
PENDING	The refund has been requested but not yet actioned
CANT_REFUND	It's not been possible to refund this transaction
NOT_SUPPORTED	Refunding this transaction is not supported

Method:	**GetRemainingTime**
Description:	Gets the expiry date for an already-paid-for time period. The time returned is in UTC. If the response code is not OK the date returned is not valid.

Inputs:

username	Your username
password	Your password
bango	Bango number
userId	User ID

Outputs:

responseCode	Result of the payment request (see below)
responseMessage	Any additional information to clarify the payment status code
expires	Date and time (UTC) paid-for access expires

Response Codes:

OK	Success
ACCESS_DENIED	Invalid username/password
ACCESS_DENIED	Invalid client IP address
ACCESS_DENIED	Invalid certificate
ACCESS_DENIED	Invalid access to specified Bango number
INVALID_BANGO	Invalid Bango number
INVALID_USERID	Invalid User ID
INVALID_ACCESSMODEL	Bango number does not have the correct access model
NO_PAYMENT	No existing payment could be found
INTERNAL_ERROR	A problem on the server meant that the request could not be processed

Method: **GetRemainingAccesses**

Description: Gets the number of accesses remaining in an already-paid-for number of accesses.

Inputs:

username	Your username
password	Your password
bango	Bango number
userId	User ID

Outputs:

responseCode	Result of the payment request (see below)
responseMessage	Any additional information to clarify the payment status code
accesses	Number of accesses remaining

Response Codes:

OK	Success
ACCESS_DENIED	Invalid username/password
ACCESS_DENIED	Invalid client IP address
ACCESS_DENIED	Invalid certificate
ACCESS_DENIED	Invalid access to specified Bango number
INVALID_BANGO	Invalid Bango number
INVALID_USERID	Invalid User ID
INVALID_ACCESSMODEL	Bango number does not have the correct access model
NO_PAYMENT	No existing payment could be found
INTERNAL_ERROR	A problem on the server meant that the request could not be processed

Method: **ConsumeOneAccess**

Description: Consumes one of the remaining accesses in an already-paid-for bundle of accesses.

Once all accesses have been consumed, the next attempt to consume an access will return the status code ACCESS_EXPIRED.

Inputs:

username	Your username
password	Your password
bango	Bango number
userId	User ID

Outputs:

responseCode	Result of the payment request (see below)
responseMessage	Any additional information to clarify the payment status code
accesses	Number of accesses remaining

Response Codes:

OK	Success
ACCESS_DENIED	Invalid username/password
ACCESS_DENIED	Invalid client IP address
ACCESS_DENIED	Invalid certificate
ACCESS_DENIED	Invalid access to specified Bango number
INVALID_BANGO	Invalid Bango number
INVALID_USERID	Invalid User ID
INVALID_ACCESSMODEL	Bango number does not have the correct access model
ACCESS_EXPIRED	No accesses remain
INTERNAL_ERROR	A problem on the server meant that the request could not be processed

API Usage Examples

The following examples are pseudo-code for illustration purposes only. Refer to your appropriate technology documentation for details of how to use web services in your chosen development environment.

Requesting a payment:

```
Long userId = Request("u")
String myBango = "12345"

DoPaymentRequest req
req.username = myUsername
req.password = myPassword
req.bango = myBango
req.userId = userId

DoPaymentResponse res = DIRECTBILLINGAPI.DoPayment(req)
If res.responseCode = OK Then
        // Successful payment
        Session("transactionId") = res.transactionId
Else
        // Deal with error
        Session("responseCode") = res.responseCode
        Session("responseMessage") = res.responseMessage
End If
```

Requesting a payment with price overrides:

```
Long userId = Request("u")
String myBango = "12345"

DoPaymentWithPriceRequest req
req.username = myUsername
req.password = myPassword
req.bango = myBango
req.userId = userId
req.priceList = new PriceList[] = { new Price(1.00, "GBP"), new Price(1.75, "USD"), new Price(2.50,
"EUR") }

DoPaymentWithPriceResponse res = DIRECTBILLINGAPI.DoPaymentWithPrice(req)
If res.responseCode = OK Then
        // Successful payment
        Session("transactionId") = res.transactionId
Else
        // Deal with error
        Session("responseCode") = res.responseCode
        Session("responseMessage") = res.responseMessage
End If
```

Creating a subscription:

```
Long userId = Request("u")
String myBango = "12345"

CreateSubscriptionRequest req
req.username = myUsername
req.password = myPassword
req.bango = myBango
req.userId = userId

CreateSubscriptionResponse res = DIRECTBILLINGAPI.CreateSubscription(req)
If res.responseCode = OK Then
        Session("userId") = userId
        Session("subscriptionId") = res.subscriptionId
        Session("transactionId") = res. transactionId
End If
```

Refunding a transaction to carrier bill:

```
Long transactionId = Request("transId")
String myBango = "12345"
String refundType = "OPERATOR"

DoRefundRequest req
req.username = myUsername
req.password = myPassword
req.bango = myBango
req.transactionId = transactionId
req.refundType = refundType

DoRefundResponse res = DIRECTBILLINGAPI.DoRefund(req)
If res.responseCode = OK Then
        // Successful refund
        Session("refundTransactionId") = res.refundTransactionId
Else
        // Deal with response
        Session("responseCode") = res.responseCode
        Session("responseMessage") = res.responseMessage
End If
```