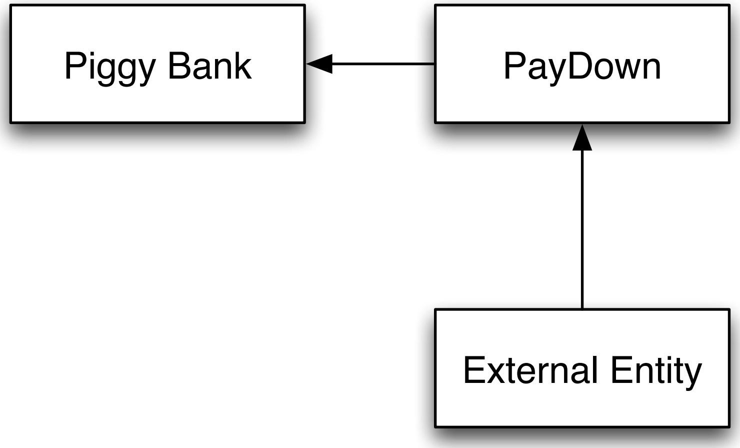
Prepaid Service Setup



An external entity, part of some non-billing system, wants to be billed for. The most common case, here, is a standard Pobox account. This also applies to domain registration and Listbox tier plans.

This also applies to current-style account plans; they are not metered, but prepaid, with the paydown object having a different rate and initial piggy bank size each cycle

1. When created, the entity tells the billing system to create a PayDown object for it.

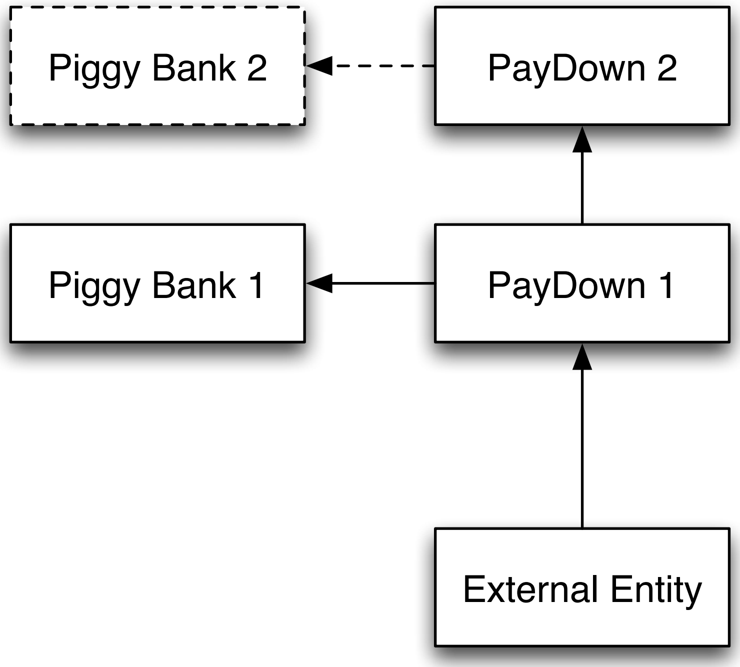
2. The newly-created PayDown object posts a charge to the ledger's current charge queue.

3. At some point, the charge queue is converted to an invoice and sent to the user.

4. When the invoice is paid, the funds for the PayDown item are deposited in a new piggy bank. The PayDown object is pointed at the Piggy Bank.

5. At regular intervals, the PayDown object posts charges to the ledger, to be payed by the piggy bank. These charges are internal, and are not displayed on customer invoices.

Prepaid Service Renewal



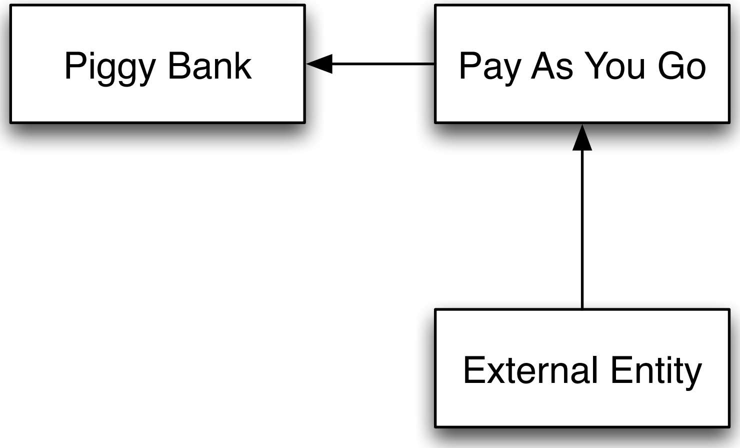
1. PayDown 1 regularly receives events that cause it to check how much longer its allotted piggy bank will last. When it's "not long enough," it issues as "spawn my replacement event."

2. The replacement for PayDown 1 is PayDown 2. It is (probably) for an identical time period and rate, but PayDown 1 may have some properties that cause its replacement to differ in some way.

3. PayDown 2 posts a charge to the ledger for its full amount. When paid, Piggy Bank 2 is created with that fund, and is targeted by PayDown 2. PayDown 2 sits inert.

4. When PayDown 1 is expired, it will look to see whether it has a replacement. If so, it will fire the "allow replacement to take over" event. If not, it will (possibly after some delay) fire the "expired without replacement event" which can do things like cause the external entity to be deactivated.

Pay as You Go Service Setup



An external entity, part of some non-billing system, wants to be billed for. The most common case, here, is a new-style metered mailing list or a Convio-style account.

1. When created, the entity tells the billing system to create a PAYG object for it.

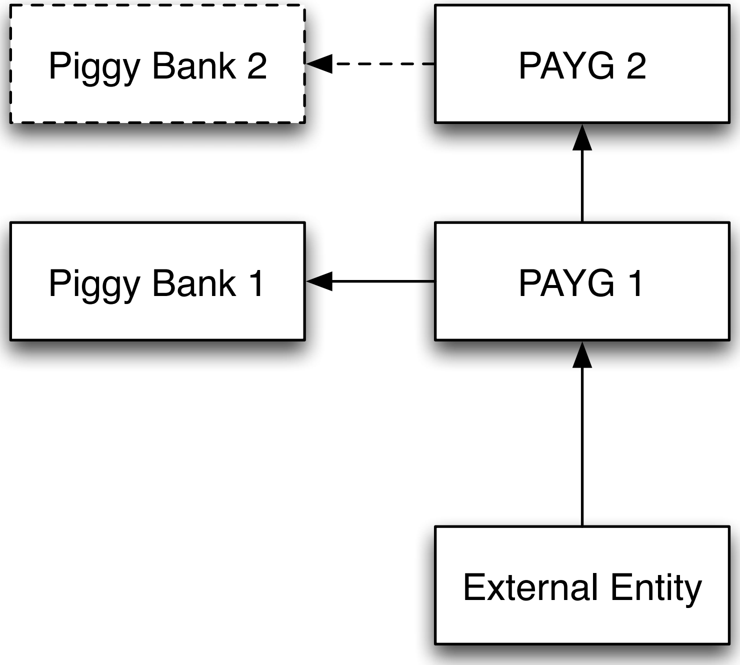
2. The user requests "buy delivery credit" and chooses an amount.

3. The PAYG object is instructed to issue an invoice-immediately charge to the ledger for the desired amount. The amount is also used to decide the rate schedule to be used when the PAYG charges the funds.

4. The user makes a payment. A piggy bank is created and the PAYG object is pointed at the piggy bank with the correct rate.

5. As needed, charges are posted to the PAYG object, which can draw down funds (microinvoices? reserved funds?) immediatley.

Pay as You Go Service Renewal



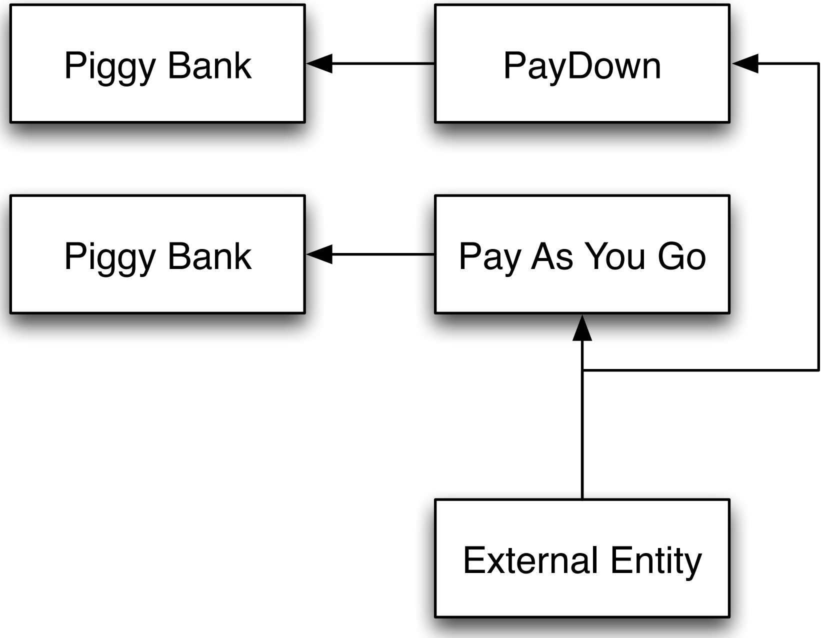
1. PAYG 1 regularly receives events that cause it to check how much longer its allotted piggy bank will last. When it's "not long enough," it issues as "spawn my replacement event."

2. The replacement for PAYG 1 is PAYG 2. It will request payment, as PAYG 1 did when being set up. When payment is made to set up PAYG 2's piggy bank, the amount paid will tell PAYG 2 how to pick its rate schedule.

3. When paid, Piggy Bank 2 is created with that fund, and is targeted by PAYG 2. PAYG 2 sits inert.

4. When PAYG 1 is expired, it will look to see whether it has a replacement. If so, it will fire the "allow replacement to take over" event. If not, it will (possibly after some delay) fire the "expired without replacement event" which can do things like cause the external entity to be deactivated.

Hybrid Payment Types



Listbox bulk plans could have two kinds of charges: a per-service-period flat fee and a metered pool against which deliveries are charged.

This is easy to model. We create a PayDown object for the yearly fee, and it follows the normal rules for PayDown objects. We also create a PAYG object with its own piggy bank. If the user uses up his delivery credits in a year, he can pay for a new PAYG setup without having to pay the annual surcharge.