Proposal for a new JMS Destination configuration

The proposal is organized as follows.

1. Use cases
2. Design concepts/notes
3. Configuration format with examples
4. Complete list of options available
5. Code patch (attached to JIRA)

Use cases

The following were requested by Qpid users via JIRA's and user list.

- Arbitrary exchange types (Ex XML exchange).
- Any kind of queue declare options (Ex. qpid.max-size, alt-exchange)
- Any kind of queue binding options
- Ability to support destination specific parameters like
  o msg credits, byte credits
  o sync-publish, sync-ack
  o whether a queue should be created/bound by producer side
- Bind a queue to multiple exchange/binding key pairs.

Design concepts/notes

- I have moved away from the previous URL format as it,
  - Does not clearly identify a resource, hence against the concept of a URL
  - It is impossible to fit all information in to a URL
- The new format is integrated alongside the old system with absolutely no change to existing way of doing things.
  A mix and match of both the old and new system could be used (if really needed).
- The new format takes ideas from the AMQP 1.0 spec.
  But it is not intended to support AMQP 1.0 when it comes out. If it ends up being a pre-cursor for supporting AMQP 1.0 it would just be a bonus.
- The new format clearly identifies the dual role of a javax.jms.Destination.
  That being the producer and consumer's view of a destination.
- The new format allows a way to support,
  - Arbitrary exchange types (Ex XML exchange).
  - Any kind of queue declare options (Ex. qpid.max-size, alt-exchange)
  - Any kind of queue binding options
  - Ability to support destination specific parameters like
    - msg credits, byte credits
    - sync-publish, sync-ack
    - whether a queue should be created/bound by producer side
  - Bind a queue to multiple exchange/binding key pairs.
- Define queues, links and then compose them to create destinations.
- Provides sensible defaults . At least I tried to 🙂.

Configuration format with examples

The new format consists of definitions for queues, publisher/consumer links and destinations in key/value pairs.

```
xqueue.<id> = name='value1'[;key2='value2';key3='value3'......]
pub.link.<id> = key1='value1';key2='value2';key3='value3'......
sub.link.<id> = key1='value1';key2='value2';key3='value3'......

xdestination.<jndiName> = queue='<id>'[;pub.link='<id>';sub.link='<id>']
```

Using queue, pub/sub links def's you can compose destinations.

Examples

In the simplest form

```
xqueue.myQueue = name='myQueue'
xdestination.myQueue = queue=myQueue
```

This is equivalent to the old queue = myQueue format.

Using qpid specific options and per destination switches

```
xqueue.tradeQueue1 = name='trade-queue1';durable='true'
xqueue.tradeQueue2 = name='trade-queue2';qpid.max_size='5000';qpid.policy_type='ring'

pub.link.trade1 = filter='amq.direct/tradeQueue1';sync-publish='all'
pub.link.trade2 = filter='amq.direct/tradeQueue2';create-queue='true'

sub.link.mylink = msg-credits='1000';byte-credits='1000';sync-ack='true'

xdestination.myLocalTrades = queue='tradeQueue1';pub.link='trade1';sub.link='myLink'

xdestination.myDailyTrades = queue='tradeQueue2';pub.link='trade2';sub.link='myLink'
```

Binding a queue to multiple exchange/routing key pairs

```
Using the above queue definition.

sub.link.multiLink = msg-credits='1000';bindings='{amq.topic/stocks.*};{amq.match//x-match='any',sym='RHT'}'

xdestination.myDailyTrades = queue='tradeQueue2';sub.link='multiLink'
```

Complete list of options

- xqueue
  - name : name of the queue
  - durable
  - exclusive
  - auto-delete
  - alt-exchange
  - no-local (??)
  - qpid.max_count
  - qpid.max_size
  - qpid.policy_type { reject | flow_to_disk | ring | ring_strict }
  - qpid.last_value_queue {1}
  - qpid.last_value_queue_no_browse {1}
  - qpid.LVQ_key
  - qpid.persist_last_node {1}
  - qpid.queue_event_generation { 0,1,2 } (0 to disable,1 to replicate, only enqueue events)
- sub.link
  - filter
  - filter-type
  - msg-credits
  - byte-credits
  - sync-ack
  - bindings - format as follows

    ```
    {exchange-name/bindingkey[/key=value,key=value,...};{....}..]
    ```
- pub.link
  - filter
  - filterType
  - sync-publish {persistent|all}
  - create-queue (producer side will declare/bind the queue)