In [1]:
```
import pandas as pd
store = pd.HDFStore('/Volumes/FreshBooks/data/store.h5')
may07 = store['may07']
```

If we want to get a graph over time like those you get from munin and graphite, we'll need to `resample()` our `DataFrame`:

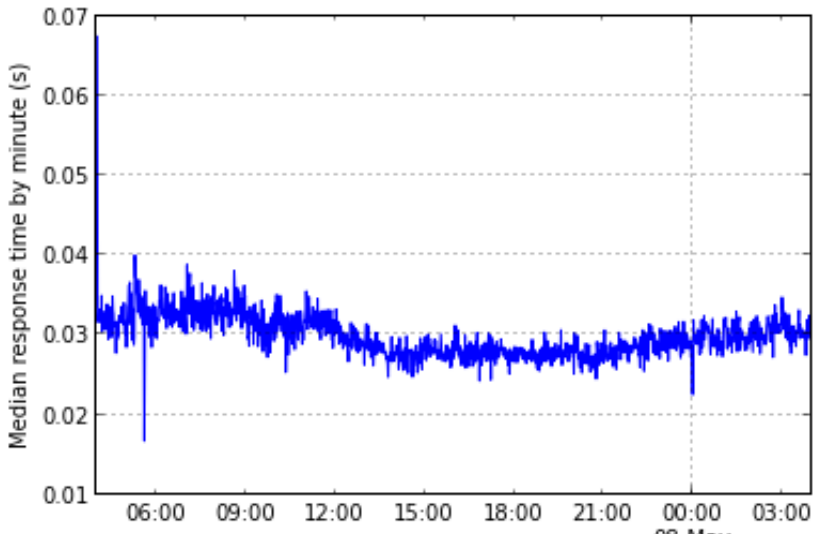
In [2]:
```
from datetime import timedelta
import numpy as np
first_minutes = may07[may07.index < (may07.index[0] + timedelta(minutes=10
first_minutes['elapsed'].resample('T', how=np.median)
```

Out[2]:
```
posix_timestamp
2012-05-07 04:00:00    0.067407
2012-05-07 04:01:00    0.033550
2012-05-07 04:02:00    0.032216
2012-05-07 04:03:00    0.031590
2012-05-07 04:04:00    0.032285
2012-05-07 04:05:00    0.033172
2012-05-07 04:06:00    0.032829
2012-05-07 04:07:00    0.031914
2012-05-07 04:08:00    0.031735
2012-05-07 04:09:00    0.034944
2012-05-07 04:10:00    0.030418
Freq: T
```

Then we can use the handy `DataFrame.plot()` method to get it displayed easily:

In [3]:
```
ylabel('Median response time by minute (s)')
may07['elapsed'].resample('T', how='median').plot()
```

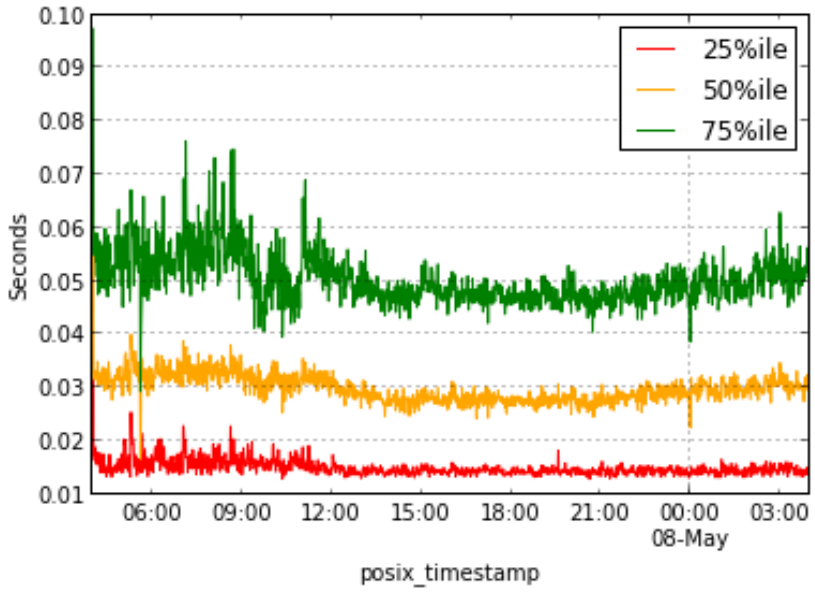
Out[3]:
```
<matplotlib.axes.AxesSubplot at 0x105809690>
```



We can use other callables to indicate how we'd like the downsampling to occur, such as a percentile function:

In [4]:
```
elapsed = may07['elapsed']
ylabel('Seconds')
# NB: style='b' kwarg is to work-around the already-fixed https://github.c
elapsed.resample('T', how=lambda x: np.percentile(x, 25)).plot(label="25%i
elapsed.resample('T', how=lambda x: np.percentile(x, 50)).plot(label="50%i
elapsed.resample('T', how=lambda x: np.percentile(x, 75)).plot(label="75%i
legend()
```

Out[4]: <matplotlib.legend.Legend at 0x118495ed0>



In [4]: