#@namespace scala com.twitter.servo.cache.thriftscala

#@ namespace strato com.twitter.servo.cache

// the java namespace is unused, but appeases the thrift Linter gods

namespace java com.twitter.servo.cache.thriftjava

enum CachedValueStatus {

FOUND = 0,

NOT\_FOUND = 1,

DELETED = 2,

SERIALIZATION\_FAILED = 3

DESERIALIZATION\_FAILED = 4,

EVICTED = 5,

DO\_NOT\_CACHE = 6

}

/\*\*

\* Caching metadata for an binary cache value

\*/

struct CachedValue {

1: optional binary value

// can be used to distinguish between deletion tombstones and not-found tombstones

2: CachedValueStatus status

// when was the cache value written

3: i64 cached\_at\_msec

// set if the cache was read through

4: optional i64 read\_through\_at\_msec

// set if the cache was written through

5: optional i64 written\_through\_at\_msec

// This optional field is only read when the CacheValueStatus is DO\_NOT\_CACHE.

// When CacheValueStatus is DO\_NOT\_CACHE and this field is not set, the key

// will not be cached without a time limit. If the client wants to cache

// immediately, they would not set DO\_NOT\_CACHE.

6: optional i64 do\_not\_cache\_until\_msec

// Indicates how many times we've successfully checked

// the cached value against the backing store. Should be initially set to 0.

// The client may choose to increase the soft TTL duration based on this value.

// See http://go/gd-dynamic-cache-ttls and http://go/strato-progressive-ttls for some use cases

7: optional i16 soft\_ttl\_step

} (persisted='true')