Xenstore API

Maxim Kharchenko, Cloudozer LLP

07/07/2014

1 Overview

Xenstore is a simple configuration database for Xen domains. Driver frontends and backends use Xenstore to negotiate page grants, (virtual) interrupts, and statuses. In this respect Xenstore operates as a software bus. Hence, the synonym for Xenstore - XenBus.

LING uses Xenstore extensively during its initialisation sequence. Now the complete interface to Xenstore is accessible at Erlang level. The Erlang interface to Xenstore uses internally a new port type to make the implementation fully asynchronous.

2 Xenstore API

All intefaces calls belong to the new eponymous module added to the kernel application.

2.1 Simple read/writes

```
xenstore:read(Key)              -> {ok,Value} | {error,_}
xenstore:read_integer(Key)      -> {ok,Value} | {error,_}
xenstore:write(Key, Value)      -> ok | {error,_}
xenstore:write(Key, Value, Tid) -> ok | {error,_}
```

`Key` must be a string. `Value` can be either a string or an integer. `Tid` is a transaction id (see below).

Example of the calls that will succeed for any domain:

```
xenstore:read("name")       %% name of the current Xen domain
xenstore:read("vm")         %% path to Xen domain info, such as the command line
```

2.2 Path operations

```
xenstore:list(Path)         -> {ok,Keys} | {error,_}
xenstore:mkdir(Path)        -> ok | {error,_}
xenstore:mkdir(Path, Tid)   -> ok | {error,_}
xenstore:delete(Path)       -> ok | {error,_}
xenstore:delete(Path, Tid)  -> ok | {error,_}
```

Any Xenstore key can have an associated value and the list of nested keys. There is little distinction between keys and paths. The list of nested keys can be retrieved using `xenstore:list(Path)`. `Path` must be a string. Similar to writes, path modifications can happen in the context of a transaction.

2.3 Permissions

```
xenstore:get_perms(Path)
xenstore:set_perms(Path, Perms)
```

It is possible to set permissions on Xenstore keys (or paths) using `xenstore:set_perms(Path, Perms)`. `Perms` must be a lists of strings. Each string is a concatenation of a character and a domain number, e.g. "r24".

Character	Meaning
r	read
w	write
b	read/write
n	none

The first string in the permission list is special. It encodes the owner of the key (or path) and the default permission for other domains. For example, suppose the current domain id is 24. To allow all domains to read `Key`, the first string in the list should be "r24".

Most of keys in Xenstore are owned by Dom0 and thus their permissions can not be relaxed. The notable exception is the (relative) path "data". Any domain can set arbitrary permission on this path and its descendants.

3 Watches

```
xenstore:watch(Path)
xenstore:unwatch(Path)
```

Watches is the most valuable feature of Xenstore. It allows the Erlang code to get immediate notifications when a given path (or its children) are changed, deleted, or set

permissions on. The noritications are delivered as messages to the process that called `xenstore:watch()`. The domain must have at least read permission to a path to watch it.

4 Transactions

```
xenstore:transaction()
xenstore:commit(Tid)
xenstore:rollback(Tid)
```

Xenstore supports simple transactions. It is possible to wrap a series of `write/mkdir/set_perms/delete` calls into `transaction/commit` to achieve atomicity. There are transactional and non-transactional versions of all modifying operations.

5 Miscellaneous

There is a utility function `xenstore:domid()` that returns the domain id of the current domain.

6 Technical details

See txt[1].

[1] http://xenbits.xen.org/docs/4.3-testing/misc/xenstore.txt