# openQA Project - coordination #14626

# [epic] backend and console capabilities interface to increase extensibility and code reuse

2016-11-03 13:28 - rpalethorpe

| | | | |
|---|---|---|---|
| **Status:** | New | **Start date:** | 2019-02-17 |
| **Priority:** | Normal | **Due date:** | |
| **Assignee:** | | **% Done:** | 17% |
| **Category:** | Feature requests | **Estimated time:** | 0.00 hour |
| **Target version:** | future | | |
| **Difficulty:** | | | |

**Description**

## Motivation

Prevent "if/else" in tests needing to distinguish different backends

## Acceptance criteria

- **AC1:** No obvious "if/else" for different types of consoles in os-autoinst-distri-opensuse are necessary anymore
- **AC2:** Same as *AC1* for different *backends*

## Suggestions

- Read what had been done in https://github.com/os-autoinst/os-autoinst/pull/1232 and https://github.com/os-autoinst/os-autoinst-distri-opensuse/pull/8718 to define "persistent" consoles
- Incorporate content from https://github.com/os-autoinst/os-autoinst-distri-opensuse/blob/master/lib/Utils/Backends.pm into os-autoinst as flags on backends rather than if/else in test code
- Look for other "if/else" code in test distributions, e.g. os-autoinst-distri-opensuse", distinguishing different backends and consoles to provide as capabilities on backends/consoles

## Further details

### Background

In an ideal world all the backends (QEMU, bare metal, Xen) and consoles (VNC, serial or hybrid) would be accessed in a uniform manner by testapi so that the distribution and test writers could write their test case once and then have it run across all available platforms without modification. In practice however different Operating systems, hardware, hypervisors and console combinations differ significantly enough in behaviour that a completely uniform API is not possible without either significantly disadvantaging some platforms or providing support for edge cases in the distribution itself.

While the functions in testapi can be kept mostly uniform in availability and behaviour it requires that the distribution handles changes in the Operating System's behaviour due to the machine (virtual or physical) which it is running on and what user interface (console) is selected. Many things can be abstracted away into the console or backend classes in os-autoinst, however OS specific behaviour can not be without making os-autoinst specific to one type of OS or even Linux distribution. Currently the SUSE os-autoinst distribution handles differences between backends by reading variables to determine which backend or architecture is being used in a variety of different places and performing some particular action for that backend. Unfortunately this doesn't just happen in (suse)distribution.pm or other modules in the lib folder, but throughout the test cases.

The problem with branching on a particular architecture or backend is that the contents of the branch statement may actually apply to a whole class of backends not just one. Thus by restricting it to one particular backend you have missed an opportunity to maximise the benefit of your code, which will lead to duplication of effort. However in some cases it may be wasted effort to try inventing general abstractions when they will only be used in one or two instances, but then that is a universal problem, we just have to make a judgement on each and every case.

### Proposal

At any rate my proposal is to introduce the notion of capabilities which can apply to consoles or backends. Any console or backend should have to declare its capabilities in a standard way which can then be read by the distribution and in some very rare cases, the distribution's test modules. Capabilities should be validated against a central list in the appropriate base class, attempting to access or set a capability which does not exist should be an error. This should make them better structured than simply adding more global variables which already serve this purpose to some extent. A list of quirks could also be maintained to indicate negative platform attributes.

The actual implementation could be done using a Perl map, object mixins from some Perl OO library or something else.

## Further rambling

In the case of the serial terminal feature, I would remove the new testapi function I have added called is_serial_terminal and instead replace it with one or more console/backend capabilities. Perhaps something like direct_read_text and direct_write_text which indicates to the distribution that we are reading and writing raw text to the terminal, the backend would of course require a serial port capability to activate the console. The Linux VNC text console would have something like redirect_write_text which indicates we can redirect output to the serial port and some other capabilities to indicate that we can use needles and send key presses. Any console on some other OS/hardware combo which doesn't support serial ports will be missing the capabilities which indicate we can do this so either run_script and wait_serial will return an error indicating the missing capability or the distribution will have to implement the testapi functions using some other capabilities or workarounds.

Along with having capabilities comes the idea of having interfaces to take advantage of them, so that a backend, console and distribution with compatible capabilities can be plugged together. Such interfaces and their associated capabilities can be invented and implemented on a rolling basis rather than attempting to do some massive overhaul of the code base. This may slow down feature development for some time, but will eventually speed it up and creates a basis for a backend/console plugin architecture. I am willing to implement this in so far that it is required for #14582 and other platform specific features I think that the LTP, and other test suites I may work with, can take advantage of.

## Alternatives

The alternatives are to forgo taking advantage of platform specific features or add the occasional function to the testapi like is_serial_terminal and use the existing vars mechanism. At least for what I am currently doing, the latter choice is acceptable to me, but it is not extensible beyond a point. There is also the console proxy feature which allows you to tightly couple your test module to a particular console implementation completely bypassing all layers of abstraction while at the same time obfuscating the code flow using Perl meta programming which should be avoided at least from tests perspective.

## Problems

It is more difficult to identify and isolate a class of behaviour shared by multiple entities and create an abstraction to encapsulate it than just to write code for a specific case. Sometimes people may attempt to create capabilities when there is no significant advantage to doing so or they may be tempted not to when there clearly is an advantage. The feature will need documenting and require effort on the part of reviewers to learn it and enforce its use. It will probably increase the codebases complexity initially until it has been reasonably taken advantage of. There is the danger of an explosion in capabilities which makes it difficult to write a new distribution which covers multiple platforms without understanding a large number of them. Regressions may be introduced while moving backends and consoles over to this system.

**Subtasks:**

| | |
|---|---|
| coordination # 38819: [qe-core][tools][functional][epic] Refactor use of backends | **Workable** |
| action # 33388: [functional][u][easy][pvm] Implement proper split from other backends | **Resolved** |
| action # 67417: Remote backend capability | **New** |
| action # 67420: Persistent console console capability | **New** |
| action # 67423: Persistent console backend capability | **New** |
| action # 67426: Raw text backend capability | **New** |
| action # 67429: Raw text console capability | **New** |

**Related issues:**

| | | |
|---|---|---|
| Related to openQA Project - action #14582: Add virtio serial console backend ... | **Resolved** | **2016-10-31** |
| Related to openQA Project - action #38486: [functional][u] add capability fla... | **Rejected** | **2018-07-17** |
| Related to openQA Tests - action #57329: [kernel][functional] PowerVM console... | **Resolved** | **2019-09-25** |

## History

**#1 - 2016-11-03 13:29 - rpalethorpe**

*- Related to action #14582: Add virtio serial console backend and API added*

**#2 - 2016-11-16 10:27 - rpalethorpe**

*- Priority changed from High to Normal*

**#3 - 2017-07-12 11:12 - rpalethorpe**

*- Assignee deleted (rpalethorpe)*

**#4 - 2017-11-18 16:10 - coolo**

*- Target version set to Ready*

this sounds like a good entry level issue

**#5 - 2019-06-20 15:29 - okurz**

*- Category changed from 132 to Feature requests*

**#6 - 2019-12-10 10:24 - okurz**

*- Related to action #38486: [functional][u] add capability flags to os-autoinst backends (or tests) added*

**#7 - 2019-12-10 11:40 - okurz**

*- Description updated*

**#8 - 2019-12-10 11:41 - okurz**

*- Related to action #57329: [kernel][functional] PowerVM console is not active all the time added*

**#9 - 2019-12-10 11:44 - okurz**

*- Subject changed from Backend and console capabilities interface to increase extensibility and code reuse to [epic] backend and console capabilities interface to increase extensibility and code reuse*

*- Description updated*

*- Status changed from New to Workable*

**#10 - 2020-05-27 11:54 - cdywan**

*- Status changed from Workable to In Progress*

*- Assignee set to cdywan*

*- Target version changed from Ready to Current Sprint*

**#11 - 2020-05-28 07:33 - okurz**

please keep in mind that this ticket is an "epic" so the tasks at hand are to create subtickets as specific user stories, not write any code

**#12 - 2020-05-28 16:21 - cdywan**

*- Due date set to 2020-05-28*

due to changes in a related task: #67417

**#13 - 2020-05-28 16:22 - cdywan**

*- Due date set to 2020-05-28*

due to changes in a related task: #67420

**#14 - 2020-05-28 16:24 - cdywan**

*- Due date set to 2020-05-28*

due to changes in a related task: #67423

**#15 - 2020-05-28 16:26 - cdywan**

*- Due date set to 2020-05-28*

due to changes in a related task: #67426

**#16 - 2020-05-28 16:28 - cdywan**

*- Due date set to 2020-05-28*

due to changes in a related task: #67429

**#17 - 2020-05-29 08:47 - rpalethorpe**

Added some members of the kernel qa team as watchers. Changing things like is_serial_terminal could have big effect on the kernel tests and baremetal tests.

**#18 - 2020-06-22 10:35 - cdywan**

*- Status changed from In Progress to Blocked*

*- Assignee deleted (cdywan)*

**#19 - 2020-06-26 10:07 - okurz**

*- Assignee set to okurz*

**#20 - 2020-07-30 18:25 - okurz**

*- Status changed from Blocked to Workable*

*- Assignee deleted (okurz)*

*- Target version changed from Current Sprint to future*

Left for community :)

**#21 - 2020-10-12 13:38 - szarate**

*- Tracker changed from action to coordination*

*- Status changed from Workable to New*

**#22 - 2020-10-12 13:47 - szarate**

See for the reason of tracker change: http://mailman.suse.de/mailman/private/qa-sle/2020-October/002722.html