Hotspot Command-line Flags: Kinds, Lifecycle and the CSR Process

Kinds

Hotspot defines 6 kinds of command-line flags (all starting with -XX):

- **product**
  These are the primary flags we intend end-users to apply for tuning, feature selection etc. These are settable via the command-line in product builds.
- **manageable**
  These are a special kind of product flag that can be dynamically set via the JDK management interface.
- **develop**
  These are flags intended for use by Hotspot developers for debugging and testing purposes. These are settable only in non-product builds, and ha ve a constant value in product builds.
- **not_product**
  A more constrained development flag that has no presence at all in a product build.
- **experimental**
  These flags are in support of features that are not part of the officially supported product, but are available in the product for experimenting with. They must be explicitly unlocked to be used. They are expected to be temporary in nature, and potentially replaced by a product flag in a later release if the feature remains. Experimental features can be dropped with no notice.
- **diagnostic**
  Diagnostic flags are not meant for VM tuning or for product modes. They are to be used for VM quality assurance or field diagnosis of VM bugs. T hey are hidden so that users will not be encouraged to try them as if they were product flags. However, they are available in the product version of the VM so they can be enabled under instruction to collect diagnostic information about VM problems.

Lifecycle

The Hotspot team have a defined (up to) three step deprecation process for removing flags:

1. Deprecate: accept the flag, act on it, but warn it is deprecated and may be removed in the future.
2. Obsolete: accept the flag but ignore it other than to issue a warning it is obsolete and may be removed in the future. This allows code associated with the flag to be removed from the codebase.
3. Expired: use of the flag causes an "unknown VM option" error and the VM refuses to start.

The full three step process is used for "product" and "manageable" flags. For the other flags we can start at step 2 and "obsolete" the flag in the current release.

The CSR Process

The Compatibility, Specification and Review (CSR) process ensures that all changes to exported interfaces are reviewed in depth to ensure they are well designed, and that compatibility concerns have been considered. These exported interfaces include the various flags and options that can be passed to the different command-line tools in the OpenJDK (java, javac, jlink, etc.)

For CSR purposes, only Hotspot's "product" and "manageable" flags are considered exported interfaces that require a CSR request when they are added, or commence the deprecation process.

For the other kinds of flags, a CSR request is not required, and it is up to the developer whether removal needs to follow (part of) the deprecation process.

If a flag changes its kind to product/manageable then that is the same as adding a new product/manageable flag and so needs a CSR request. If a flag changes it kind from product/manageable then that is the same as removing a product/manageable flag, and so also needs a CSR request, and likely a transitional "deprecation" process.

A CSR request is only needed at the commencement of the deprecation process for a product/manageable flag, not for each step.

A CSR request can always be made, even if not required, if there is some compatibility concern over the change being made.