**Haskell Foundation Stability Working Group**

Charter, December 2021

## 1. Purpose of the working group

Haskell serves (at least) two roles:

1. As a production language used by thousands of developers to get their day job done.
2. As a research language, exploring the bleeding edge of statically typed, purely functional programming.

The breaking changes necessary for (B) impose substantial costs on (A), so there is a real tension between these two goals. This tension can leave group (A) frustrated as an apparently endless stream of changes; and group (B) frustrated at the brake on progress that (A) imposes.

And yet many of the features developed by (B) are wildly popular with (A). No one is advocating freezing Haskell, halting all further language development. Indeed most of the proposals considered by the GHC Steering Committee (over 400 of them in the last three years) come from industry not academia.

This tension cannot be removed entirely. The goal of the working group is to mitigate its worst consequences, principally by improving communication between the various parties involved so that everyone’s expectations are aligned.

One alternative would be to move to a policy of 100% stability, especially of the language itself and its APIs; that is, no breaking changes. There are other language ecosystems that do this. But the Haskell Foundation believes that would be a mistake to seek 100% stability, a mistake that would kill off the ferment of intellectual change, deep research, and innovative (and genuinely useful) features that characterise the Haskell ecosystem. Haskell is a thought leader, and must continue to grow and change.

**So then the question becomes one of how to manage change, and mitigate its costs. That is what the working group seeks to do.**

## 2. Members

The Stability Working Group currently has the following members:

* HF
  + Simon Peyton Jones
  + Alexander Bernauer
  + Chris Dornan
  + Andrew B (Exec Director)
* Ecosystem
  + CLC: Bodigrim (accepted)
  + CLC: Oleg Grenus (accepted)
  + GHC: Ben Gamari (accepted)
  + Stackage, Hackage: Adam Bergmark (accepted) (also aeson)
  + Cabal: Mikolaj Konarski (accepted)
  + HLS, Meta: Pepe Iborra (accepted)
  + head.hackage: Ryan Scott (accepted,but cannot attend calls)
* Library authors
  + Dmitrii Kovanikov (accepted) (package maintainer -- sent an email to the board)
  + Tom Sydney Kerckhove (invited) (developer -- sent an email to the board)
  + Viktor Dukhovni (accepted) ([GitHub home page](https://github.com/vdukhovni)) ([live bytestring patch](https://github.com/haskell/bytestring/pull/438))
* Consultants
  + Trevis Elser (accepted) (consultant)
* Industrial users
  + Neil Mitchell (accepted) (Meta, hoogle, hlint, shake -- weighed in on Twitter)

## 3. Plan

* Talk to the stakeholders and determine their priorities.
* Define our initial scope — what the core ecosystem is.
* Develop a plan that will, we hope, help to mitigate the tension between stability and change.
* As a first step in this plan, to develop, in consultation with relevant stakeholders, a clear Managing Change policy. The policy might cover
  + The kinds of breakage we can expect to see
  + When breakage can occur
  + What ‘tools’ and guidance should be expected with each breaking change. (Here we mean tool in the most general sense, ranging from a detailed explanation of the change to an actual tool to help with migration.)

Here are some ideas of the kind of things that might be in that plan:

* Be clear about what constitutes a “breaking change”. Some changes are much, much more disruptive than others, and it may help not to treat them all identically. (See e.g. [this thread](https://discourse.haskell.org/t/is-adding-hascallstack-a-breaking-change/3696).)
* To gather and record all scheduled breaking changes to the core ecosystem on an issue tracker
* To see what we can do to derive a schedule that satisfies all of the stakeholders — entirely voluntarily; everybody functions as before, just hopefully with more information.
* To communicate the schedule and what we are doing to the wider community in order to minimise 'surprises'.
* On the basis of what we find, propose any extra mechanisms that would make all of this easier.
* Another straw-man from SPJ. A new release of GHC often leads to a wave of trivial-but-necessary package upgrades (e.g. a new export from package X forces a ‘hiding’ clause in package Y, and hence a minor version bump; that in turn forces package Z to widen its version bounds. etc). The GHC team (and friends) already implement these changes as an overlay in head.hackage, to allow testing of GHC. Idea: on an opt-in basis invite package authors to pre-approve these changes, so they can all be implemented on a single day, alongside the GHC release, without being bottlenecked on the serial availability of package authors.