Code Point Management
IETF 97

Jeffrey Haas <jhaas@juniper.net>

Introduction

- This presentation is based on an internal presentation given at my employer regarding the issue of doing protocol development work.
- To set the expectations, the talk is subtitled: “Stop making customers regret early deployment.”

Motivation for this Talk

- Code point allocation and use have impact when people are trying to deploy protocols.
- Improper allocation practices in code and in specifications cause issues for implementers and users of protocols.
  - Sometimes these issues are local; e.g. hop-by-hop or constrained to a given internal network.
  - Sometimes they are terribly global; e.g. BGP or DNS.
  - The “blast radius” of doing something wrong is a strong motivator to try to do it right the first time.

Code Points

- Protocols need code points in order to operate.
- When it is a completely new protocol operating disjointly with everything else, the specification authors can usually make up their own code point namespaces and choose initial assignments without concern.
- Once a protocol has been deployed, maintenance of the code points requires coordination. While it is possible that these code points are solely maintained as part of specification revisions, typically a *registry* is created somewhere to coordinate the assignment and publication of code points.
  - For IETF, that registry is often IANA.

Registration Policies in IETF

- RFC 5226, Guidelines for Writing an IANA Considerations Section in RFCs, provides a number of common policies for registries:
  - Private Use: Not regulated at all. Use at peril.
  - Experimental Use: Used for experimentation.
  - Hierarchical Allocation: Permits delegation of ranges to organizations.
  - First Come, First Served: Anyone can ask for a code point.
  - Expert Review: Requires input from the Designated Expert.
  - Specification Required: The document *might not* be an RFC.
  - RFC Required: Requires an RFC.
  - IETF Review: Requires the document to work its way through IETF review process, either as an Area Director sponsored document or as a Working Group document.
  - Standards Action: Requires an RFC targeted for the Standards Track.
  - IESG Approval: Catch-all of last resort.

Why are we Having this Conversation?

- As ~~developers~~ protocol authors, when it becomes necessary to write code for a protocol feature that requires a new code point, one must be chosen.
  - The question is *what* to choose.
  - Your choices have far-reaching impact on the deployability of a new feature.

If You Pick One, and You're Wrong…

- A common strategy for a public registry that has sequentially allocated code points is to pick the next one.
  - The feature ships, and customers start deploying it.
  - And potentially another implementer does the same thing for a totally different feature.
    - Now things can't interoperate.
    - Who wins?
  - If you're lucky, you might "publicly squat" upon an undelegated code point as part of the standards document process.

… Examples of When You're Wrong

- draft-snijders-idr-deprecate-30-31-129-01
- BGP Large Communities was assigned the next code unallocated BGP Path point by IANA, 30. It was in use by code that had shipped early development of other features… and the feature was not globally deployable in the Internet.
- Interesting note: Code points 30 and 129 were both done during IETF Hackathons.

Or You Pick One and Ship a Pre-RFC Feature

- As part of development work, you use a code point and at a later point change the behavior of the feature-in-progress.
  - This happened with the BGP Independent Domain feature. (RFC 6368) (also known as ATTR-128)
  - It was developed as a 2-byte Autonomous System Number feature and later updated to a 4-byte Autonomous System feature without using a new code point.
  - This implicit incompatibility resulted in multiple globally visible BGP Internet routing outages.
  - Partially as a result of these issues, this feature and its code point is considered *toxic.*
- *FCFS allocation isn't a guarantee of safety. Stability of an implementation (and thus versioning of the protocol component) is important.*

Picking a Code Point

- Many IETF registries are fairly permissive, having First Come First Served policies for a portion of the range. Once a feature is far enough along initial development and has made some progress through the standards process (if applicable), an allocation should simply be requested and any standards documents updated.
- Several of the more restrictive policies require the document to have reached the RFC publication point of the process. This can lead to bootstrapping issues since interoperable code is often needed and that requires coordinated code points.
  - *Early allocation*, covered in 5226bis, permits the IESG to “temporarily” allocate a code point to documents that have these dependencies.
  - These allocations have a one year life span and thus tend to come along much later in the life of a document. In practice, it can be longer, but it’s not permanent.
  - This is all partially complicated because the RFC process is slow.

What to Do Until You Can Get a Real Assignment

- Many registries provide for an Experimental range if they don't already have a Private range.
- You can use these as we wish.
  - But this means you must coordinate these assignments internally.
  - Experimental ranges tend to be small. This means that we can't have too many "experiments" running at the same time.
  - We have to have a migration strategy to move to a real assignment at a later time.
  - ***You can't ship your code using this experimental code point.***
  - A later presentation in IDR discusses a way we can potentially do more experiments.
- If there we can't get a FCFS or use an experimental point, we have to "squat" on a number.
  - And it becomes even more important to have a migration strategy.
  - ***And you still can't ship your code using this code point.***

Code Point Migration Life Cycle

- For feature development, it's most important to simply get a number. This number serves for development and testing purposes.
  - Try to use an appropriate number, ideally an experimental number until your work is far enough along.
  - If the feature must squat upon a code point including an experimental number, the feature *must* require configuration of the code points in order to function.
  - While it's tempting to provide configuration to change from a default number, defaults end up being wrong.
  - It is *critical* to not ship features that may use wrong code points.
  - The test group must sign up to verify the feature with its properly assigned code point and not just the one controlled by the configuration.
- **It is *dangerous* to ship features with configurable code points.**
  - **It lets the customer squat on arbitrary code points.**

Registry Policy Considerations

- AKA ”What do *I* put in my protocol spec?”
- A ”libertarian” policy would be FCFS as much as possible and Standards Action when needed – but mostly because Early Allocation isn’t a boundary to moving forward.
  - Make it easier to get a real code point than to squat on one.
- Making it easy also entails having enough code points to be generous with giving them out. Don’t be stingy with bits in your PDU unless you really need to be.

Review Considerations – in Code

- Code points are often scattered across general feature header files.
- Code points managed by registries should be refactored into separate files.
  - These files should be under expert reviewer approval control, preferably parties that also participate in the standards process.
  - A process for ensuring releases block when code points aren't properly allocated must be implemented. For example, a blocker PR.
- Your expert reviewers should be able to help developers choose code points at the time of design specification work.
  - And they should move for early allocation, if needed, when the feature is suitably stable.

Review Considerations – In Specifications

- Never create a type field without also creating a registry for it.
- When you review a draft, especially at WG adoption, review it carefully to see if there are type fields that have no registries.
- Registry policies SHOULD be FCFS or Standards Action unless there's a very good reason to not do so.
  - The WG should own the registry as much as is reasonable.
  - Policy speed bumps cause rash behaviors. See EVPN route type allocation issues as an example.
- Refer to code points by feature name rather than code point number.
- IANA is incredibly easy and pleasant to deal with. FCFS allocations are very fast.
- Document Shepherd work should include verifying registries as part of document progression to IESG.