Interversion Compatibility

We can choose to promise not to do any of these:

- ever
- except in the transition from DSTU to Production (which requires a way to distinguish between)
- except between DSTU releases and between DSTU and production (which requires a way to distinguish between)
- except when we decide a fundamental breaking change is required (which requires a way to distinguish between)

I've documented the changes that apply to profiles as well, though the primary concern is what can happen to resources. Deprecating and replacing profiles is less of a concern, so feel free to ignore those bits that are marked as specific to profiles.

Fundamental changes:

- New version of Atom
- Changes to what we expose in atom
- Introduction of additional XML attributes (or changing names of existing XML attributes)
- Changing the fundamental structure of FHIR elements (e.g. narrative, contained, extensions) - name changes, structure changes
- Introducing mixed text
- Introducing additional namespaces
- Introducing changes to how XML equivalents get expressed in JSON

Non-backward compatible (Changes that break constraints on an old version) (These will most commonly occur when there's an expansion of the use-case or recognition we'd been overly restrictive)

- Adding new elements (includes promotion of extensions to elements)
- Changing minOccurs from 1 to 0
- Changing maxOccurs from 1 to *
- Adding a new "code" in a required, non-extensible code list (whether in an enumerated list or expression-based value list)
- Changing a value set/code list from required to preferred or example or removing it entirely
- Changing a value set/code list from non-extensible to extensible
- Converting a singleton datatype and turning it into a choice of data types
- Adding additional choices to an existing choice data type
- Allowing a new resource as an allowed type for a resource reference
- Loosening or removing an ad-hoc constraint
- Increasing or removing maximum length
- Changing an element to be "must understand"
- Changing an element to be "required" (profiles only)
- Removing or loosening a profile on a resource or data type reference (profiles only)

Non-forward compatible (Introducing constraints not satisfied by old version) (These will most commonly occur when there's a change to the understanding of the 80% or a determination the existing resource doesn't work)

- Removing elements (usually demoting to extensions)
- Changing minOccurs from 1 to 0
- Changing maxOccurs from * to 1
- Removing a code from a required, non-extensible code list (whether in an enumerated list or expression-based value list)
- Declaring a required value set/code list where there wasn't one previously
- Declaring a previously extensible value set/code list to be non-extensible
- Converting a choice data type back to a singleton data type
- Removing choices from a choice data type
- Removing a resource from the list of allowed types for a resource reference
- Adding or tightening an ad-hoc constraint
- Declaring or decreasing maximum length
- Changing an element to no longer be "must understand"