MSC1425: Room Versioning

Problem/Background

It's likely that in the immediate future we'll want to change the properties of rooms in a way that will not be compatible with existing servers - for example, changing the rules for event auth or state resolution, or changing the format of an event id.

Typically we want to ensure that all servers participating in the room understand and implement the new rules, to avoid developing a split-brain situation.

This is a topic which has been covered a number of times before[1] and many approaches have been discussed. Here we aim to solve a simplified subset of the problem, by assuming that the version of a given room is fixed for the lifetime of that room. **A future MSC will address mechanisms to improve the experience of upgrading existing rooms.**

Proposal

Here, changes to the room model only apply to newly-created rooms; old rooms continue to exist and follow the old rules until they are manually shut down at some future point in the way that !cURbaf[2] was (writing a PSA in the room, making it read-only via PLs, and repointing the alias at the new room).

Mechanics for implementing this are:

- The server keeps track of the "room version" for each room. This is set to 1 for existing rooms; a configuration option (in homeserver.yaml or equivalent) sets the default for newly created rooms.
- We also extend the `/_matrix/client/r0/createRoom` API to take a `room_version` parameter, which can override the server's default room version, for the purpose of testing new versions.

  If the server does not recognise the given version, it should reject the request with a 400 error with the body:

---

[1] https://docs.google.com/document/d/1L2cr8djdpOFXJgqGTf3gUrk-YGBYf--rP8Nw6mYYAu8, https://docs.google.com/document/d/1UqcfS72qh8V7iXJ4oEkMIQ_E63xybRkpLxMEWw-EHBo (several iterations), https://docs.google.com/document/d/1_N9HhXEqO9yX1c4TSlVAAvTaiyzDXTuVmGW-3hJe840

[2] for the uninitiated: !cURbaf is "Old matrix HQ", aka `!cURbafjkfsMDVwdRDQ:matrix.org`

```
{
    "errcode": "M_UNSUPPORTED_ROOM_VERSION",
    "error": "Your homeserver does not support this room version",
}
```

- When a room is created, its version is recorded in the `m.room.create` event, in a `room_version` property.
- `federation/v1/make_join` includes (multiple)[3] `ver` query parameters, which lists the room versions which the joining server supports:

  `GET /_matrix/federation/v1/make_join/!room:id/@user:id?ver=1&ver=2`

- `make_join` requests which do not include the current version of the room are rejected with a 400 error with the body:

  ```
  {
      "errcode": "M_INCOMPATIBLE_ROOM_VERSION",
      "error": "Your homeserver does not support the features required
  to join this room",
      "room_version": "2"
  }
  ```

  This error code should be passed through to clients, so that they can give sensible feedback to the user.

- ~~We extend the response from `make_join` to include a `room_version` property, which tells the joining server the version to apply. (It does not yet have the `m.room.create` event, so cannot use that).~~

Grammar for room_version

`room_version`, in the `m.room.create` event and elsewhere, is a string. The length of a version identifier must not exceed 32 codepoints[4].

`room_versions` using only the digits `0-9` and the period `.` are reserved for use by future versions of the Matrix protocol. Version strings with alphabetic characters may be used for

[3] There is precedent (eg /join's server_name param) for repeating keys rather than something like ver=1,2,3.
[4] 32 codepoints is chosen as a balance between allowing room for experimentation and keeping the length of the `make_join` query string within reasonable bounds.

experimental changes. It is recommended that version strings are limited to the characters `a-z`, `0-9`, `.` and `-`.

Servers should not attempt to parse version strings, but should instead treat them as opaque identifiers.

Possible future refinements

[None of the following has yet been implemented]

- Prevent sending room invites to people whose HSes don't support the room version, with support via `/_matrix/federation/v1/invite`
- Include the room version in `directory` responses, to avoid future disappointment when trying to join a newly-versioned room.

Notes on client support

The proposal above focuses on server-side support; however it is also likely that changes to the way rooms work will also require changes in clients. Possibilities include:

1. Existing clients are able to fully interact with new-versioned rooms without any changes. Examples might include changes to the state resolution algorithm: since clients play no part in this process, no changes are required.
2. Existing clients are *almost* able to interact with new-versioned rooms as normal, but small areas of functionality will not work correctly.

   For example, if we changed the way that user bans work in a room so that they are stored in a separate `m.room.ban` event (rather than being in the `m.room.member` event as today). Here, clients would be able to join and leave, and send and receive messages, as normal, but they might not correctly show users being banned, or the list of currently-banned users.
3. Existing clients cannot interact with new-versioned at all. Some of the changes proposed for MXID anonymisation[5] might fall into this bracket.

It is clear from the above list that older clients should not automatically be precluded from interacting with new rooms. Even in case 2, we would rather allow users on older clients to use the room with reduced functionality than to lock them out altogether.

[5] MSC1228

It is also worth noting that, whereas a server's interaction with a newly-versioned room can be policed when it joins the room, this does not work for clients, since it is easy for a user to join a new room with one client (which might fully support the new room version), and then switch to another device.

Clients which understand multiple room versions could inspect the m.room.create event to see which version they are dealing with and adjust their behaviour accordingly.

Otherwise, it seems best to suggest a range of measures that we might apply when considering room version upgrades, and leave the decision of which to use until we know exactly what is changing. Such measures might include:

- Allow clients to list the room versions they support in the /sync request. Any rooms which are deemed "incompatible" could be elided from the /sync response.
- Similarly, we could reject attempts to join incompatible rooms.
- ...