1. `$events` — reference the IG operation directly [EDIT]

v0.6 says the operation "retrieves a specific event-number or range" returning "notification-event Bundle(s)". The IG defines it completely: parameters `eventsSinceNumber`, `eventsUntilNumber` (both string, inclusive bounds) and `content` (a hint the server may ignore); the reply is one `history` Bundle whose first entry is the SubscriptionStatus. Our first cut read v0.6's prose as "unspecified" and invented parameter names — the divergence a paraphrase invites.

Recommendation: replace the prose with a normative reference to `OperationDefinition/backport-subscription-events`, and state whether `content` is honoured.

2. `$status` — reply shape does not match the IG [EDIT]

v0.6: "200 OK → SubscriptionStatus Parameters". IG: the operation returns a **`searchset` Bundle** containing one SubscriptionStatus Parameters per subscription queried. v0.6's SubscriptionStatus field table also omits `events-since-subscription-start` ("the total number of actual events generated since the Subscription was created … NOT incremented for handshake and heartbeat"), which is the one field a client needs after a missed heartbeat. Our first implementation followed v0.6's wording and was therefore non-conformant to the IG.

Recommendation: reference `OperationDefinition/backport-subscription-status` and the `backport-subscription-status-r4` profile rather than restating them.

3. Duplicate delivery has no defined response [v0.6]

Neither the IG nor v0.6 states what a receiver answers to a re-delivered notification. v0.6 introduced the idempotency obligation (obligation 6), so it should complete it.

Recommendation: "A receiver SHALL acknowledge a previously processed notification with 200 and SHALL NOT re-process it." A receiver answering 422 "already processed" would loop against a sender that retries on failure.

4. Subscription status after persistent event-delivery failure [EDIT/v0.6]

The handshake table says persistent failure → `error`. The event-notification table has no 5xx row at all (only 200 and 422) and never says what happens to the subscription when event delivery keeps failing. The R4/R4B value set v0.6 inherits already provides the semantics: `error` = "the server has an error executing the notification", `off` = "too many errors have occurred or the subscription has expired".

Recommendation: add the 5xx row to the event table with the same retry rule as the handshake, and state the transitions: transient failure → `error`, retries exhausted → `off`. Also state what a sender does on a 422 to an event notification (no retry; is the subscription `error`? does the event count as sent?).

5. Out-of-band: the receiver cannot honour its SHALL [v0.6]

The IG's server-managed mode needs no extra transaction — the sender creates its own Subscription. What v0.6 adds is a receiver obligation: it *MAY* inspect the payload mode "by fetching the Subscription resource" and *SHALL* refuse the handshake if the mode is outside the agreement. But every v0.6 example carries a **relative** reference (`Subscription/7f3e…`), and neither the notification Bundle nor the IG provides sender identity or a base URL. A receiver facing a handshake for a subscription it never created has nothing to dereference, so the optional inspection is impossible and the mandatory rejection cannot be performed. Our POC only passes this permutation because it emits absolute references.

Recommendation — either of:

- mandate an **absolute URL** in `SubscriptionStatus.subscription` for all notifications; or
- define **sender identification** at the transport level — the mTLS client certificate or the OAuth access token's organisation identifier (URA), as introspected by the receiver's PEP — and specify that the receiver resolves the sender's FHIR base via GF Adressing (mCSD) to dereference the relative reference.

6. Handshake example contradicts the status table [EDIT]

The SubscriptionStatus table says `status` is "`active` or `off`"; the handshake example carries `requested`. Verified against the IG: the `status` parameter is bound (required) to the R4B subscription-status value set — `requested`, `active`, `error`, `off` — and a handshake is sent while the subscription is `requested`, so the example is right and the table is wrong.

Recommendation: list the full value set in the table.

7. Heartbeat: shape not named, failure mode undefined [EDIT/v0.6]

Shape [EDIT]: heartbeats are "optional per Subscription, agreed at creation", but v0.6's Subscription element table has no heartbeat element. The IG defines it: `backport-heartbeat-period` on `Subscription.channel`, `unsignedInt` seconds, "if a value is present, a keepalive notification is transmitted at regular intervals matching the value; when absent, no heartbeat notification occurs". Name it.

Failure mode [v0.6]: the IG only says a server "may send" heartbeats and is silent on a server that cannot honour a requested period. "Agreed at creation" is therefore v0.6's own construct and v0.6 must define what happens when the agreement is not met. In our POC the server strips the extension; the client is not told to re-read the created resource, keeps its timer, and issues a `$status` query per missed window — indefinitely, harmless only because this client tolerates it.

Recommendation: mirror the IG's own rule for payload content ("servers SHALL reject the subscription request if a client asks for a content level the server does not intend to support"): a server that does not support the requested heartbeat period SHALL reject the in-band creation with 422 rather than silently strip it. For out-of-band, the period the sender put in the resource is authoritative.

8. Filter criteria: syntax is fine, the NL value convention is not [v0.6]

Verified against the IG: `[resourceType].[filterParameter]=[value]` is a valid form, so the example `Task.owner=Organization/receiver-org` is syntactically conformant (`owner` is a Task search parameter). What is undefined is the **value**: in the Dutch context the partner is identified by URA, not by a reference the sender can resolve. Our POC uses the token form `Task.owner=http://fhir.nl/fhir/NamingSystem/ura|00000020`.

Recommendation: since this filter is the only per-partner scoping mechanism on a broad topic, v0.6 (or the GF IG) should fix the NL value form to the URA token.

9. Endpoint discovery needs one new payload type, not two [GF]

Resolving where to send notifications is GF Adressing's job, but no `Endpoint.payloadType` code exists for a rest-hook notification receiver. Our POC initially introduced two codes. On reflection only **one** is needed: the receiver's notification endpoint is a distinct surface and needs its own code (there is precedent for `Twiin-TA-notification` in earlier Nuts material). The sender-side operations — `POST /Subscription`, `$status`, `$events` — are ordinary interactions on the sender's **existing FHIR endpoint**, which the IG says should advertise subscription support via `CapabilityStatement.instantiates` (R4B) or the `capabilitystatement-subscriptiontopic-canonical` extension (R4).

Recommendation: standardise one payload-type code for the notification receiver endpoint in GF Adressing, and require senders to advertise supported topics in their CapabilityStatement per the IG. Note the interlock with point 5: the reverse lookup (notification → sender) is what a relative subscription reference requires and what mCSD alone does not provide.