FHIR Versioned APIs -- 2018-05-HL7-WGM

As FHIR is adopted by more health care organizations, it becomes increasingly important for servers and clients to be able to communicate using more than one version of the FHIR standard. This also enables asynchronous upgrades of servers because it is not necessary for all clients of a server to simultaneously update to use a newer version.

Scenario 1: Simple read of resources from multiple servers with different versions

Some services will need to read from multiple FHIR servers that are maintained by external organizations. It is unlikely that all servers will upgrade their FHIR versions at the same time. Server operators, in an effort to not break integrations with partners, will eventually become stuck on an old version, and clients will be reluctant to upgrade because the server operator has not.
If clients have the ability to detect and properly interface with FHIR APIs using more than one version, then server operators can confidently upgrade their systems to newer versions without fear of breaking their clients.

Workflow:

1. Construct a GET request to '/Resource/:id' and set header 'Accept: application/json+fhir;fhirVersion=<preferred-version>;application/json+fhir;fhirVersion=<older-version>;application/json'
2. Server interprets the accept header as a prioritized list of requested formats/versions and picks the first one it supports, generates the response and adds a 'Content-Type: application/json+fhir;fhirVersion=<chosen-version>'
3. The client can extract the FHIR version number from the content-type in the response headers and use that to process the response. If the server does not support multi-versioning, the response will not contain version information. The client can the determine the version from the capability statement of the server.

Servers SHOULD be able to return any known resources in any requested version for which that resource exists. It is possible for the server to extend its understanding of the Accept header to return the first format and version for which the resource exists. Resource IDs MAY NOT vary across FHIR versions.

For example, a client may request a Patient in v3.0, or v1.0 and the server may respond with a 3.0 Patient in some cases and 1.0 in others if they are unable to convert to 3.0 for some reason.

Scenario 2: Search for resources

Clients will frequently need to be able to query a server for information about resources.

Workflow:

1. Client requests the /metadata for the server using versioned Accept headers and uses this statement to determine how to properly construct a search query. In this case the mime-type for the content-type would be `application/x-www-form-urlencoded;fhirVersion=x.x`
2. Using information from the /metadata, the client makes a search request for the required Resource. In this case the Accept header should only include a single version that corresponds to the version used to construct the query. The content-type header should also be set to the same versioned mime type. Do not structure the query such that it may be making the request in one version and asking for a response in a different one.
3. The server receives the request and responds to it. Using the search semantics appropriate to the requested version of FHIR. The server sets the content-type header to the requested version and will return a Bundle in the requested version. All resources returned by the query MUST be the same FHIR version.

Scenario 3: Create/Update a Resource

Another common use case is to create or update a resource. This workflow is similar to the search case in that the client needs to understand the capabilities of the server and select a single FHIR version for communication.

1. Client selects a mutually supported FHIR version by inspecting the target server's /metadata.
2. A POST / PUT request is issued with the Accept and Content-Type headers set to use the versioned mime type appropriate to the selected version.
3. The client executes the request
4. The server executes the request according to the semantics appropriate to the selected version of FHIR.
5. The server responds as usual but sets the content-type of the response to the appropriate versioned mime type.

It is possible for a resource to be created in a particular version and not become immediately available in other supported FHIR versions. For example, if the server stores records as version specific files then a new resource would not be available until the converted file is created.

Scenario 4: Delete a resource

Workflow:

1. Client issues a DELETE request for a resource with versioned mime type accept header (which may include multiple versioned mime types). The Resource type MUST be referenceable in the requested FHIR versions.
2. Server performs the DELETE request according to the semantics of the version of FHIR selected.
3. Server responds appropriately and sets the content-type with negotiated versioned mime-type if there is a response body.
4. Conditional deletes behave like a search

Note: Deletions are a semantic assertion that a resource should be removed. When a record is removed, it should be removed from all versions it was previously available in, i.e, the server should not delete a 1.0 Patient without also deleting the 3.0 version of the same Patient. Likewise, undeletion of a deleted record should make it available in all appropriate supported FHIR versions.

Scenario 5: Operations

Workflow:

1. Client determines a mutually agreeable FHIR version between the client and the server.
2. Client issues a GET/POST request with an Accept & Content-Type header containing the versioned mime type.
3. Server processes the request according to the semantics of the chosen FHIR version and responds in the agreed FHIR version format.

Additional Comments

- The fhirVersion parameter should be set to the semantic version of the FHIR version desired. Patch level versions are not significant for these interactions so a major.minor version is preferred. Providing the patch version MUST be ignored by the server (for this purpose 3.0.1 == 3.0).
- It may be useful for a server to report all possible format/version combinations in its CapabilityStatement.format as a mechanism for clients to discover which versions are supported.
- Clients and Servers should add the fhirVersion parameter to all mime types used except when explicitly being used to refer to the unversioned mime type.
- Servers MUST return a different CapabilityStatement / ConformanceStatement resource depending on the requested version. Servers MAY vary their capabilities by version. For example, a server may offer full support for 3.0 and read-only access for 1.0.

- Servers that support multiple FHIR versions MUST define a default FHIR version to use in the event that the client does not request a specific version. The default version does not need to be the most recent FHIR version. Some operators may choose to default to an older version of FHIR to support older clients and allow version-aware clients to use newer versions.
- a light-weight request for getting the supported version of the server might be GET /metadata?_elements=fhirVersion
- A version-aware server MUST return a 406 Bad Request (and OperationOutcome if appropriate) if none of the requested FHIR versions are understood and the client has not included a generic, versionless fallback format (i.e, application/json or application/xml), which should be interpreted as the default FHIR version.
- This approach is backwardly compatible with existing servers. Servers that do not understand the fhirVersion parameter on the mime type MUST simply return the requested format in whatever version they support.
- The FHIR spec allows the format of a request to be overridden by the _format query parameter. It may be useful for this parameter to also accept the fhirVersion parameter to allow specifying the expected FHIR version of the resource. However, most FHIR server implementations in production use currently fail to handle _format with the additional parameter. Servers either ignore the _format parameter completely or they raise exceptions in response.