HL7 Process for Publishing a FHIR IG

- 1 Publishing a FHIR IG
- 2 Pre-conditions
- 3 Record Keeping
- 4 Checks
- 5 Local Build
- 6 Publishing the IG
- 7 Finishing the Publication Process

Publishing a FHIR IG

This process is followed by anyone when publishing a FHIR IG to http://hl7.org/fhir/

Pre-conditions

To follow this process, you must have:

- permission from the FHIR Product Director to publish FHIR IGs (including write access to the [FHIR IG Publication Record])
- permission from FMG+TSC to publish the particular FHIR IGs
- an FTP account with access to the source for http://hl7.org/fhir
- permission to push changes to https://github.com/FHIR/ig-registry
- A local clone of the FHIR IG registry github repository https://github.com/FHIR/ig-registry to {registry}
- A local clone of the FHIR History Template github repository https://github.com/HL7/fhir-ig-history-template to {history}

You must also know how to run the IGPublisher locally

Record Keeping

All publication runs must be recorded in the [FHIR IG Publication Record]

When you being processing the publication request, create a new row in the sheet, and add the following columns:

- Operator : Your name
- IG Name : The formal name for the ballot or publication request
- Publication Request: a link to the publication request that initiated the work (a link to confluence)
- IG approval: a link to the IG approval by FMG (a link to confluence - if there isn't one, consult the FHIR product director and put "(n/a)"
- PPS Link: a link to the original project approval (either confluence or project manager - if there isn't one, consult the TSC co-chair)
- TSC Approval: a link to the TSC approval (if this is a formal publication, otherwise blank)
- Primary Editor : the name of the editor who will handle any QA issues
- IG Source : The github repository URL where the source is located - Check that the repository URL is HL7 (or an other wise approved Organization - consult CTO for approved list)
- Branch: the branch that is being published (defaults to master)
- new IG : "yes" if this is the first time that any version is being published for this IG (whatever version - draft or ballot)
- Milestone?: "yes" if this a formal publication release (usually approved by TSC)

Checks

The following checks must be run. All these checks confirm that the various HL7 paper work which lives all over the place is all lined up, and that the IG correctly identifies itself as approved. Note that these records are all over the place, and it's really hard to work to get everything aligned, so expect to find problems.

0. Preparation

- Find the IG (based on it's github URL org and name) in https://fhir.github.io/auto-ig-builder/builds.html, and check that the IG was built in the last 24 hours. If not, rebuild it. Wait for it to rebuild (refresh the page, or watch https://chat.fhir.org/#narrow/stream/179297-committers.2Fnotification).
  Now find the QA Page.

1. Realm

- IG Approved realm: enter the realm that FMG approved when the IG was approved
- Project Realm: enter the realm that was approved for the project under which this IG is approved. If the project realm does not match the IG realm, consult the FHIR Product Director or the TSC co-chair
- Requested Realm - enter the realm that was specified in the publication request
- IG Actual realm - enter the realm stated for "Realm Rules" in the QA page. Make sure it is lowercase. If it's not, or either the stated or the actual realm doesn't match the approved realm, consult the FHIR Product Director
2. Code
- IG Approved Code: add the code as identified in the IG Approval page (note that the name of the github repository does not need to be the same)
- IG Actual Code: add the code as stated in the QA page ("Publication Code"). If there is in error shown, or if the codes does not match what was approved, consult the FHIR product director (important: this code is case sensitive - make sure you check the case)

3. Version
- Requested Version: enter the version requested in the publication request (should be in Requested name for published standard). If this is not specified in the publication request, then talk to the nominated editor.
- Stated Milestone: if this is a milestone, the stated version (e.g. "STU 1") - you can get this from the version check in the QA file. If it is not populated, or doesn't match the publication request, consult the editor and/or TSC co-chair
- Actual Version: copy this from the version specified in the "Version Check" row in the QA page. If there are any errors shown, or it doesn't match the requested version, consult the FHIG Product Director
- Check the stated version against the sequence and the stated STU intent.
  - The major version should match the STU intent (per the NIB) and this should also match the sequence too.
  - If the publication is a milestone, there should be no label e.g. 2.0.0 for STU 2 in the sequence STU 2.
  - If the publication is not a milestone, there should be a ballot label e.g. 2.0.0-ballot. For the first ballot, the version label will simply be ‘-ballot’. Subsequent ballots of the same document will be -ballot2 -ballot3 etc.
  - FMG may authorise additional publications such as -draft and -preview. FMG approves the labels explicitly in this case
  - If clarification is needed, either the FHIR Product director or the FMG co-chairs can advise.

4. Publishing Integrity
- Fill out the canonical from the Publication Code row in the QA page, and check that it's the same as http://hl7.org/fhir/{realm}/{code} - consult the FHIR Product Directory if it's not
- Fill out the package id from the Publication Code row in the QA page, and check that the package id the same as hl7.fhir.{realm}.{code} - consult the FHIR Product Directory if it's not. Add the version (#version)
- Fill out the subdir column using the value from the version check row. If the value is ??, consult the FHIR product director
  - If it does not conform to the pattern YYYYMMM (with the date correct) or STU(X), check with the FHIR product director
- Validate the location of the URLs listed in the package-list.json e.g. ci-build URL

5. Content quality
- Check the QA page Quality Checks section. Specific things to review:
  - Publisher Version is identified, and not reported as out of date
  - Publication Code: no errors reported
  - Realm Check: FMG / US Steering committee checks this - can be ignored
  - Version Check: check no errors reported
  - Dependency checks: no errors / issues identified
  - HL7 Publication rules: no errors/issues identified
  - HTA Analysis: this can be ignored (for now)
  - Previous version comparison: can be ignored
  - Summary: copy to "QA Summary column"
    - If the error count > 0, refer to the FMG co-chairs or the FHIR Product director, unless you are sure that the list is unchanged from FMG approval
- Template checks
  - Check template currency (at end of Dependency Checks). The stated version for fhir.base.template must be 'current' or more recent than '0.2.1'
  - If there is no stated template, or it is older than 0.2.1, consult the FHIR product director
  - Record the fhir.base.template version in the spreadsheet
- IG Review
  - Open index.html, and check that the header shows the name of the IG and the ballot status correctly

OK, if you get to here, we are good to publish.

Local Build

Now that the main checks have been done, you need to do a local build. To do that, you need a copy of the IG Publisher - see IG Publisher Documentation for assistance

- Download the latest igpublisher from https://github.com/HL7/fhir-ig-publisher/releases
- Upgrade sushi:
  ```npm install -g fsh-sushi```
- Clone the github repository to your local drive = {clone}
  - you should call it [packageId]#(version)
- Run the IG publisher, clearing the terminology cache (run the publisher.jar -ig {clone} -resetTx -publish {url})
  - where {url} is the url that the guide will be published at (which is the path entered into the spreadsheet above)
- Check that the local {clone}/output/qa.html file matches the CI build qa.html
  - Troubleshooting on https://chat.fhir.org/#narrow/stream/179165-committers
  - Once it matches, record the summary in the local qa column
• Zip up the folder (clone) as it is to a file named {packageId}#{version}.zip and ftp to ftp://hl7.org/fhir/ig-build-zips

You are now good to publish the IG

Publishing the IG

Note: This describes doing this as part of a batch, since this is a common case, but you can do the entire sequence for a single IG:

Do once for the batch:

• create a local directory = {root}
• use FTP to copy the current contents of ftp://hl7.org/fhir to {root}
  • 1 million+ files. Keep this maintained in advance, or give 48 hours + to build it
  • Check with other publishers before making the copy (publishers = Grahame & Lynn)
• if you keep this maintained, use FTP to sync your copy each time the contents of http://hl7.org/fhir/package-feed.xml changes (subscribe with a feed reader), or as discussed between publishers
• check that {registry} and {history} are up to date (git PULL)

when all these are synchronised, do once for each IG:

• If this is the first time the IG has been published, set up the destination subfolder
  • create the directory for into the root destination subfolder using the ig code in {root}/us or ../uv
  • copy into the root destination subfolder the package-list.json file from the clone and edit it to remove the ballot version from the list, leaving only the ci-build in the list
  • check the title, introduction, and category to ensure they have been specified correctly
    • the title should match the approved title for ballot or the approved name in the IG Proposal
    • the introduction should be copied in from the short description in the IG Proposal
    • the category should be filled in, and generally matches one of those already in use at http://fhir.org/guides/registry/
• run publisher.jar -go-publish -source {clone} -destination {root} -registry {registry}/fhir-ig-list.json -history {history} milestone
  • only add the -milestone part if you are publishing a milestone (from the checks above)
  • If this process fails, you need to resync {root} so that it has the same contents as before you ran it, and then consult the FHIR Product Director, and then repeat
  • Warning: this resyncing can take a long time. The process will tell you when it starts making changes to {root} so you know if you have to. Keep a spare local copy if resync will take too long
• mark “yes” in the Published column

Finishing the Publication Process

• if you are doing a milestone release: use FTP to delete the contents of ftp://hl7.org/fhir/{realm}/{code}; yes, delete all the files
• use FTP to copy the entire contents of {root}/({realm})/{code} to ftp://hl7.org/fhir/{realm}/{code} - no need to upload files that have not changed
• mark “yes” in the FTPed column
  • if this is a milestone publication, let Lynn know that it’s time to mark applied tasks as published, and considered for future use tasks as open (see Administering Specification Feedback Projects#Publishing issues)
  • if this is a milestone publication (not for ballots) make an announcement on the https://chat.fhir.org/#narrow/stream/179240-Announcements/topic/FHIR.20Publication.20Announcement topic in Zulip using the following template:
    • new Publication: [publication name] of the [IG Name] implementation guide: {canonical}/{subdir}
    e.g. New Publication: STU Update 3.2 of the QI-Core implementation guide: http://hl7.org/fhir/us/qicore/STU32/

Final steps. Note that if you are publishing more than one IG, do once for the batch:

• FTP the files {root}/package-feed.xml to ftp://hl7.org/fhir/package-feed.xml and {root}/publication-feed.xml to ftp://hl7.org/fhir/publication-feed.xml
• check the changes to {registry}/fhir-ig-list.json (use git diff) and fill out any missing information (look for ?? in the changes) - may need to talk to FHIR product director to decide what to put
• commit and push the changes to the FHIR IG registry to github to a new branch, let Grahame know, & mark “yes” in the Register IG column