CityGML 3.0

# Using Git and GitHub

Participants must

1. Obtain a GitHub user name and password by registering at github.com
2. Clone the master branch of the repositories (just the one repo is shown in the diagram but the process is the same for both). This is most easily done on the web via the GitHub “fork” button. Having this clone on GitHub serves three purposes:
   1. The participant has full access rights and can make changes with no interaction with or effect on anyone else
   2. The participant’s changes are accessible to anyone who knows that their cloned repo exists (e.g. via email or pull requests)
   3. The participant can share their changes with others by sending an invitation to sync with the clone – this is most easily done with a GitHub pull request
3. Clone the participant’s clone to one or more local machines to which the participant has physical access. Having this clone serves two purposes
   1. The participant can add, modify, and delete files and directories – there is no direct mechanism to do this with the clone hosted on GitHub.
   2. The participant can experiment with code or create documents without disturbing the posted (GitHub clone) until ready to share by committing the local changes and pushing the local repo back to the participant’s clone on GitHub
4. When the participant wishes to submit changes to the original master on GitHub, he/she can send a “pull request” to one of the maintainers
   1. The maintainer should pull the files to their local copy, review them, and – if OK (for example, targets the correct work package) – add and commit them to their local repo and then push them to the GitHub original repo.
   2. The maintainer should inform the participant of the action.

**Normal participant workflow:**

1. Participants work locally, add and commit changes locally on their own schedule.
2. When they want to make their changes visible, they push to their GitHub clone
3. When they want their changes incorporated into the GitHub master they inform a maintainer via a pull request or other means
4. The maintainer reviews and publishes the changes.

**Publication from the private to the public GitHub repo on approved SWG motion:**

1. Participant makes motion to SWG
2. SWG accepts motion to publish
3. Maintainer copies file from private local working copy to public local working copy as SWG motion directs
4. Maintainer adds, commits, and pushes public repo to GitHub origin

**Un-publication from the public GitHub repo on mistaken publication or SWG motion:**

1. Maintainer remove the file public working copy as SWG motion directs
2. Maintainer adds, commits, and pushes public repo to GitHub origin

Of course, a maintainer can add, remove, or change files directly on their local repo. That’s what we have done so far. It’s also possible for a participant to avoid using Git by sending changes by email to a maintainer with a request describing what they want done.