**Background**

Whenever users work with the Cadastral Editor, a description of each edit is appended to the database. These edits are not instantly visible to other users of the system. Instead, they remain visible only to the user who made the edits. The intention is that the user would subsequently publish their work to the wider user community. An initial (flawed) implementation of this idea was previously coded (the **File – Publish** menuitem).

This document outlines a more powerful approach that is based on the idea of editing branches. It takes on board the ideas from a technical paper that was produced by Smallworld in the early 1990s (see <http://cfis.savagexi.com/pages/technical_paper_4>).

**What is a Branch?**

A branch has the same meaning as a branch in a revision control system such as Subversion. However, whereas Subversion branches contain a collection of software code, Backsight branches contain a sequence of edits (where each edit is described by an xml document).

The sequence of edits in a branch is strictly append-only. If you need to revise a previously defined edit (e.g. you discover that an observation was incorrectly entered), you can do so using the Cadastral Editor – however, this revision will be represented by an additional edit that is appended to the sequence.[[1]](#footnote-1)

Branches can be arranged in a hierarchy, making it possible to specify an organized data entry regime. For example, you might define branches for workgroups, with child branches for each individual operator. Alternatively, you might want to create a branch for each survey plan that comes into the system.

Backsight tries to be agnostic in terms of branching – it tries to avoid imposition of a branch hierarchy, because that might interfere with the operational goals of people trying to administer the system. The only rule that Backsight does enforce is to ensure that each map layer must correspond to a top-level branch. Users can only define sub-branches that are children of these top-level nodes.

**Editing Sessions**

Whenever you start the Cadastral Editor, the software needs to know two things:

1. The branch where the data should be appended.
2. The spatial zone that identifies where the data exists.

As previously noted, all branches are ultimately related to a map layer. The branch as a whole represents the entire coverage of the layer. The zone is a name for a specific region within the layer, (which makes it possible to deal with large databases in a scalable fashion). In normal practice, both these items will be picked up from the CEDX file that is used to launch the Editor (in a situation where the user does not have a CEDX file, they will be prompted).

When a session is initiated, a row will be appended to a database table called Sessions. Each row in the Sessions table contains a permanent record of the branch and zone. Every session is also referred to a previous session known as the *branch tail*.

**Posting and Receiving**

**Map Layers, Themes, and Branches**

Backsight makes it possible to relate map layers in a hierarchy called a *theme*. For example, the *Property* theme (defined as part of the sample Manitoba environment) consists of three layers called *Survey*, *Ownership*, and *Assessment*.

Data on the Survey layer is regarded as base material for the Ownership layer. Most of the spatial data is exactly the same, except for a few small differences that reflect a different perspective on the data. For example, an organization dealing with ownership issues might combine adjacent survey lots if they are owned by the same person. An organization dealing with tax assessment might also have a slightly different view that is based on ownership.

Given that a theme represents a hierarchy of map layers, it is tempting to think of these layers as a hierarchy of branches (Survey 🡪 Ownership 🡪 Assessment).

**Database Structure**

1. I believe that the git version control system (see <http://git-scm.com>) also takes an append-only approach. [↑](#footnote-ref-1)