**Create New Project**

To create a new project, you must specify three things:

1. The name for the project
2. The map layer the project is associated with
3. The default coordinate system

Given this information, the following steps are taken:

* Create a file called {project-name}.txt in %Backsight%\private\index
* Generate a GUID and write it to the project file
* Create a folder for holding data called %Backsight%\private\{ project -guid}
* Write a file called 00000000.txt to the data folder. This holds the *NewProject* event data (consisting of the project name, the layer ID, the default coordinate system, the project GUID, the name of the login user, and the name of the local computer).

When you create a new project, you also begin a new editing session. This is considered to be a separate “editing event” that will be serialized to a file called 00000001.txt. This contains the time when the editing session started, the name of the login user, and the name of the local computer.

As you continue with the editing session, a further data file will be created for each edit. The number assigned to each successive data file may increase by more than 1. The number indicates the total number of internal IDs that have been utilized by the project. These IDs refer to a variety of the objects created during editing work. Every event gets a distinct ID, and every spatial feature gets a distinct ID as well. For example, an edit that creates two features would utilize 3 IDs – one for the overall edit, and two for the new features. In situations where an edit needs to refer to a previously created item, that reference will be persisted using the item ID.

When you exit from the Cadastral Editor application, an *EndSession* event will usually be recorded. It merely holds the time when the session was completed, and acts as the closing bracket on the initial *BeginSession* event. This makes it possible to subsequently confirm that the session completed normally.

An *EndSession* event will not be recorded if no edits have been performed (i.e. the preceding event is the *BeginSession* event). In that case, the *BeginSession* data file will just be removed.

**Re-open a Private Project**

To re-open a project, you select a file from the %Backsight%\private\index folder. The software opens this file to read back the GUID that was assigned to the project. This leads to the data files in the %Backsight%\private\{project-guid} folder.

Once all the data files have been deserialized into memory, the geometry of spatial features will be calculated. A *BeginSession* event object will also be created in memory at this time. However, it will not be written to disk until you perform your first edit.

Suppose for example that you access a project data folder that holds 5 files:

00000000.txt – CreateProject event data

00000001.txt – BeginSession event data

00000008.txt – First edit (used IDs 2 to 8 inclusive)

000003A5.txt – Second edit (used IDs 9 to 3A5 inclusive)

000003A6.txt – EndSession event data

If you do not perform any edits, the content of the data folder will remain unchanged (i.e. nothing changes if all you do is look). However, if you perform a single edit, then exit, you will create 3 additional files:

000003A7.txt – BeginSession event data

000003B0.txt – Edit (used IDs 3A8 to 3B0 inclusive)

000003B1.txt – EndSession event data

**Make a Project Public**

New projects are considered to be private (only users on the local machine are expected to work with them). You do not need to register the project in any database or repository. This helps to cover a use-case in which Backsight is used only for single-user, occasional work.

To make a project available to anyone else (or to easily access it from another place), you must "publish" the project to the cloud (currently Google Docs is used for this purpose - referred to below as the "server"). When you publish a project for the first time, you will be asked to supply Google account login details. Given this information, the following happens:

* The private data files will be combined into a single file that is created in a new folder called %Backsight%\public\{project-guid}. The combined file in the public folder takes the same name as the last data file in the private data folder.
* A folder called {project-guid} will be created on the server, and the combined data file will be copied there.
* The project file in %Backsight%\private\index will be moved to %Backsight%\public\index
* The project file will be copied to an index folder on the server.
* All files in %Backsight%\private\{project-guid} will be deleted

Having completed all the above, a new empty file will be added to the private data folder. It takes a file name called {combined-data-file-name + 1}.txt (i.e. if 00001234.txt has just been added to the public data folder, an empty file called 00001235.txt will be added to the corresponding private data folder).

Publishing a project makes it equally accessible to others. To start working with the project, they need to subscribe to it.

**Subscribe to a Public Project**

To subscribe to a public project, you use the Cadastral Editor to open a project on the server. When you do that, the following happens:

* The project file will be copied to your local %Backsight%\public\index folder
* A folder called %Backsight%\public\{project-guid} will be created on your machine, then all data files for that project will be copied in from the server.
* An empty file will be created in %Backsight%\private\{project-guid}. The file takes a name that is 1 greater than the last public data file. For example, if the last data file in the public folder is 00001234.txt, an empty file called 00001235.txt will be created in the corresponding private folder.

The local file structure is now identical to the structure on the machine of the original publisher. Either copy of the project can be edited. Any edits will generate new data files that appear in the private data folder. To make these changes available to others, you need to publish them.

**Publish Changes to a Public Project**

When you publish a new project, or subscribe to a project, your private data folder holds a single empty file. The name of this file is 1 greater than the last public data file. This file is a placeholder for the next publication.

When you perform edits, the data files are get names that are numbered after this placeholder. So suppose your placeholder is 00001235.txt, and you have performed a series of edits that generated data files up to 00002340.txt. When you say you want to publish this data, the software will contact the server to download any data files that have been placed there since your last subscription to the project. Two cases:

1. No changes have been received on the server
2. Someone else has published changes to the server

To cover the second case, the extra edits must first be added to your map, as described in the next section. Having done that, the logic runs as follows:

* Record the number of the last public data file in your placeholder file. The number of changes that have taken place on the server (if any) since your last subscription defines an offset ∆ to apply to IDs in the data that you are publishing.
* Combine the files in your private data folder into a single file, placing the result in the public data folder. The combined file name is defined by the name of the last private data file + ∆
* The combined data file is copied to the server
* Delete all files in your private data folder

For example, consider the simple case before and after:

|  |  |
| --- | --- |
| **Before** | **After** |
| public\{project-guid}  00000000.txt  00000100.txt  00000200.txt | public\{project-guid}  00000000.txt  00000100.txt  00000200.txt  00000300.txt |
| private\{project-guid}  00000201.txt  00000205.txt  00000300.txt | private\{project-guid}  00000301.txt (empty) |

In this case, the beginning of 300.txt records the fact that no offset is involved. Now consider a case where someone else has published 250.txt:

|  |  |
| --- | --- |
| **Before** | **After** |
| public\{project-guid}  00000000.txt  00000100.txt  00000200.txt  00000250.txt | public\{project-guid}  00000000.txt  00000100.txt  00000200.txt  00000250.txt  00000350.txt |
| private\{project-guid}  00000201.txt  00000205.txt  00000300.txt | private\{project-guid}  00000351.txt (empty) |

In this case, the beginning of 350.txt records the fact that all internal IDs between 201 and 300 should be offset of 50. This will be done whenever 350.txt is deserialized.

**Receive Changes**

Open Published Job

Publish Revised Job