# Guidelines for Driver and Simulator Developers

The ASCOM development servers are available for any developer who wishes to contribute code related to The ASCOM Initiative. The Subversion source code control system makes it easy to collaborate with a team of developers, carry out peer reviews and give other ASCOM developers visibility of your code. It’ll also make sure you code is backed up, both on other developer’s working copies and in the daily tape backup that we run.

# Where to Put Your Code

As a driver/simulator developer, you’ll have read-only access to the entire ASCOM source tree.

You’ll have read/write access to any projects that you’re actively participating in. This way, you have plenty of reference code to look at and you can freely commit changes to your own part of the tree without fear of affecting others.

When you create a new project, we recommend the following naming convention:

* /*Vendor*/*DeviceName*

Your solution file goes in the /DeviceName folder, with sub-folders under that for each project. If you are working on a compound driver based on the LocalServer (singleton) template, then one of your projects will be named *DeviceName*Server, and you’ll have a series of projects named *DeviceType*Driver for each of the devices supported. Example: say you’re working on a Meade LX-200 Classic driver, with support for both the telescope drive and the built-in focuser. You need to implement both the Telescope interface and the Focuser interface and you’ll expose those interfaces by wrapping them in a LocalServer. This will be your project structure:

/Meade/LX200/LX200.sln  
 /LX200 Server/  
 /Telescope Driver/   
 /Focuser Driver/

# Assembly Naming and Versioning

We recommend naming your projects (and the assemblies they produce) along the following lines:

* **ASCOM**.*DeviceName*.*DeviceClass* (example: **ASCOM.LX200.Telescope**)

We recommend that you give your assemblies the same major and minor version as the ASCOM platform that they were compiled against. So if you’re using ASCOM Platform 5.1, then your assembly should have Major Version = 5 and Minor Version = 1. We suggest letting Visual Studio manage the Revision and Build numbers automatically, by placing an asterisk (\*) in the Build number and leaving the Revision number blank.

# Referencing Platform Components (Platform Dependencies)

When developing drivers, we recommend that you do not rely on any assemblies being installed in the Global Assembly Cache. Your driver should be able to build and run without any version of the ASCOM Platform installed. This will enable our build server to build your solution and help keep the codebase consistent and in good working order.

To facilitate this, you’ll find a folder called Platform Dependencies in your working copy, which contains the latest build of all of the core platform components. Your drivers should reference these files directly, using relative paths. When adding a reference to your project, do not use the .NET or COM tabs. Use the Browse tab and navigate directly to the Platform Dependencies folder.

# Development Servers

There are several servers of interest to ASCOM developers. In the table below, the first two in the list will be essential for you to become familiar with. The others are useful, but you can get by just fine without ever using them.

|  |  |  |
| --- | --- | --- |
| Product | Purpose | URL |
| VisualSVN Server | Subversion source code control | <http://svn.tigranetworks.co.uk/svn/ASCOM> |
| Atlassian Jira | Issue tracking and project management | <http://teamserver.tigranetworks.co.uk:8010> |
| JetBrains TeamCity | Build & Continuous Integration Server | <http://teamserver.tigranetworks.co.uk> |
| Atlassian Fisheye | Source code metrics | <http://teamserver.tigranetworks.co.uk:8060> |
| Atlassian Crucible | Code Reviews | <http://teamserver.tigranetworks.co.uk:8010> |
|  |  |  |

* **VisualSVN**: You’ll use the VisualSVN server on a daily basis as you develop and commit your code.
* **Jira**: This is where you (and your customers/users) create and monitor bugs and improvement requests, project milestones and releases.
* **TeamCity**: The TeamCity build server builds the entire platform each time any new code is committed and checks that everything builds OK. The build happens in an isolated environment. It can also perform unit tests. The standard practice is that you should only commit code that builds with no errors. If your commit breaks the build, then the build server will designate you as the ‘responsible person’ and you’ll be expected to fix things as soon as possible.
* **Fisheye**: Generates metrics, statistics and charts based on the code committed to the source code control server and events that happen in the other servers. Some of this information feeds back into the other servers (e.g. Jira) and helps to provide a comprehensive view of all developer activities. One of the most useful outputs from Fisheye is the ‘activity stream’ which lists everything happening to the code base. You can subscribe to the RSS feed for this activity stream and keep up to date with what’s happening.
* **Crucible**: Supports code reviews. When new code is committed, a formal review can be created. Developers can examine the code and comment on it, perhaps even raising defect reports. In this way, the ‘many eyes’ approach, code quality should be improved and defects detected early while they are still easy to fix. Use of Crucible is optional but highly recommended.