



# ATLAS Products talking to each other

## SUMMARY

ATLAS products store and retrieve site-settings through a shared file called "atlas.ini". A site is defined as a computer network, or a single un-networked computer or laptop. This file is a standard windows ini file that contains information such as the location of the production database and the installation location for each ATLAS product that has been installed on the site.

The atlas.ini file is found by ATLAS products by using the environment variable ATLASINI, which specifies the folder containing the atlas.ini file. This variable may be set through a network login script, or will be prompted for when the first ATLAS product is installed.

ATLAS products find each other via the atlas.ini file, and may communicate with each other using a formalised interface stored in a separate shared library. This library may also be used by third party developers to communicate with installed ATLAS products. The communication allows one product to trigger business processes within another product, and to even show screens from other products in order to facilitate such things as the selection of objects that are controlled by the other product (e.g. selecting a Cruiser assessment to provide summary information for a GeoMaster event).

## ATLAS.INI – SETTINGS FOR A SITE

ATLAS products save their site-wide settings in a file called the "atlas.ini" file. Generally, there is one atlas.ini file in an accessible location on a company's network, or in the case of a stand-alone computer, in the "All Users" section of the "Document & Settings" folder. This file standardises the way on which ATLAS products store and retrieve site specific settings.

### Atlas.ini Structure

The atlas.ini file is a standard windows ini file - a text file with a number of sections each with a header line consisting of a name in square brackets such as "[Database]". Each section contains a number of named properties and their values in the form "name=value", one per line.

Each product has its section, which typically contains standard properties such as the location of the production database, and the installation location of the product.

The following is an example:

```
[FieldMan]
Exe=FieldMan.exe
Path=H:\ATLAS\FieldMan
ProductionDatabase=FieldManData
ProductionServer=C21569

[Forecaster]
Exe=Forecaster.exe
Path=H:\ATLAS\Forecaster
```



# ATLAS

```
ProductionDatabase=Forecaster
ProductionServer=C21569

[Licensing]
Path=X:\public\ATLAS\Licences

[Yield Table Manager]
Dll=YTManagerComInterface.dll
Exe=YieldTableManager.exe
Path=H:\ATLAS\YieldTableManager\YTM.exe
ProductionDatabase=YieldTableManager
ProductionServer=G_Auck
```

Generally, the atlas.ini file is read-only to ordinary users of the ATLAS products. The site settings are set by an administrator, usually when the products are installed. If the product is installed on a laptop or non-networked PC, the user is generally also the administrator, and is therefore able to both read and write the atlas.ini file.

### Finding the atlas.ini file

A system environment variable called ATLASINI contains the path of the folder that contains the site's atlas.ini file. This is set so that ATLAS products can find the atlas.ini file and read the site settings. This environment variable is best set as through a login script in the case of large networks. However, it is prompted for (and then set) by the ATLAS product installers, if it is not already set. This allows stand-alone installations to succeed without the PC owner needing to know too much about setting an environment variable through the control panel.

ATLAS programs may use the programming library object *AtlasIniFile* to do the work of finding, loading and saving the atlas.ini file using the environment variable.

### Using the Site Settings to Connect to the Database

Each database-driven product will have entries in the atlas.ini file named "ProductionServer" and "ProductionDatabase". When the product starts it uses a programming library object *ConnectionServer* to read these settings from the atlas.ini file, and use them to locate and connect to the product's database.

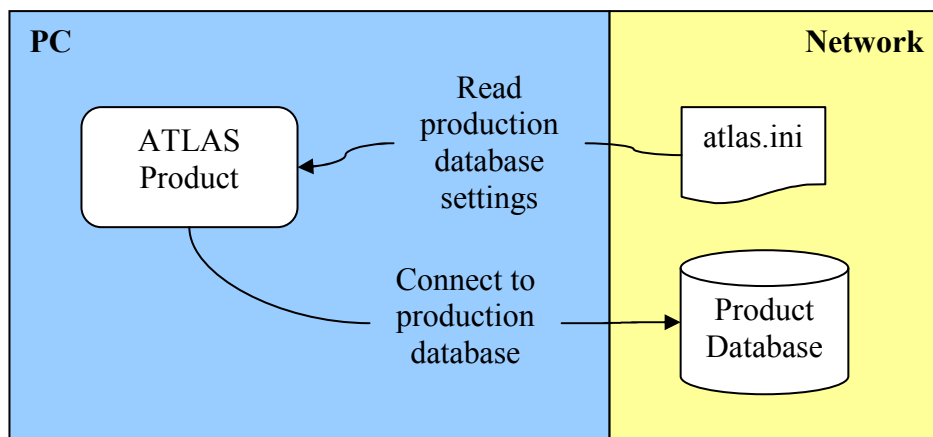


Figure 1: Connecting to the Production Database

This means that every computer on a network will automatically use the same database for the ATLAS product. If the administrator changes these settings all of the computers on the network will connect to the new database automatically.



# ATLAS

## Using an Alternate Database

In some cases it is useful to have a personal (or group) database for experimentation and/or upgrade testing. The *ConnectionServer* object that ATLAS products use to get a database connection has settings for an alternative database that may be configured by each user, and stored locally for just that user.

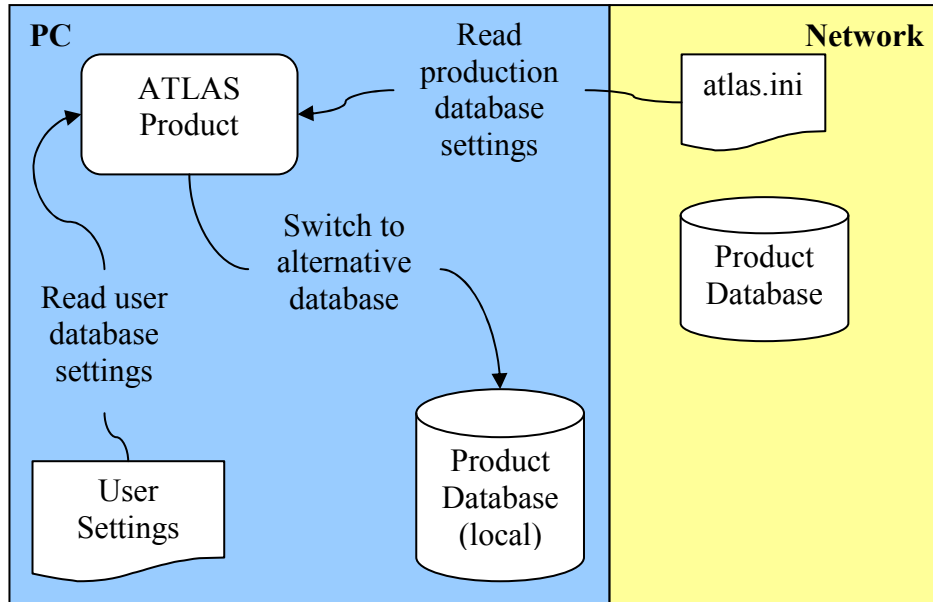


Figure 2: Connecting to an Alternate Database

The user may then switch between the production database and the user’s alternative database using a single button or menu option. Most products will indicate on the product’s main screen if the currently connected database is not the production database.

## ATLAS PRODUCTS TALKING TO EACH OTHER

### Finding who to talk to

The atlas.ini file not only stores site-wide settings for each product’s own use, but is also used to locate other products that are installed on the same site. Note that the other products will need to be installed in a location that the current user has access to, and all of the pre-requisites for running the program will need to be met. For instance, the user may need to be licensed to run the other program, and have permission to access the other product’s database.

Each product section may have the settings “Path” and “Exe”, which specify where the product is installed and what the name of the main program file is, respectively. Other products may use this to locate and start a companion product. A programming library object *AtlasProductFinder* may be used within the program to do the work of looking up a product and starting it.

### Speaking the same language

In order to regulate communications between ATLAS products, a formal *interface* is designed for each product that defines the available methods and properties. This interface defines the language that other programs can use to talk to the product. This interface is shared with other programs by putting it in a library of its own. The



## ATLAS

library can be included in other ATLAS products, or even third party products, to enable communication with the product.

The interface also includes the functionality to find and connect to the product it was built to interface to, through the settings in the atlas.ini file. So, the product that is using the interface only has to create the interface object, and all of its services are then available.

### **Integrating products through communication**

The interface acts as a bridge to actually include the relevant parts of the product into a second product that is using the interface, as it is running. Therefore, it is not necessary to separately start the product that the interface was built for. This bridge allows the product that is using the interface to take advantage of the logic behind the interface, and even show windows through the interface. Each product will still be responsible for maintaining the business logic of various objects (such as a Cruiser assessment) and control how they are stored in its database, but the interface allows other products to trigger those business processes.

This level of integration may start to blur the distinctions between products. For example, GeoMaster may show a screen for selecting a yield table for a harvest area that is actually provided by Yield Table Manager, or Cruiser may push the results of an analysis into the summary of the related GeoMaster assessment event, close the event, and then show the results on a GIS map.