

# C-ISM-A.5.6: Managing iShare Maps 5.6

## *An Astun Technology iShare Course*

<b>Code</b>	C-ISM-A
<b>Title</b>	Managing iShare Maps
<b>Description</b>	Covers all aspects of iShare Maps administration.
<b>Required Software</b>	iShare Maps 5.6.*
<b>Target Audience</b>	iShare Administrators
<b>Pre-requisites</b>	None
<b>Duration</b>	1 day
<b>Version</b>	1.0
<b>Updated</b>	19 Sep 2017
<b>Updated by</b>	<a href="#">Jo Cook</a>
<b>Status</b>	Complete

## Introduction

This course is designed to introduce administrators to iShare Maps, including

- an overview of the application and its architecture
- management and administration tasks
- troubleshooting and debugging

## Prerequisites

Delegates should have:

- access to an installed and configured instance of iShare Maps
- access to the Astun training instance

## Course Modules

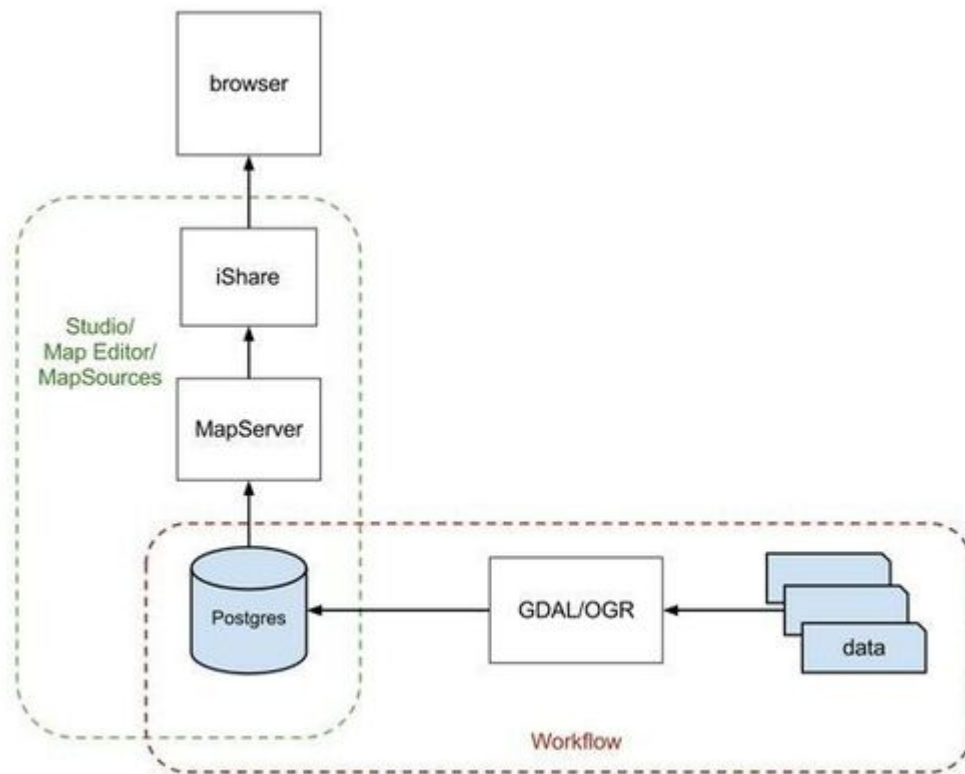
- [IS-1.56: iShare for Administrators](#)
- [ISM-1.56: iShare Maps for Administrators](#)

- 1. Introduction to iShare
  - 1.1. Software Components
  - 1.2. Physical Architecture
  - 1.3. Components
  - 1.4. Summary of Configuration Files
- 2. Studio Overview
  - 2.1. Map Sources
  - 2.2. Data Share Connections
  - 2.3. Workflow
- 3. Using Studio
  - 3.1. Add a New Layer
  - 3.2. Thematic Layers
  - 3.3. Add a Classic Layer
  - 3.4. Layer and Map Source Management
  - 3.5. Legends
  - 3.6. Using HTML in Attributes
- 4. Managing iShare
  - 4.1. Folder structure
  - 4.2. Log Files and Debugging
  - 4.3. MapServer debugging
  - 4.4. Studio Logging

<b>Code</b>	IS-1.56
<b>Title</b>	iShare for Administrators
<b>Description</b>	Covers all aspects of iShare administration.  Topics specific to iShare Maps or GIS are contained in a separate modules, to be added to this one depending on requirements.
<b>Required Software</b>	iShare GIS 5.6.16 or iShare Maps 5.6.16
<b>Target Audience</b>	iShare Administrators
<b>Pre-requisites</b>	None
<b>Duration</b>	.5 day
<b>Version</b>	1.0
<b>Updated</b>	06 Oct 2017
<b>Updated by</b>	<a href="#">Andrew Bailey</a>
<b>Status</b>	Complete

## 1. Introduction to iShare

iShare comprises a number of components, including some open source GIS packages. The diagram below represents the iShare architecture.



## 1.1. Software Components

For iShare to display maps, retrieve information and run spatial queries against features the following stack of open source software is used.

- PostgreSQL/PostGIS - a database to hold spatial and non-spatial data.
- MapServer - to generate map images and respond to information requests on a given location.
- GDAL/OGR - to pipe data from a series of formats into PostgreSQL via Spatial Data Transformation tasks.
- TileCache - to generate base mapping caches.

### PostgreSQL / PostGIS



- PostgreSQL is an object-relational database (<http://www.postgresql.org/>)
- PostGIS is a spatial database extender for PostgreSQL. It adds support for geographic objects allowing location queries to be run in SQL (<http://postgis.net/>)
- PGAdmin is the PostgreSQL/PostGIS Administration GUI (<http://www.pgadmin.org/>)

### OGR/GDAL



- GDAL is a translator library for raster geospatial data (<http://www.gdal.org/>) e.g. transforming aerial photography from ECW format to geoTIF format.
- OGR is a translator library for vector geospatial data. This is used to bring import data into the iShare database e.g. importing council parking zones held in MapInfo or SDE to Postgresql. This removes dependency on external systems.

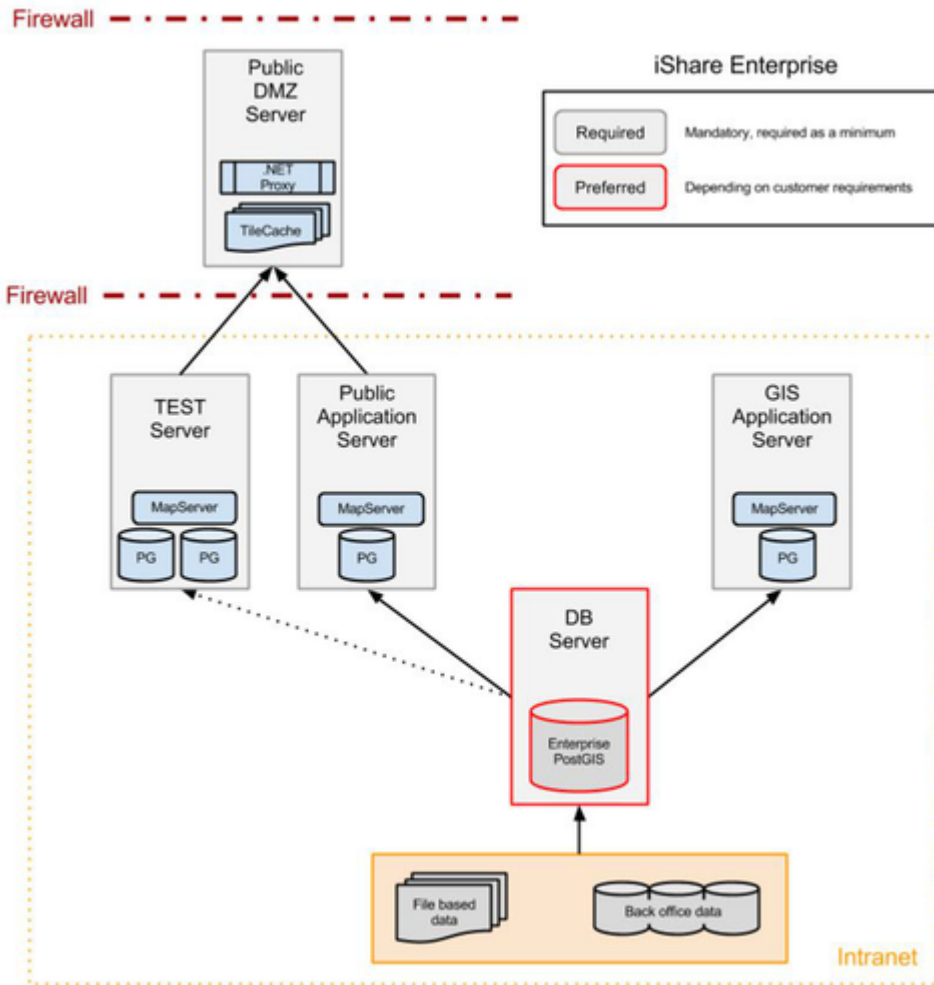
## MapServer



- MapServer is used to render base maps and overlays <http://mapserver.org/>
- MapServer .MAP file documentation <http://mapserver.org/mapfile/index.html>
- Astun Mapserver Styling Guide <https://astuntech.atlassian.net/wiki/display/ISHAREHELP>

## 1.2. Physical Architecture

The diagram below shows a typical iShare installation.



The key components are -

- Front-end - Public DMZ Web Server
- Back-end - Public Application Server
- Separate Database Server - Optional

The iShare Maps Web front end on the Public DMZ server has a web.config file which points to the iShare Maps Web Service on the Public Application Server.

Load up web.config and identify the iShare.config value. In the iShare Maps Training instance this looks like this;

```
<add key="iShare.config" value="http://localhost/iShareLIVE.WebService/" />
```

For iShare GIS the web.config contains the following;

```
<add key="iShare.config" value="http://localhost/iShareGISLIVE.WebService/" />
```

Now that the iShare web application knows where the Web Service is, it communicates via HTTP.

- Web Pages - atMyCouncil.aspx, Solo/Lite Maps, iShareGIS.aspx
- Data Requests - getData.aspx, MapGetImage.aspx (info clicks)

- MapServer Image Requests for overlays - MapGetImage.aspx
- MapServer Image Requests for client-maintained Basemaps - MapGetImage.aspx
- ADS Basemaps Requests are made from the Client web browser to [t0.ads.astuntechnology.com](http://t0.ads.astuntechnology.com)

### 1.3. Components

The following components are installed usually in a second drive e.g. D:\Astun

- Studio - D:\Astun\iShare[GIS]\x.x\Studio\Astun.iShareMaps.Studio.Shell.exe

The same folder contains Astun.iShareMaps.Studio.Shell.exe.config which defines the top-level configuration - the location of the settings.xml. If you have several installations of iShare then this will let you know where Studio is pointing.

- WebService - D:\Astun\iShare[GIS]\x.x\WebApps\WebService

This is the back-end web service application that talks to the Web front-end.

The folder also contains the Config folder which contains the map sources and data sources.

- Web - D:\Astun\iShare[GIS]\x.x\WebApps\Web or a DMZ location

In the case of iShare Maps the web folder would be generally held on a DMZ server. In the case of iShare GIS both the WebService and the Web applications would sit on the same server.

The folder contains the end-point for iShare

- For iShare Maps this would be client styled versions of the atMyCouncil.aspx file and example Solo and Lite Map files. Resources used for client styling would sit in the custom folder.
- For iShare GIS this would be iShareGIS.aspx

- MapServer files - E:\iShareData\<Client>\\_mapserverconfig\\*.map

Each mapsource in Studio is associated with a MapServer .mapfile - generally held in this directory.

### 1.4. Summary of Configuration Files

- Mapsource .xml files under WebServices\config\<client>\
- MapServer .mapfiles under iShareData\<client>\\_Mapserverconfig\
- Settings.xml leads to Data.xml, iShareDataRoot.xml and the DataShare\_<client>.xml.

## 2. Studio Overview

This section provides an overview of Studio functionality, including Map Sources, Layers, Data Share Connections, and Workflow.

### 2.1. Map Sources

Map Sources in iShare define a set of source data, with styling and other configuration, which can be managed independently and used to populate different parts of the iShare interface. Each Map Source is associated with a MapServer .map file, though several Map Sources can use the same .map file.

There are several categories of Map Sources, each of which has its own purpose and characteristics.

- Base Maps - These include background mapping such as OS data, Aerial Photography,

OpenStreetMap and so on. Base Maps are used in all mapping modules except for My House and My Nearest.

- My Maps - These are used in My Maps, iShare Lite, iShare Solo and iShare GIS.

The following Map Sources are used in iShare Maps, so are covered separately if required.

- My House - These are only used in My House and My Nearest.
- Logger - These are only used in the Logger module.

## Base Maps

Any GIS platform needs base maps, and iShare supports numerous types e.g. Aerial Photography, OS MasterMap, Historic Maps etc. Currently the workshop uses the OS OpenData maps available through Astun Data Services.

Base maps will either use a tilecache or a data source for the base mapping. If a tilecache has been defined for a base map that will be used in preference of the source data. Inspect how the Surrey Air Survey map source is defined and note the presence of the tilecache.

The XML files that define base map sources are stored in the `WebService\config\mapsources` folder. For this workshop this can be found here:

```
D:\Astun\iShare[GIS]\LIVE\WebApps\WebService\config\mapsources
```

Each map source will have a corresponding `.map` file pointing to the source data. In the case of the workshop this is

```
E:\iShareData\LIVE\_MapserverConfig.
```

Open Studio and see the existing Base Maps.

Expand the BaseMaps node and choose the OS OpenData map source. If you select this, you will see four tabs in the main window:

- MapSource: Basic information about the map source. Basemap type controls whether it is a WMS source or a legacy source (e.g. a tilecache or a MapServer source)
- Map Editor: A text editor for the MapServer mapfile
- Map Viewer: A simple viewer for the MapServer mapfile
- Metadata: Where you may enter the details of the Catalogue Service for the Web (CSW). This is a web service like WFS or WMS but for accessing Metadata rather than Spatial data.

If you expand the OS OpenData node then you will see two additional nodes below it:

- Start up: Zoom level and boundary settings
- Details: The location of the MapServer mapfile that this source refers to, copyright statement, tilecache URIs (if configured) and zoom scales.

This is where base mapping is configured. Each base map has an associated MapServer `.map` file. Typically clients have an Ordnance Survey and an Aerial Imagery base map.

Sometimes a tilecache will be constructed from a client's own base mapping and is placed on the public facing DMZ server to reduce overheads. Tilecache configuration is beyond the scope of this workshop material but is available through Astun's online help, training sessions or additional consultancy hours.

The workshop base maps include:

- OS OpenData - from Astun Data Services
- OS OpenData BW - from Astun Data Services
- Surrey Air Survey - an aerial imagery base map for Surrey
- No Map - a blank 'no maps' base map

## MyMaps

Any Map Source that you wish to use in iShare GIS or iShare Maps modules (My Maps, iShare Lite maps, iShare Solo maps) will need to be defined under the MyMaps node.

Like the BaseMaps node, clicking on the main map source name will provide access to the Map Source, Map Editor, Map Viewer and Metadata tabs. The MapSource tab contains the following:

- The MapServer mapfile location
- Paths and Watermarking
- BaseMap Sources: the base maps that you want to make available to this overlay - select the one you want to be loaded by default and click **Set default**
- Find Address: The type of address lookup. Note: this is generally configured by Astun and should not need changing from the default

Expand the Astun Maps node to show:

- Start up: Zoom level and boundary settings
- Take Me To: Any short-cut locations defined (applicable to iShare Maps only)
- Layers: The Layer Groups and Layers to be included in the map source
- Roles: Control the users that are allowed to see this map source (applicable to iShare GIS only)

You can set a default map source by right-clicking one of the map sources in the list and choosing "Set as Default MapSource".

## 2.2. Data Share Connections

Data Share Connections are used if you have non spatial data that you wish to use in iShare. You may want to:

- Present the data in My House / My Nearest
- Join the data to an existing spatial dataset
- Spatialise the data, if it contains Easting / Northing or Long / Lat as opposed to geometry type
- Make the data accessible through the iShare API or Publisher

This is where connections to external systems are configured, and selected data is synchronised or copied into the iShare PostgreSQL database. (Data Share is also used within layer configuration, but this will be covered in later sessions).

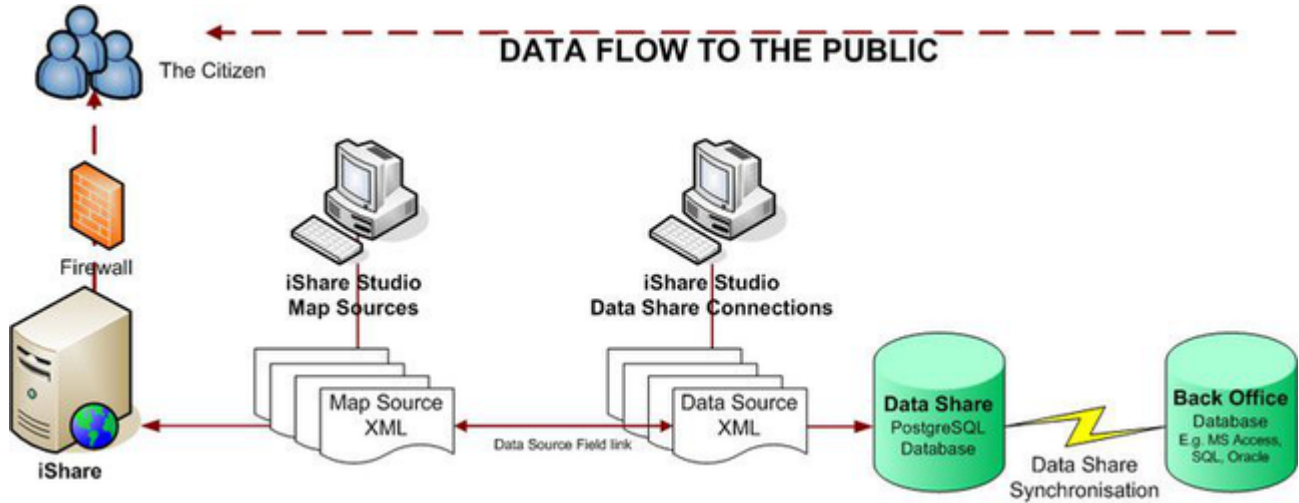
Data Share connections can be to any external database such as Oracle, MS SQL server, Microsoft Access, or flat files such as csv or xls. It is also possible to connect to RSS feeds.

Data Share connections can be placed into groups, for ease of organisation. Groups are configured by selecting the main Data Share Connections node, and clicking **Edit Groups**. This section of Studio also allows configuration of the main iShare database.

Data Share Connections are definitions of data sets that can be used in Map Source layer definitions such as UPRN-based council tax information.

If the data is from an external database or feed the data is replicated locally in PostgreSQL through scheduled synchronisation.





This allows the data to be displayed through iShare Maps or iShare GIS. It is also exposed through the iShare web services.

Data Share Connections can be formed from the following:

- A SQL Select statement to an external database,
- A SQL Select statement to the local PostgreSQL iShareData database,
- The product of a function or command in the local PostgreSQL iShareData database,
- An XML feed such as RSS or GeorSS.

The local PostgreSQL database can be viewed using pgAdmin. It's commonly called "iShareData".

We will now demonstrate different kinds of Data Share Connections, by doing the following:

- Run pgAdmin and connect to the iShareData workshop database.
- Selecting Data Share Connections > Edit will demonstrate the DSN definition to the local iShareData database.
- Select the Council Tax data share. This is used in My House as a data source. The icon indicates it is an internal iShareData select statement and therefore can not be synchronised.
- Select the RSS Feeds > BBC Surrey News RSS. The icon indicates that is an XML feed. The number of records limits what is held in the iShare PostgreSQL database and thus what is displayed in My House. We can synchronise the data share connection manually by right-clicking on the data share.

## 2.3. Workflow

### Workflow Connections

Jobs and Tasks can be configured to retrieve data from databases through Workflow Connections. By default there will be a connection to the iShare PostgreSQL database. For the Workshop database, this is the connection "DataShare".

There might also be connections to external data sources such as Astun Data Services (for NHS Choices or Police Data).

### Workflow Jobs

Studio workflow organises individual Tasks (discussed below) into Jobs, where the Tasks can be ordered, and the Job set to run at a particular time, using Microsoft Task Scheduler.

Any Task that is not part of a Job is placed in the **Unassigned tasks** group.

Right-click on a Job to add a new Task, copy an existing one, remove a Task from the Job, delete the Job, or run it.

Note: Task and Job names must be unique.

## Workflow Tasks

There are four types of task, the most common of which will be demonstrated in more detail later:

- Spatial Data Transformation: transforms spatial data from one type to another
- Stored Procedure: runs a function within the Postgres database
- VB Script: runs a custom visual basic script
- Program: runs any type of executable, such as python scripts, batch files and so on

## Workflow Notifications

Studio can be configured to send notification emails when a scheduled job has failed to run. To configure the Notification Settings, click on the **Workflow** node and then the **Edit** button. This will activate the **Notification Setup** button, where you can add the following:

- From: the email address you wish the emails to come from
- SMTP: the address of the SMTP server
- Subject: the email subject should contain some useful identifiers e.g. '<Council name> iShare Maps Live Server'
- To: the email address you wish the emails to be sent to
- Astun: the Astun email address you wish the emails to be sent to (the default is [notifications@astuntechnology.com](mailto:notifications@astuntechnology.com))

When you have filled these in, you can click the **Test** button to send a test email. Save your settings if you change anything.

## Scheduled tasks

- Scheduled tasks should be generally be set to run in off-peak hours.
- You can list all scheduled tasks using the command line executable **schtasks.exe**, or use **Task Scheduler** in **All Programs > Administrative Tools**.
- Inspect last run results for anything that isn't running correctly.

## 3. Using Studio

### 3.1. Add a New Layer

This section describes how to add a new layer using the festival amenities and boundaries data, to an iShare installation using Studio. The exercise assumes that source data has first been loaded into the PostgreSQL database using a Workflow Task ready to be configured in Studio.

#### Overview

iShare supports two types of layers: OGC Layers (the default), and Classic layers.

OGC Layers read data directly from the database, and are configured in Studio using a simple styling editor, This creates styles in SLD (Styled Layer Descriptor) format, a standard GIS representation of styling used by many other GIS tools.

Classic Layers are configured using a MapServer .map file, which contains both the location and connection details of the source data, and the styling.

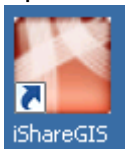
Astun recommends that most layers should be created as OGC layers, but you may sometimes want to use Classic layers. The table below sets out some considerations to bear in mind when choosing whether to use Classic or OGC layers.

Feature	Layers (OGC)	Classic Layers	Comments
Styling	Studio Editor	Mapfile	Styling via the Studio editor is more limited, but uses a graphical interface.
Source	Postgres Layers only	Postgres, Shapefile etc.	If you need to display shapefiles or .TAB files directly in iShare, you'll need to use a Classic layer.
Thematics	Yes	Yes	Styling can be edited in iShare GIS with Classic thematic layers.
Editing in iShare GIS	Yes	No	Classic layers can't be edited in iShare.
Styling based on expressions	Studio	Mapfile	For OGC layers, SLD needs to be manually edited, or created externally (e.g. in QGIS) and copied to iShare.
User filtering in iShare GIS	Yes	No	Allows end user to filter display based on attribute values.

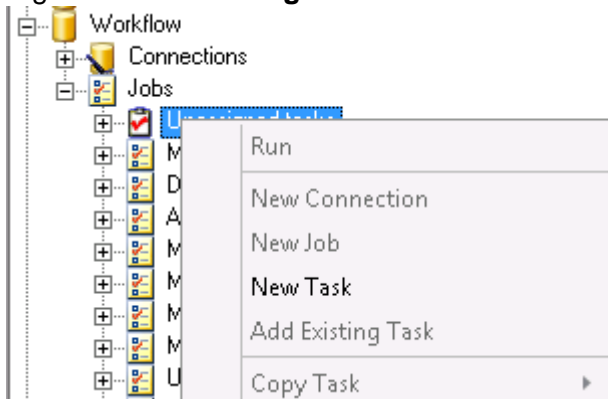
### Importing data into PostgreSQL

The first part of the exercise will take you through the process of importing data into the PostgreSQL database using Studio. You will import a shapefile into PostgreSQL, so that in later stages it can be configured as an iShare layer.

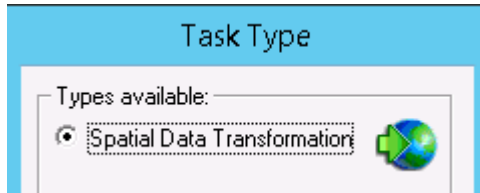
1. Open either iShare Studio or iShare GIS Studio from the desktop as instructed.



2. Expand the **Workflow** node by clicking on the plus beside the node.
3. Now expand the **Jobs** node.
4. Right click on **Unassigned tasks** and select **New Task**.

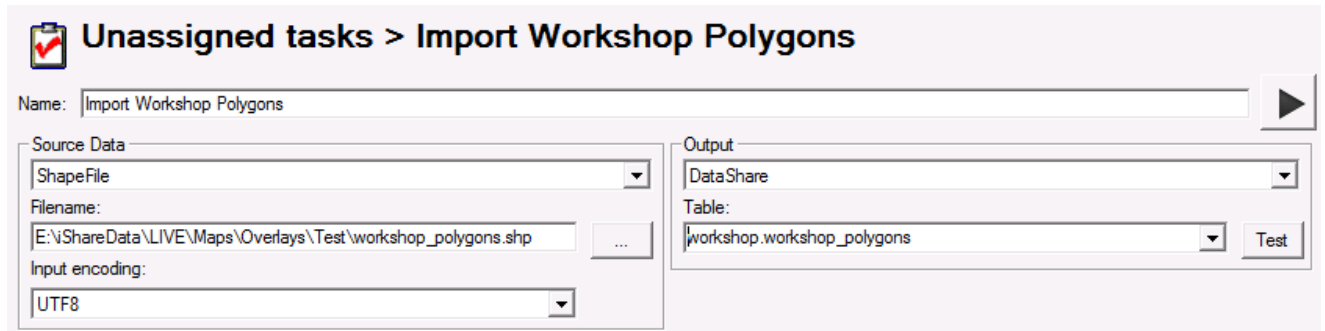




5. Select **Spatial Data Transformation** as the Task Type and click OK.

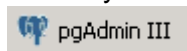


This is because we want to transform the spatial data to load it into Data Share.

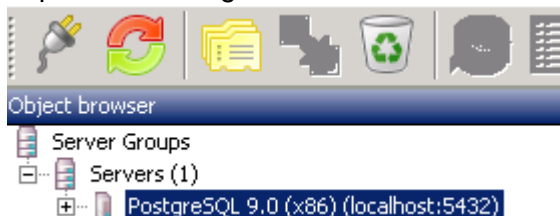
6. First import data from the Shapefiles in **E:\iShareData\LIVE\Maps\Overlays\Test**, selecting the **workshop\_points** and **workshop\_polygons** shapefiles and putting them in the workshop schema of the iShareData database - (this has been configured as the **DataShare** Workflow Connection). Use the same table names as the files.



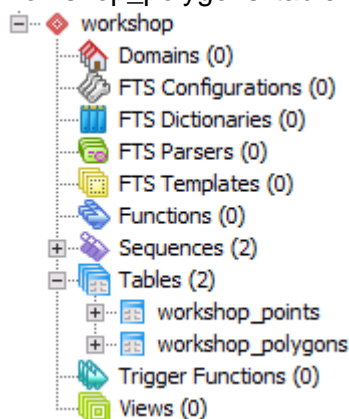
7. Save the changes using the main toolbar 
8. Run the task from the button to the right of the task name   
The data from the shape file has now been loaded into PostgreSQL.
9. Repeat this for both the **points** and the **polygons** data.
10. To verify that the tasks have been successfully completed run pgAdmin from the Windows taskbar



11. Expand the PostgreSQL 9.0 'tree'



12. Expand/ navigate to the 'iShareData' database and look for your new 'workshop\_points' and 'workshop\_polygons' table within the 'workshop' schema

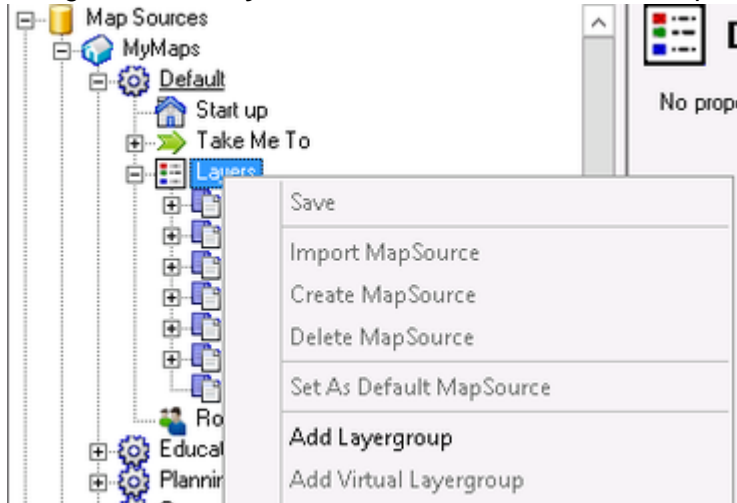


13. Right click on this table and select 'View Data' then 'View All Rows' to view the data.

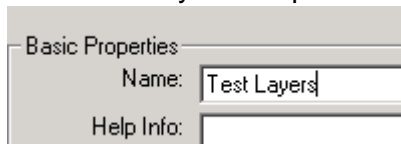
## Create a new Layer Group and Layer in Studio

Carry out the following steps to create a new Layer Group, then a new Layer, in order to make the layer created in the step above available to iShare (GIS).

1. Navigate to the **Layers** section within the **Default** Map Source.

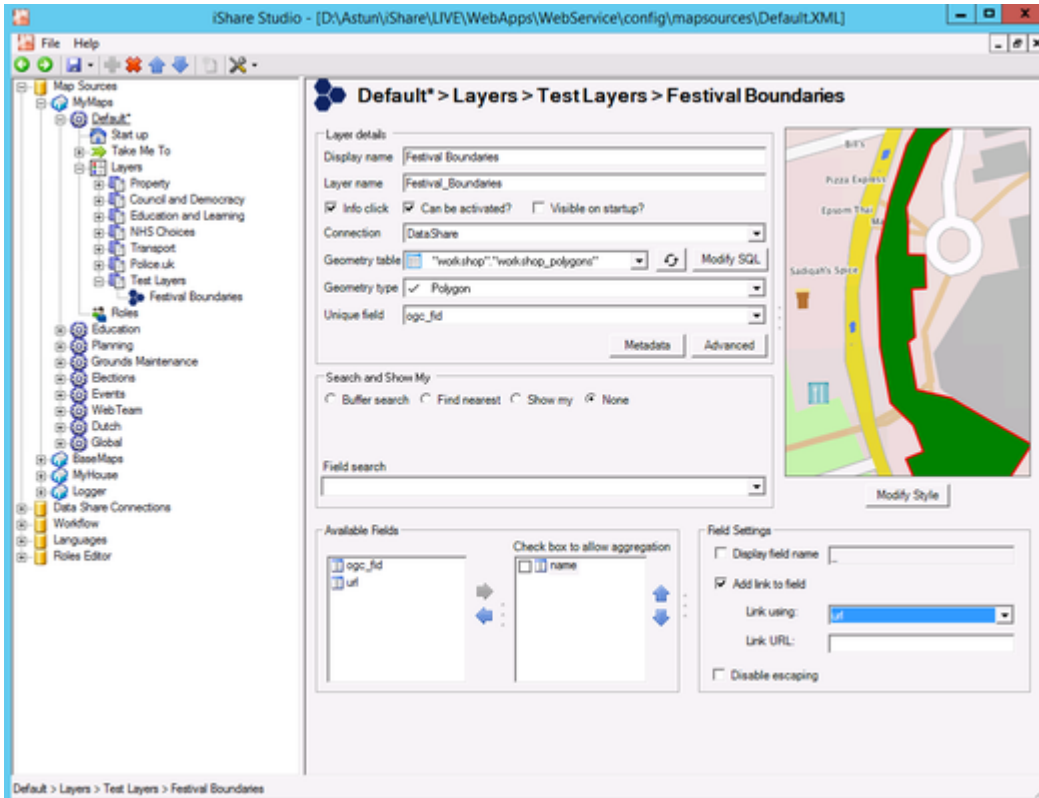


2. Right click on **Layers** and select **Add Layergroup**.
3. Name this Layer Group Test Layers.

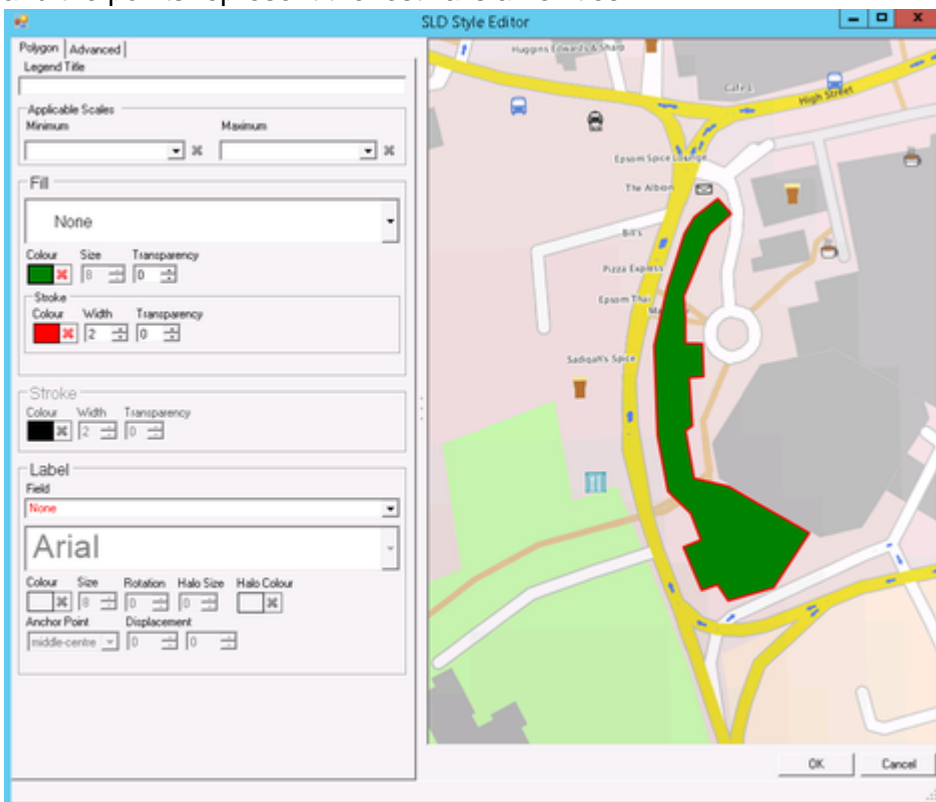


4. Save the changes made in the main toolbar.
5. Right-click on the newly-created 'Test Layers' Layer Group, and select **Add Layer**.
6. Edit the **Display name** (*not* the **Layer name**).
7. Select the underlying dataset by selecting the appropriate **Geometry table** ("workshop"."workshop\_polygons") and make sure that the **Geometry type** has been correctly determined - it should be Polygon.

Note: You can also define a dataset by a SQL select statement using the **Modify SQL** button.



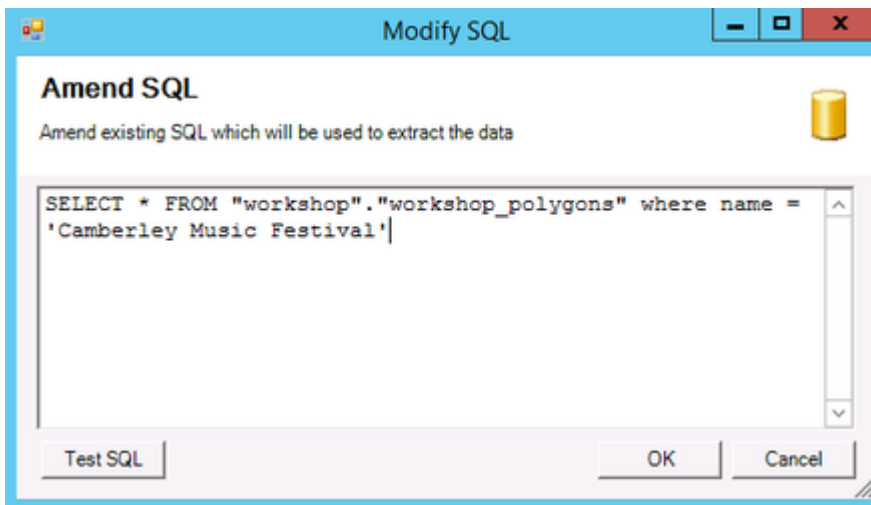
8. Now we will select the fields that we want to display. Double click on the **name** field in the list of Available Fields. Now uncheck **Display field name** and click the **Add link to field** box. Now change the **Link using** entry to pick the **url** field. See above screenshot for details.
9. Use the **Modify Style** button and choose a style that suits the geometry type. Any changes that you make will be shown in the preview window. Our polygon dataset is the basis for a festival boundary, and the points represent the festival's amenities.



Note: The **Advanced** tab allows you to edit the SLD (the code which represents the styling) directly. If you are comfortable working with SLD, and want to tweak the styling in the source code, you could try this - but take a backup of the file before you start. Alternatively, you can generate SLD in other

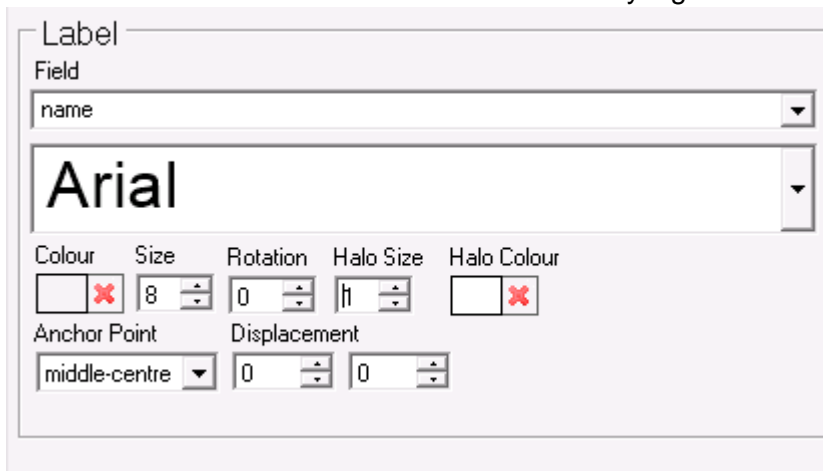
tools (for example QGIS) and copy it to iShare, though it may need some tidying. This will be demonstrated in a minute if required.

The polygon dataset contains two polygons. We now wish to display the 'Camberley Music Festival' polygon and not 'Donkey Derby'. How can we do this? You can modify the SQL and just select the record you require e.g.



Now we need to repeat the process for the **Festival Amenities** (using the "workshop"."workshop\_points").

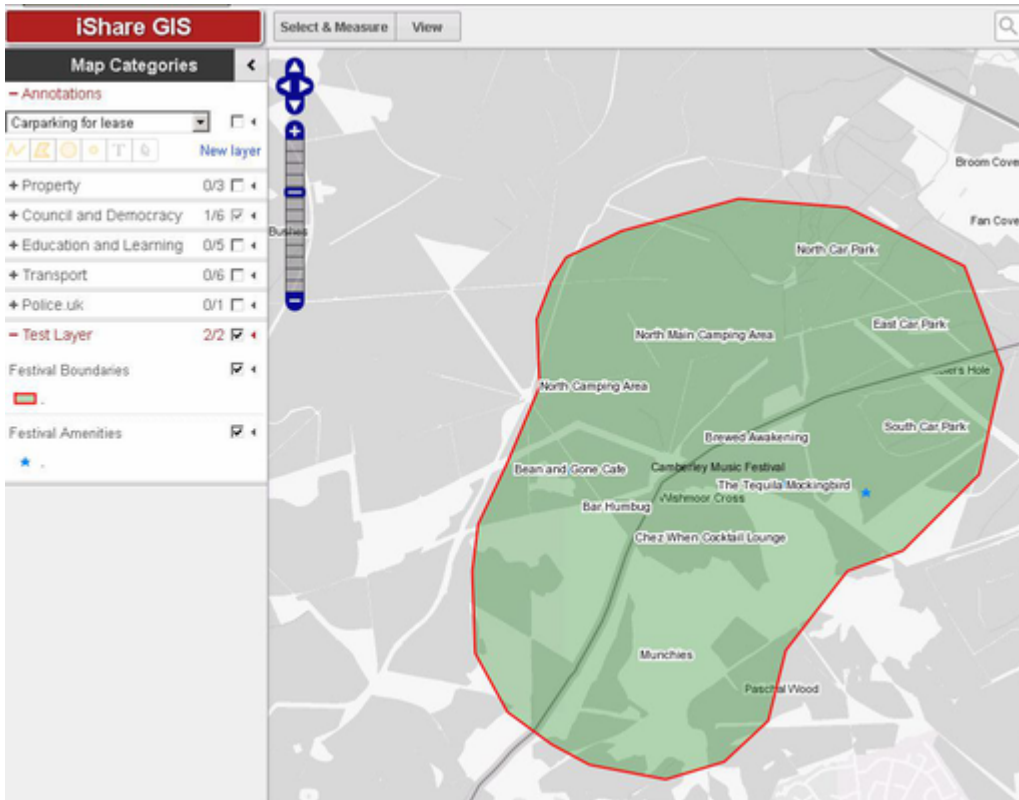
1. Make sure that have have selected both the **name** and **icon** fields from the list of Available Fields.
2. Now click **Modify Style** and pick an icon to be displayed for your points layer.
3. To label the amenities, as we have brought across the **name** field across from the Available fields list, this can be selected as the label source when styling. Create a **Halo** around labels for clarity.



4. If you haven't already done so rename the two layers: Festival boundary and Festival amenities.

View <http://localhost/iShareGISLIVE.web/isharegis.aspx> in a browser and select the default profile. If you are asked for credentials, use the ones you were provided for logging in.

The end result should look something like this (the interface will be slightly different in iShare Maps):

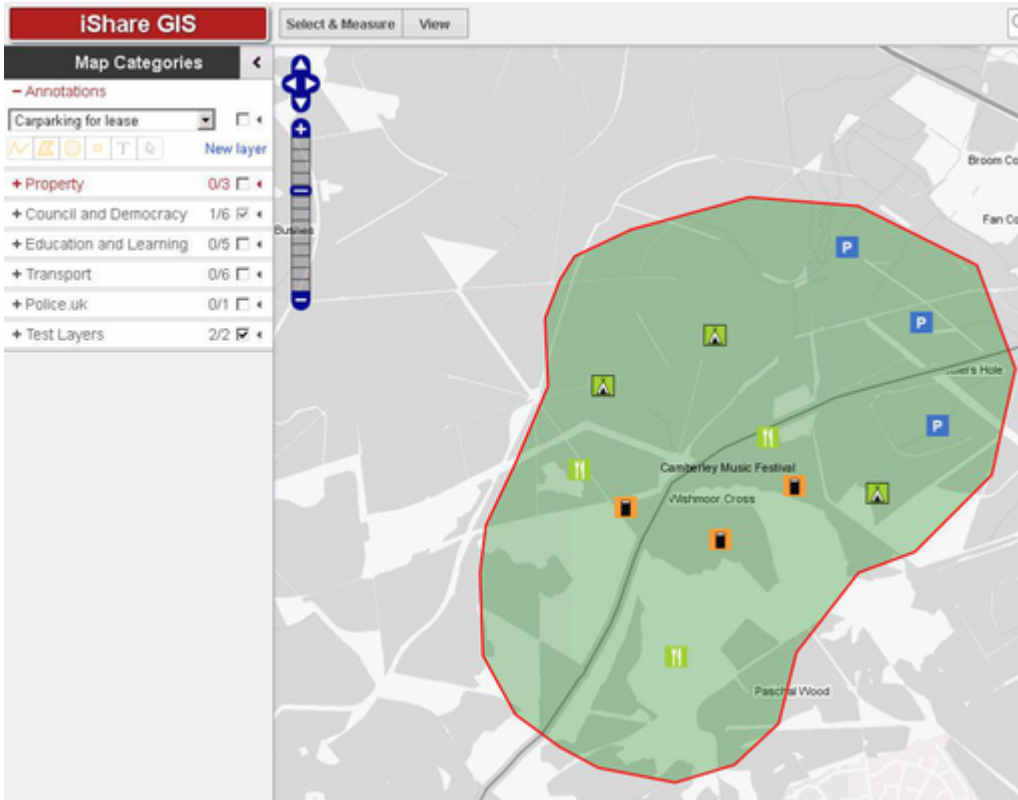


We have icons in our amenities layer, so it would be nice to be able to use these for each of the amenities. This involves using the **Advanced** button in the SLD Style Editor.

In true "Blue Peter" fashion "here is one we made earlier".

1. Expand the **Events** map source in Studio and select the **Festival Amenities** layer.
2. Click **Modify Style** and copy all the rules.
3. Now navigate back to your **Festival Amenities** layer in the Astun Maps **Test Layers** layer group.
4. Click **Modify Style** and select the **Advanced** tab.
5. Now paste the rules that you have previously copied and click **OK**.
6. Note that you must bring the icon field across from the available fields list as this is used in the rules.
7. Save your changes and see the difference.





### 3.2. Thematic Layers

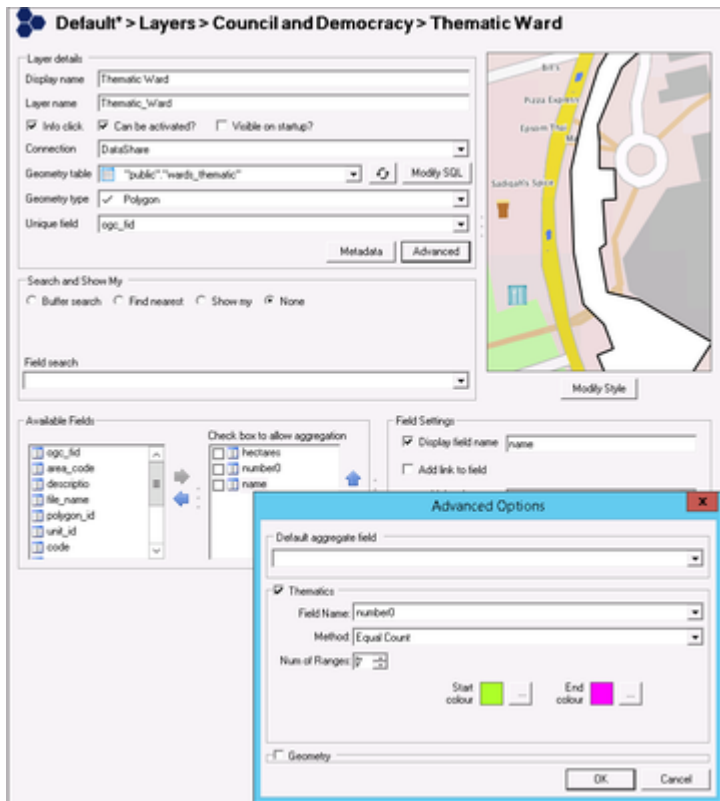
iShare can also represent datasets as thematic layers. Styling is applied to each feature according to the value of a field in the source data - for example, a ward layer could be coloured on a red to green colour ramp based on the number of road accidents in the ward in the previous year.

The editing interface includes settings for:

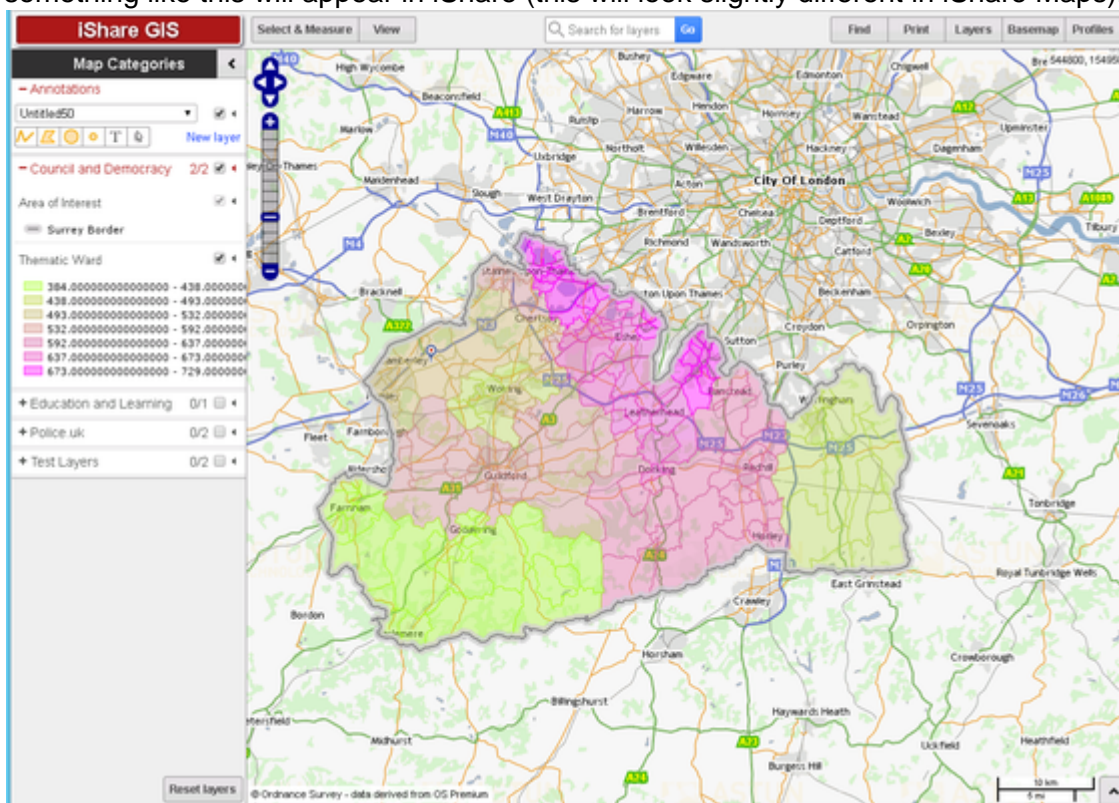
- **Method:** i.e. the way that the values in the range are grouped for each classification, including Equal Range, Equal Count, and Standard deviation
- **Number of Ranges:** the number of classification groups to be used for the values, from three to ten
- **Starting colour:** the starting colour to be used in the colour ramp, chosen from a number of preset colours, or specified with a hex value
- **Ending colour:** the ending colour to be used in the colour ramp, chosen from a number of preset colours, or specified with a hex value
- **Fill opacity:** the opacity of the fill colour used for the polygons, selected using a slider
- **Stroke opacity:** the opacity of the border of the polygons, selected using a slider

#### Creating a Thematic Ward Layer

1. Under the Council and Democracy Layer Group right-click and select 'Add Layer'.
2. Rename the layer (in **Display Name**) as **Thematic Ward**.
3. Select the 'public'.wards\_thematic' as the geometry table and polygon as the geometry type.
4. Select the **hectares**, **number0** and **name** fields from the list of Available fields.
5. Click **Advanced** and check the **Thematics** checkbox.



6. In the **Advanced Options** dialogue, select **Field Name**: number0, **Method**: Equal Count and **Range** of 7, and choose a suitable colour scheme.
7. Save in Studio, and refresh the iShare application in the browser.
8. Select your new Layer from the Layer Catalogue (if using iShare GIS)
9. In the browser select the select Council and Democracy > Thematic Ward layer, and thematic layer something like this will appear in iShare (this will look slightly different in iShare Maps):



### 3.3. Add a Classic Layer

This section describes how to add a new Classic Layer to an iShare installation using Studio. The exercise assumes that source data will first be loaded into the PostgreSQL database using a Workflow Task and then configured in Studio.

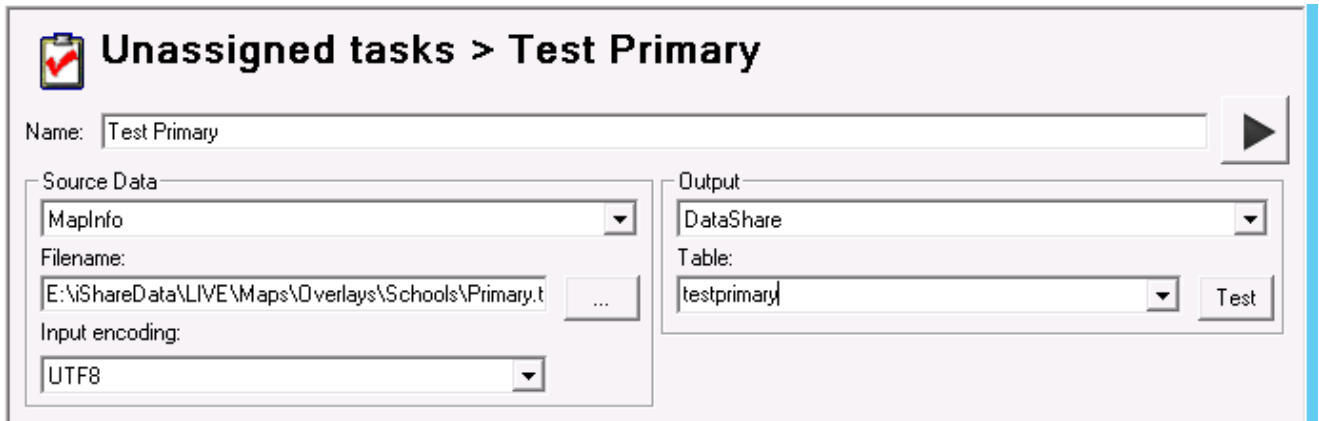
Classic Layers are required if you are presenting data in My House, My Nearest or if you wish to use Clustering as a way of styling your Layer. There is a useful page in the on-line documentation on [Configuring MapServer Cartography](#) which you may find useful.

This exercise creates a layer using a layer definition held in the map source's associated .map file. This is the way that layers were defined before the availability of OGC layers.

### Importing data into PostgreSQL

First we need to import a MapInfo TAB file into PostgreSQL using a Workflow Spatial Data Transformation task as we did earlier, so that it can be configured as an iShare layer.

1. Expand the **Workflow** node by clicking on the plus beside the node.
2. Now expand the **Jobs** node.
3. Right click on **Unassigned tasks** and select **New Task**.
4. Select **Spatial Data Transformation**
5. The New Task dialogue will display, ready for you to enter the appropriate parameters.



6. Enter the following parameters:

<b>Name</b>	Test Primary
<b>Source Data</b>	MapInfo
<b>Filename</b>	E:\iShareData\LIVE\Maps\Overlays\Schools\Primary.tab
<b>Input encoding</b>	UTF8
<b>Output</b>	DataShare
<b>Table</b>	testprimary

When you have completed the dialogue:

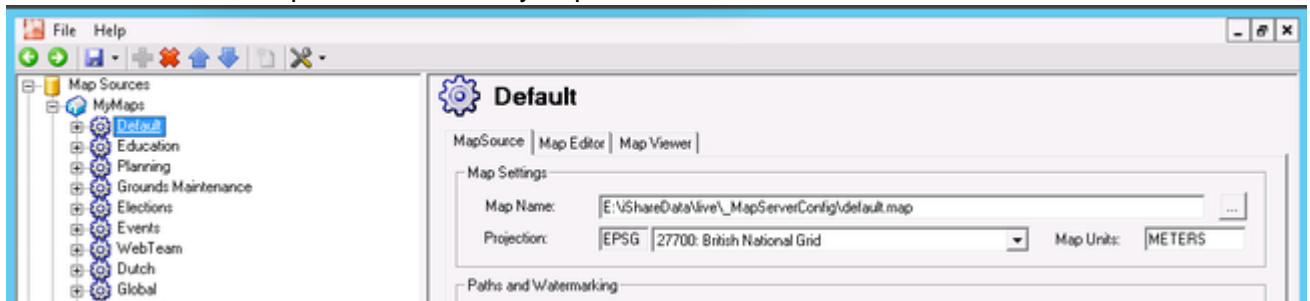
1. Save the changes using the main toolbar.
2. Run the task from the button to the right of the task name.

### Configuring the Layer

Now we have the data within the PostgreSQL database, we need to create a layer in the MapServer mapfile in order to use the data in iShare. In this exercise we will use the Studio Map Editor to first copy and paste an existing MapServer layer definition, and then edit the definition to use the PostgreSQL 'testprimary' data loaded in the previous step.

## Create a MapServer Definition

1. Select the **Default** map source within MyMaps in Studio.



2. Select the **Map Editor** tab.

The Map Editor shows the current MapServer mapfile configuration. In the **Layers** tab below the editing window, the layers defined in the mapfile are listed.

Line	Name
130	transport_road_accidents
186	transport_roadaccidents_clusters
206	crimestreets
387	addresses

3. If you scroll down the map entries you will find a commented out LAYER for "primarytestdata"
4. Remove the # (uncomment) from each of the lines making up the primarytestdata layer as shown in the screenshot below (making sure you select the **LAYER** and **END** tags)

```

366     LAYER
367     #NAME "testprimarytestdata"
368     STATUS OFF
369     TYPE POINT
370     INCLUDE "basestyles.lyr"
371     DATA "test_primary_testdata (select * from testprimary)"
372     METADATA
373         "getting_validation_patterns" "1"
374         ows_title "primary test data"
375         ows_abstract "primary test data"
376     END
377     UNITS METERS
378     CLASS
379         NAME "testprimary"
380         STYLE
381             SYMBOL "circle"
382             COLOR 255 0 0
383             SIZE 20
384         END
385     END
386 END
387

```

5. Now save this layer using the icon in the Map Editor toolbar.

You have now created a new MapServer layer called 'primarytestdata'.

In summary, we've moved the data into the PostgreSQL database with a Workflow Task and we've created

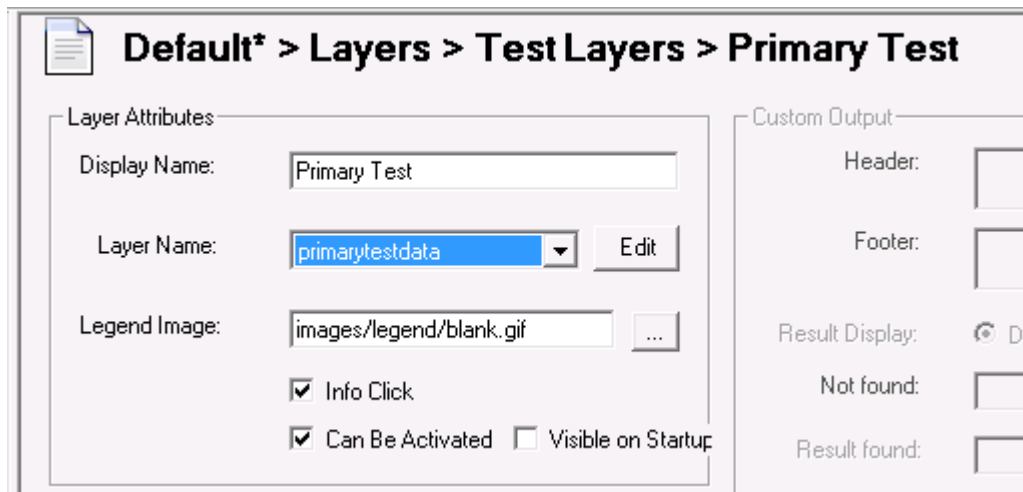
a Layer definition in the map source's associated MapServer mapfile. The next step is to create the layer in the map source.

### Create a new Layer in Studio

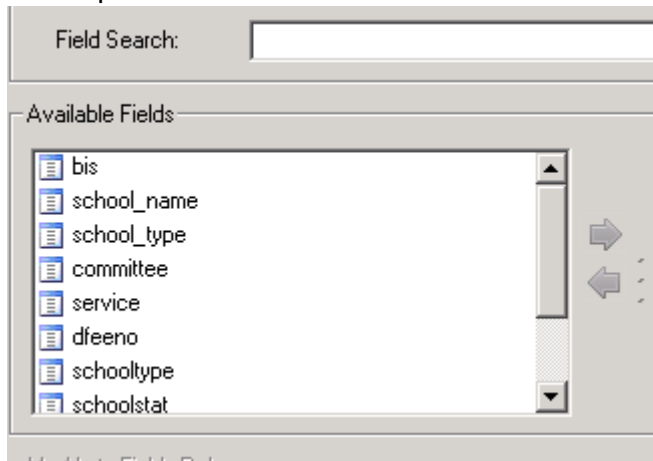
1. Right-click on the **Test Layers** layer group and select **Add Classic Layer**
2. Configure the new layer as follows:

Display Name: Primary Test

Layer Name: primarytestdata (selected from the drop-down menu)



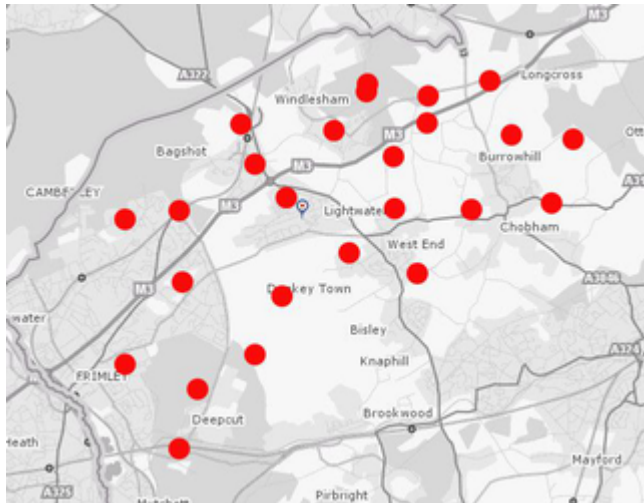
You should now have the 'testprimary' database table column names displaying in the 'Available Fields' panel



3. Double-click on some or all of these fields to move them to the right hand panel; this will allow the content of the fields to be displayed in iShare.
4. Save the changes you have made.

### Viewing the new iShare layer within a browser

1. Refresh your browser using ctrl+f5
2. Select your new layer from the Layer Catalogue (if using iShare GIS)
3. Switch on your new Primary Test Layer



4. Click on one of the symbols to reveal the fields that you selected for your layer.

In summary, you have: (NB: it is worth following the trail of events below within Studio)

- Created a Workflow Task (called 'Test Primary') within Studio to import a MapInfo tab file ('Primary.tab') into a new PostgreSQL table (called 'testprimary')
- Within Studio you then created a MapServer mapfile layer (called 'primarytestdata')
- You then added a new Studio layer called 'Primary Test' (which uses the MapServer 'testprimary' layer), and you then selected a number of fields to display.
- You then viewed your new layer within a browser.

### Other things to explore

The presentation of attribute information can be controlled by adjusting a layer's Field Settings within Studio. It's possible to control which attributes are visible, add aliases to fields and generate hyperlinks.

**Field Settings**

Display field name

Add link to field

Link using:

Link URL:

Disable escaping

We can also control whether a layer should be visible on startup (in conjunction with the **Can Be Activated** button) and whether information should be returned from an info click.

**Layer Attributes**

Display Name:

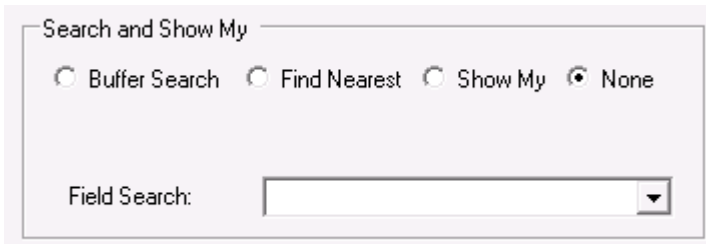
Layer Name:

Legend Image:

Info Click

Can Be Activated  Visible on Startup

The layer **Search and Show My** options allow users to search against a given field, and to perform **Find Nearest** functions.



### 3.4. Layer and Map Source Management

To create a new Map Source, or re-organising Map Sources, you may need to use the the following procedures.

- Copy a Mapsource
  - Right-click on the map source group in Studio and select **Create New MapSource**
- Copy a MapServer .map file
  - Go to mapserver .map files folder (normally in iShareData\LIVE\\_Mapserverconfig), and copy and rename.
  - Note that there are also example mapserver files in iShareData\Astun\\_Mapserverconfig
- Copy Layers between MapSources
  - Close Studio
  - Navigate to the folder where the map source files are held, for example Astun\iShare[GIS]\LIVE\WebApps\WebService\Config\mapsources\, and locate and open both map source xml files in a text editor
  - Copy and paste the whole of the <Layer> element from the source to the destination map source
  - Save the amended map source
  - Open Studio, and check the the new layer is in place.

Note that Layer Groups **cannot** be copied as they contain unique IDs.

### 3.5. Legends

#### Dynamic Legends

In order to use dynamic legends with classic layer, so that an icon for the style of the layer automatically appears in the list of layers in the web interface, you must make sure that you have configured the following.

1. MapServer must be enabled for WMS support.

WMS and WFS require a `NAME` and `PROJECTION` to be defined – British National Grid is EPSG:27700 (WGS84 is 4326), as in the example below. A series of `METADATA` tags need to be defined in the .map file, together with `WEB` and `LAYER` levels to enable this functionality. As the WMS and WFS are both OGC web services the generic tags can often be substituted (i.e. `wms_title` can be used as `ows_title`).

MAP

```
EXTENT 480000 149000 504600 169000 # Tile Cache

FONTSET "D:/mapserver/shared/fonts/fonts.list"
```

```
TEMPLATEPATTERN "."
SYMBOLSET "D:/mapserver/shared/symbols/symbols.sym"
SHAPEPATH "D:/Maps/Workshop/"
SIZE 512 512
MAXSIZE 8192
STATUS ON
UNITS METERS
NAME "Astun Technology Workshop"
IMAGETYPE AGG_Q
OUTPUTFORMAT
    NAME "AGG_Q"
    MIMETYPE "image/png; mode=24bit"
    DRIVER "AGG/PNG"
    EXTENSION "png"
    IMAGEMODE "RGBA"
    TRANSPARENT TRUE
    FORMATOPTION "TRANSPARENT=ON"
END
CONFIG "PROJ_LIB" "D:\mapserver\cgi-bin6.0\proj\nad"
PROJECTION
    "init=epsg:27700"
END
WEB
    METADATA
        ows_title "Astun Technology Workshop WMS Server example"
        ows_enable_request "*"
        ows_srs "EPSG:27700"
        ows_abstract "This is an example WFS server from MapServer"
    END
END
```

2. If you don't already have a LEGEND section in your then you will need to add it e.g.

```
LEGEND
```



```
IMAGECOLOR 255 255 255
KEYSIZE 20 10
KEYSPACING 5 5
LABEL
    SIZE MEDIUM
    TYPE BITMAP
    BUFFER 0
    COLOR 0 0 0
    FORCE FALSE
    MINDISTANCE -1
    MINFEATURESIZE -1
    OFFSET 0 0
    PARTIALS TRUE
END
POSITION LL
STATUS OFF
END
...
```

3. In Studio the **Legend Image** entry for the Layer needs to point to a non-existent file, e.g. images/legend/blank.gif.

4. The . needs to include a NAME entry for each CLASS in the LAYER. Note that this can be blank, for example CLASS " " .

```
...
CLASS
    Expression "CHLK"
    NAME "Chalk"
    STYLE
        OUTLINECOLOR 0 0 0
        WIDTH 2
        OPACITY 60
    END
    STYLE
        COLOR 200 255 175
        OPACITY 80
    END
```

END

CLASS

Expression "CSDS"

NAME "Calcareous sandstone"

STYLE

OUTLINECOLOR 0 0 0

WIDTH 2

OPACITY 60

END

STYLE

COLOR 220 220 200

OPACITY 80

END

END

...

### Using Static Legends

To use Static Legends, that is a fixed legend image, make sure that you have configured the following.

1. In Studio, the Legend Image entry for the Layer needs to be defined as a valid image, e.g. images/carpark.gif.
2. This image must exist in same relative path in the both the Web and WebService folders e.g.

D:\Astun\iShare\LIVE\WebApps\Web\images\carpark.gif

D:\Astun\iShare\LIVE\WebApps\WebService\images\carpark.gif

### 3.6. Using HTML in Attributes

In this example we create some HTML code, which will open a URL in a new window, as a layer's attribute value.

Normally we would select the Link field in Studio and check the checkbox to generate a hyperlink for the field. Studio currently doesn't support the creation of a hyperlink for a new tab/window directly but it does provide an opportunity to see how html can be stored in a layer's field.

The following steps presume that you're using a table in PostgreSQL but this can work for other sources such as a shapefile or MapInfo tab file.

1. Open PgAdmin, and use the SQL dialogue to create a column in your table

```
ALTER TABLE mytable ADD COLUMN myhtml text;
```

2. Populate the column with some HTML that includes the link with the target parameter

```
UPDATE TABLE mytable SET myhtml = '<p><a href="http://www.google.co.uk" target="_blank" >Here's Google in a new window</a></p>';
```

3. Alter the SQL in the MapServer .map file to include the myhtml column as myhtml\_raw - this will ensure that the HTML is passed through without alteration.

```
LAYER

  NAME "mylayer"

  STATUS OFF

  TYPE POINT

  INCLUDE "datashare.inc"

  DATA "wkb_geometry from (select *, myhtml as myhtml_raw from
public.mytable) as foo using unique ogc_fid using srid=27700"

  TOLERANCE PIXELS

  CLASS

    NAME " "

    STYLE

      SYMBOL "circle"

      SIZE 14

    END

  END

END
```

4. In Studio include myhtml\_raw to in the list of columns displayed.

Note: Make sure you include the http:// in the myhtml column - failing to so will result in the browser attempting to load the page as a local resource.

## 4. Managing iShare

### 4.1. Folder structure

iShare is an application constructed from two web applications, 'Web' and 'WebApplication'. Typical locations for key files are as follows.

- Web application
  - iShare Maps: D:\Astun\iShare\LIVE\WebApps\Web\atMyCouncil.aspx
  - iShare GIS: D:\Astun\iShareGIS\LIVE\WebApps\Web\isharegis.aspx (windows authentication is enabled)
- Webservice application
  - D:\Astun\iShare[GIS]\LIVE\WebApps\WebService
- Map sources
  - D:\Astun\iShare[GIS]\LIVE\WebApps\WebService\config\<client name>
- MapServer .mapfiles
  - D:\iShareData\LIVE\\_MapServerConfig
- MapServer resources
  - symbols: D:\MapServer\shared\symbols
  - fonts: D:\MapServer\shared\fonts
  - projection information: D:\MapServer\shared\proj

## 4.2. Log Files and Debugging

### Locations

iShare uses a range of log files to record actions and errors, and these can be useful in diagnosing problems with configuration or other errors. Because iShare uses third-party components, logs are held in a range of forms and locations. This section highlights where the main log files are, and how logging levels may be configured.

Component	Normal Log Location	Comments
MapServer	D:\mapserver\tmp\overlays.map.error.log	Location and logging level set in .mapfile
PostgreSQL	C:\Program files (x86)\Postgresql\9.0\data\pg_log\	Logs are in same location as PostgreSQL data
iShare Studio	D:\Astun\iShare[GIS]\LIVE\Studio\logs	Studio.log (main Studio log) ConsoleApp.log (workflow) iShareDataSyncConsole.log (Data Share)
Web front end	D:\Astun\iShare[GIS]\LIVE\WebApps\Web\logs	iShareMaps.all.log
iShare Maps/GIS	D:\Astun\iShare[GIS]\LIVE\WebApps\WebService\logs	iShareMaps.all.log

Make sure you look for the latest log files, with dates/times corresponding to the time of the actions you are investigating.

## 4.3. MapServer debugging

In order to enable MapServer debugging you need to perform the following steps:

- Set the `MS_ERRORFILE` variable
- Set the `DEBUG` parameter [ OFF | ON | 0 | 1 | 2 | 3 | 4 | 5 ]

### The `MS_ERRORFILE` variable

This variable defines the name and location of the MapServer .log file. The recommended way to set the `MS_ERRORFILE` variable is in your .MAP file, within the MAP object e.g.

```
MAP
...
CONFIG MS_ERRORFILE "..\..\..\mapserver\tmp\overlays.map.error.log"
...
END
```

### The `DEBUG` parameter

If a particular layer is not behaving as expected you can place the `DEBUG` parameter in any LAYER in the .MAP file, or instead, set it once in the MAP object so that it applies to each layer. In the example below a

debug level of 0 is set at the Map level but a layer has received the verbose debug 5 level to track problems.

MAP

```
...
CONFIG MS_ERRORFILE "..\..\..\mapserver\tmp\overlays.map.error.log"

DEBUG 0
...
LAYER

    DEBUG 5
    ...
END
END
```

Use the value of the `DEBUG` parameter to set the type of information returned, as follows:

Level 0 - Errors only (`DEBUG OFF`, or `DEBUG 0`) - only `msSetError()` calls are logged to `MS_ERRORFILE`. No `msDebug()` output at all. This is the default.

Level 1 - Errors and Notices (`DEBUG ON`, or `DEBUG 1`) - Level 0 plus `msDebug()` warnings about common pitfalls, failed assertions or non-fatal error situations (e.g. missing or invalid values for some parameters, missing shapefiles in tileindex, timeout error from remote WMS/WFS servers, etc.)

Level 2 - Map Tuning (`DEBUG 2`) - Level 1 plus notices and timing information useful for tuning mapfiles and applications. this is the recommended minimal debugging level.

Level 3 - Verbose Debug (`DEBUG 3`) - Level 2 plus some debug output useful in troubleshooting problems such as WMS connection URLs being called, database connection calls, etc.

Level 4 - Very Verbose Debug (`DEBUG 4`) - Level 3 plus even more details...

Level 5 - Very Very Verbose Debug (`DEBUG 5`) - Level 4 plus any `msDebug()` output that might be more useful to developers than to users.

## 4.4. Studio Logging

The name of the `.log` files and the type of logging in Studio is controlled by the `log.config` files, typically in the `D:\Astun\iShare[GIS]\5.4\Studio` folder, corresponding to the log files detailed earlier. If you open these in a text editor, you will see that there is an `<appender>` section which takes care of the name and location of the log file together, with the process of rolling over files by size or date. In the example below when, the log file reaches 100000 in size, a new file will be generated and a maximum of 1000 log files will be kept.

```
<appender name="RollingFileAppender"
type="log4net.Appender.RollingFileAppender">
    <param name="File" value="logs\Studio.log" />
    <param name="AppendToFile" value="true" />
    <param name="MaxSizeRollBackups" value="10" />
    <param name="MaximumFileSize" value="100000" />
    <param name="RollingStyle" value="Size" />
```

```
<param name="StaticLogFileName" value="true" />
<layout type="log4net.Layout.PatternLayout">
    <param name="ConversionPattern" value="%d %-5p %c %m%n" />
</layout>
</appender>
```

Note: The file suffixed .log will always be the current log file. Any log files which have been “rolled over” will be renamed LogfileName.n where n is an incrementing number with the largest number being the oldest log file.

Following the `<appender>` section you will see various `<logger>` sections controlling the name, level and appender-ref for different areas e.g.

```
<logger name="Studio">
    <level value="INFO" />
    <appender-ref ref="RollingFileAppender" />
</logger>
```

The level value can be:

ALL | DEBUG | INFO | WARN | ERROR | FATAL | OFF

Where ALL is the most verbose down to OFF where no logging is performed.

<ol style="list-style-type: none"> <li>1. The iShare Maps Front End             <ol style="list-style-type: none"> <li>1.1. My Maps                 <ul style="list-style-type: none"> <li>Searching against a Layer</li> <li>Find Nearest</li> <li>Layer visibility</li> <li>Query features</li> <li>'Take Me To' locations</li> </ul> </li> <li>1.2. My House</li> <li>1.3. My Nearest</li> </ol> </li> <li>2. iShare Maps Tasks             <ol style="list-style-type: none"> <li>2.1. My House Examples                 <ul style="list-style-type: none"> <li>Point-in-Polygon</li> <li>Data Share</li> <li>MiniMap</li> <li>Location Free</li> </ul> </li> <li>2.2. My Nearest Example</li> </ol> </li> <li>3. Embedded Maps             <ol style="list-style-type: none"> <li>3.1. The Lite Map</li> <li>3.2. The Solo Map</li> <li>3.3. LocalInfo Data Requests</li> </ol> </li> </ol>	<p><b>An Astun Technology iShare Training Module</b></p> <table border="1"> <tr> <td><b>Code</b></td> <td>ISM-1.56</td> </tr> <tr> <td><b>Title</b></td> <td>iShare Maps for Administrators</td> </tr> <tr> <td><b>Description</b></td> <td>Topics specific to iShare Maps, including My House, My Nearest.</td> </tr> <tr> <td><b>Required Software</b></td> <td>iShare Maps 5.6.0</td> </tr> <tr> <td><b>Target Audience</b></td> <td>iShare Administrators</td> </tr> <tr> <td><b>Pre-requisites</b></td> <td>iShare for Administrators</td> </tr> <tr> <td><b>Duration</b></td> <td>2 hours</td> </tr> <tr> <td><b>Version</b></td> <td>1.0</td> </tr> <tr> <td><b>Updated</b></td> <td>19 Sep 2017</td> </tr> <tr> <td><b>Updated by</b></td> <td><a href="#">Jo Cook</a></td> </tr> <tr> <td><b>Status</b></td> <td>Complete</td> </tr> </table>	<b>Code</b>	ISM-1.56	<b>Title</b>	iShare Maps for Administrators	<b>Description</b>	Topics specific to iShare Maps, including My House, My Nearest.	<b>Required Software</b>	iShare Maps 5.6.0	<b>Target Audience</b>	iShare Administrators	<b>Pre-requisites</b>	iShare for Administrators	<b>Duration</b>	2 hours	<b>Version</b>	1.0	<b>Updated</b>	19 Sep 2017	<b>Updated by</b>	<a href="#">Jo Cook</a>	<b>Status</b>	Complete
<b>Code</b>	ISM-1.56																						
<b>Title</b>	iShare Maps for Administrators																						
<b>Description</b>	Topics specific to iShare Maps, including My House, My Nearest.																						
<b>Required Software</b>	iShare Maps 5.6.0																						
<b>Target Audience</b>	iShare Administrators																						
<b>Pre-requisites</b>	iShare for Administrators																						
<b>Duration</b>	2 hours																						
<b>Version</b>	1.0																						
<b>Updated</b>	19 Sep 2017																						
<b>Updated by</b>	<a href="#">Jo Cook</a>																						
<b>Status</b>	Complete																						

## 1. The iShare Maps Front End

This section will demonstrate what the user can see and do with the standard client interface for iShare Maps. The URL for this in the standard Astun training instance is <http://localhost/iShareLIVE.Web/atMyCouncil.aspx>. A bookmark has placed in the bookmarks toolbar in Firefox.

Before we look at configuration, we will run through the iShare Maps user interface to give you an idea of what is available to end users.

### 1.1. My Maps

My Maps provides access to all the spatial layers defined in the Map Sources. This includes the ability to:

- **search** against map layers,
- conduct radial searches against a layer and a known location (**Find Nearest**)
- **turn layers** on and off

- **query** features by clicking on the map
- **jump** to defined map views (useful for popular locations or big events)

### Searching against a Layer

Several layers have been configured for searches in the demo instance - configuration usually makes use of a street address or the name of a facility such as school name. For example we can search on the site in:

- Parishes and Wards (use 'west end')
- Schools (use 'hill')
- Airports (use 'heathrow')
- Railway Stations (use 'guildford')

### Find Nearest

A user can provide a location, which is then displayed as the current address, and from this point they can make use of the **Find Nearest** functionality. This will return a list of the nearest features from a specified layer. The user can restrict results by maximum search distance, and by the number of results.

We can see Find Nearest in action using an address (such as 'GU18 5QR') and the following layers:

- Primary schools (use a small radius)
- Secondary schools (use a wider radius)
- Cycle Networks (use the maximum radius)

Note that Cycle Networks is a line geometry layer - Find Nearest operations work against polygons and lines as well as point-based layers.

### Layer visibility

Users can control which layers are displayed on the map by selecting a layer using its associated checkbox. Note that we can configure whether a layer is visible at start up, whether it is on all the time and whether a user can make info click requests against it.

To demonstrate this, we will:

- Switch / off any layer
- Look at the Council and Democracy layer group - the Area of Interest layer is always on

### Query features

Users are able to click on the map and retrieve information about the layers that are present in that location. In iShare Maps, an administrator can configure which layers information is returned for, which fields from each layer are returned, and how the information is presented to the user.

### 'Take Me To' locations

An iShare administrator can create a series of bookmarks for a map. This enables the user to jump to a particular location with a given set of layers switched on. This is useful for occasions such as sporting events, where information such as car parking and toilet locations are required.

## 1.2. My House

My House uses the same spatial data as is used in My Maps, but in a form that is more relevant for an



individual resident. Most of the data is location-based and relies on the address provided by the user to extract and filter the information.

Note: to view My House information a location is required - use 'GU18 5QR' as a postcode for this workshop.

We will look at some examples of My House functionality in action.

**Polygon:** The county councillors information is based on the ward data. This is a polygon dataset, and iShare returns the information that relates to the resident's location. This is a **point in polygon** operation.

**Nearest:** The Transport information uses point locations, and iShare returns the nearest railway station to the resident's location. This is a **find my nearest** (or distance) operation.

**Data Share:** The council tax band information is stored without geometry, while the tax bands relate to UPRNs (Unique Property Reference Number) in a table. iShare looks up the tax band based upon the UPRN of the resident's location.

**Mini Map:** The Property Mini-Map and the Aerial Overview Mini-Map both make use of location parameters exposed by iShare, in this case Easting and Northing coordinates, passing them to MapServer as part of image requests.

**Location Free:** The BBC News feed is non-spatial and has no location identifier. iShare takes this feed and re-publishes it - in this case by limiting it to the latest five items.

Using the side panel, users can control the information which is presented to them. This selection is stored by the browser as a cookie, so the next time the user visits the site they are only presented with the information they have previously selected.

### 1.3. My Nearest

My Nearest provides information on the nearest facilities to a resident's location. Categories can be expanded and collapsed.

For example, the **Education and Learning** category shows the nearest schools to a resident's location, and if the school website is included in the data, a link is created using the school name. The distance to each school is displayed, as well as a link to My Maps. Distance has been limited to 1.5 km for Primary Schools, while other school types have a maximum distance of 5 km.

Note that My House and My Nearest use different map sources - they can hold different groups of information and be independently configured, even though they may use the same underlying data sources.

## 2. iShare Maps Tasks

### 2.1. My House Examples

#### Point-in-Polygon

We'll have a look at a point in polygon layer in Studio.

1. Open iShare Studio if it's not already open.
2. View the County Councillors Layer in Studio by selecting Map Sources > MyHouse > My House > Elected Members > Your County Councillor. Note that the Layer definition in the myhouse.map file is 'county\_councillors'.
3. Open the myhouse.map file in Studio by selecting Map Sources > MyHouse > My House and clicking the **Map Editor** tab.
4. Find the Layer definition for the 'county\_councillors' layer. Note that the data is the **moderngov.spatial** table in the local PostgreSQL iShareDataAstun database.

5. Open QGIS, and connect to the moderngov.spatial table, using the iShareDataAstun connection.
6. Use the info tool to look at the HTML content in the **full\_html** attribute.

Note that:

1. MapServer will automatically alter HTML held in information requests when the user clicks on the map feature, meaning that it can't be used in a web page. To stop this, we use the '\_raw' suffix in the data definition in the myhouse.map file.
2. In Studio, the 'full\_html\_raw' and the ward name are selected to be visible for My House.
3. Under 'Search and Show My' options the 'Show My' option is selected. This means that iShare will determine which ward polygon the user's location falls in, and will generate the HTML accordingly.

## Data Share

The next layer uses a data share connection.

1. Examine the Council Tax Layer under the Property Details Layer Group. The data comes from a Data Share Connection called 'council\_tax'.
2. The data is indexed by UPRN, which is known to iShare from the user's location.
3. The Data Share is identified in the layer definition as a Property-based lookup datasource.
4. Data Share Connections are added to the Map Source by selecting **Lookup Settings** under Map Sources > MyHouse > My House.

## MiniMap

The MiniMap shows a small version of one of the base maps, with parameters to determine size, zoom etc.

1. Select the Mini Map virtual layer under Map Sources > MyHouse > My House > Property Details.
2. Open the Row Settings for the row by double clicking on it.
3. The row settings makes use of the user's location (easting and northing) to generate a MiniMap and the UPRN is also displayed (referenced as the uid).

The code for the row is shown below.

```
<dl><dd id="MiniMap">
<br /><strong>Unique Property Ref (UPRN)</strong><br />##uid##, E:
##easting##m, N: ##northing##m</dd></dl>
```

## Location Free

We'll look at a layer using a non-spatial data source.

1. In Studio, look at the **BBC Surrey News RSS** Data Share connection in the **RSS Feeds** group, and note that the feed is limited to 5 records.
2. Look at the **BBC News** layer under the **Feeds** layer group.
3. Note that the lookup data source is row-based and references the BBC Surrey News RSS data share.
4. The **Row Settings** field holds the HTML for each row in the Data Share. Double hashtags define the field names in the Data Share definition.

## 2.2. My Nearest Example

My Nearest works in much the same way as My House, with the content focussed on nearest amenities to

the user's location. The information returned is displayed in expandable panels.

In the web application, go to the **My Nearest** tab.

- Notice the use of images in the **Education and Learning** layers.
- Notice the wider search distance for Secondary Schools.

The transportation datasets are point locations, and so these can be used in 'find my nearest' (or distance) operations.

Create a point layer which uses a distance operation to display results.

1. Identify the **Transport** layer group under Map Sources > MyHouse > My Nearest.
2. The data for the Train stations comes from the transport schema in the iShareData Astun database.
3. Add a new Classic Layer called **Heliports** in this Layer Group that makes use of the 'transport\_heliport' layer definition in the myhouse.map file.
4. Select **Find Nearest** and choose a very large search distance and a maximum number of results to be listed.
5. Select **name** as a field to display and turn off the display field name option.
6. Save your changes.
7. Run iShare Maps in a browser: <http://localhost/iShareLIVE.Web/atMyResponsiveCouncil.aspx>.
8. Use the **My House** tab and enter 'GU18 5QR' as the location.
9. Observe the Heliport information in the **My Nearest** tab.

## 3. Embedded Maps

### 3.1. The Lite Map

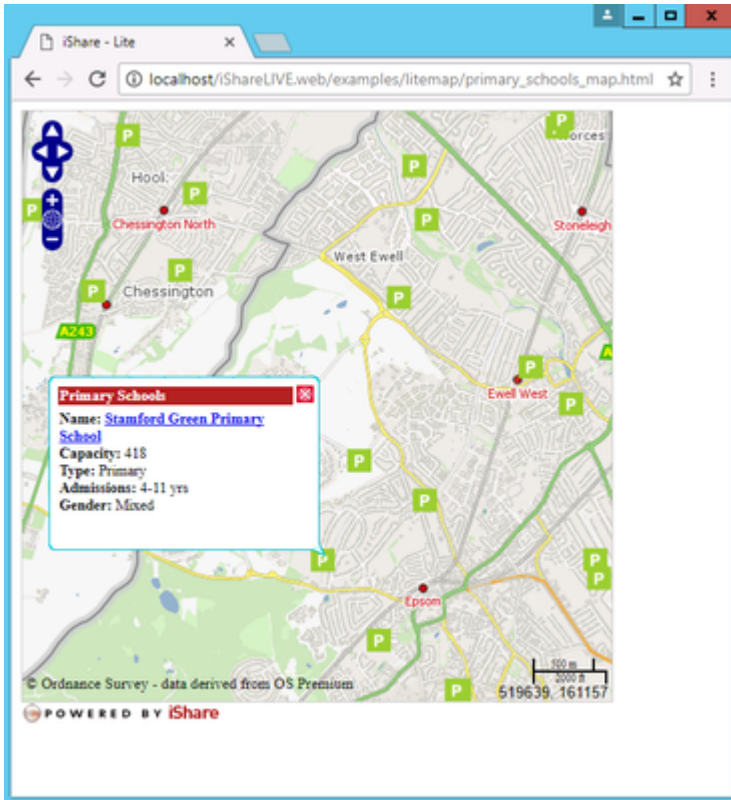
The Lite Map is a simple interactive map that can be constructed with a few parameters. The user can pan, zoom and click on map features for further information. The web designer can create a Lite Map in a matter of minutes using a few lines of JavaScript.

- Demonstrate the MapSource naming convention

Note: the Lite Map parameters include a MapSource: this is the relative path and file name of the XML file that defines our Map Source. For this workshop, the Map Source is **mapsources/Default**. This is because our Workshop Map MapSource definition is held by the D:\Astun\iShare\LIVE\WebApps\WebService\config\mapsources\Default.xml.

Exercise: Create a Lite Map for Primary Schools

1. Copy D:\Astun\iShare\LIVE\WebApps\Web\examples\litemap\atLiteMap.html as primary\_schools\_map.html.
2. Open D:\Astun\iShare\LIVE\WebApps\Web\examples\litemap\primary\_schools\_map.html in a text editor.
3. Check that the mapSource is 'mapsources/Default' and the layers variable is 'Primary\_Schools\_OGC'.
4. Save.
5. Run [http://localhost/iShareLIVE.Web/examples/litemap/primary\\_schools\\_map.html](http://localhost/iShareLIVE.Web/examples/litemap/primary_schools_map.html)



### 3.2. The Solo Map

The Solo Map provides the same panel as in iShare MyMaps. This time the web designer is able to control which panel features appear.

- Demonstrate the Solo Map
  - Run <http://localhost/ishareLIVE.Web/examples/solomap/atSoloMap.html> in a browser.
  - Explain Solo Map takes the same optional parameters as the Lite Map plus the options to rename / remove panels.

### 3.3. LocalInfo Data Requests

The LocalInfo Data Request controls allow users to request information directly from iShare in order that it may be displayed in context on their website. You can choose to stream JSON or JSONP [as well as the old RSS].

Exercise: Investigate LocalInfo requests & responses

1. The first step is to configure Studio for LocalInfo Requests. Open Studio and click on the Settings icon. On the General Settings tab scroll down until you can see the Web section. There are two entries that require attention.
  - a. MapPage = the URL for your iShareMaps.
  - b. MapSource = the MapSource that you wish to use.
  - c. Now scroll down a little further to the WebService section. Here there is just one entry requiring attention.
    - RSSXL = set to atLocalInfoJSON.xsl to stream JSON.

Now the LocalInfo Settings have been configured you are ready to browse to the sample page.

1. Run <http://localhost/ishareLIVE.Web/examples/localinfo/localinfo.html> in a Firefox browser. Click on Firebug in a separate window.
2. In the address search box leave the easting,northing as 518000,161000.

3. Choose **Default** as the MapSource, select **Council and Democracy** as the Layer Group, and **Councils** as the Layer.
4. The URL below shows the example form of localinfo request, constructed from the parameters chosen.
5. The boxes below show the results in both json and html format.
6. The html-formatted results include a link out to an iframe showing a Lite Map.