

## **Displaying Linked or Embedded Files in QS-Pro**

QS-Pro includes, as standard, the ability to link or embed various files into the Batch View to display, for example, drawings, photographs, or working instructions. These will be presented to the process operator while he is collecting measurement data. While linking/embedding is fairly straightforward, it can be difficult to actually see the linked object if the host PC does not have the right server software installed, and correctly installed, at that! In this case 'server software' means the software application that naturally recognises and handles the type of file you have embedded. For example, Windows' PaintBrush for a bitmap (.Bmp) drawing file, Microsoft Word or WordPad for a document (.Doc) file.

On its own CADAR QS-Pro does not have any knowledge of the dozens of formats for picture, document, or spreadsheet files and therefore cannot display them without help. The help comes in the form of Microsoft's OLE technology where Windows itself remembers associations between different types of file and their server applications. When you double click on a bitmap file in Explorer or 'My Computer' and Paintbrush launches, this is an example of OLE at work.

Normally when you install a server application the Setup program will install all the appropriate registry links to make the OLE technology work seamlessly with it. However, if you add or modify such links by hand with the Folder Options dialog in Windows Explorer it is all too easy to miss something out. In fact, Explorer usually does miss out something fairly vital and leaves you staring at an icon instead of a picture or document.

This application note describes the registry entries necessary to tie an OLE object to a server application, and will use as an example, the Microsoft PhotoEditor displaying .Jpg picture files.

### **How OLE Objects and Servers are Tied Together**

Each server application has what is known as a 'Class ID', a long multi-digit hexadecimal code that uniquely identifies it. Servers record their Class IDs in a registry entry under

```
HKEY_LOCAL_MACHINE/Software/CLASSES/<ServerName>.
```

Thus Microsoft PhotoEditor records its Class ID under

```
HKEY_LOCAL_MACHINE/Software/CLASSES/MSPhotoEd
  CLSID = "{11943940-35DE-11CF-953E-00C08A4029E9}"
```

This entry also contains another entry, the current version number of the server app...

```
CurVer = "MSPhotoEd.3"
```

We then go and look under this to find all the up to date information about this server.

```
HKEY_LOCAL_MACHINE/Software/CLASSES/MSPhotoEd.3
  CLSID = "{11943940-35DE-11CF-953E-00C08A4029E9}"
```

The Class ID is the same as before but here there is also information about the server capabilities of the application. These are the sort of commands (often known as OLE verbs) that the server can respond to in respect of a file: things like 'open', 'print', 'edit' etc....

```
Shell
  Open
  Print
  Printto
```

Each command has it's own 'command' entry which shows the applications command line – it's own path and any parameters it requires using MS-DOS batch file type notation: '%1' etc.

For the open command/verb we find...

```
Command...
  Default = "C:\Program Files\Common Files\Microsoft Shared\
  PhotoEd\PhototEd.Exe "%1""
```

For print we find...

```
Command...
  Default = "C:\Program Files\Common Files\Microsoft Shared\
  PhotoEd\PhototEd.Exe /p "%1""
```

And for printto...

```
Command...
  Default = "C:\Program Files\Common Files\Microsoft Shared\
  PhotoEd\PhototEd.Exe /pt "%1" "%2" "%3" "%4""
```

All these will normally have been installed into the registry when the server app was installed.

## File Extensions

At the other end of the chain each type of file extension that is to be used with some sort of server application must also be supported by a registry entry. These are also found under HKEY\_LOCAL\_MACHINE\Software\CLASSES, each one under it's own extension name (including the dot), hence...

```
HKEY_LOCAL_MACHINE\Software\CLASSES\.jpg
```

...which will include at least the entry...

```
Default = "jpegfile"
```

..or something similar. This is the 'file type' for the extension. It is possible for more than one extension to have the same file type. All this means is that each of the extensions that share the same file type will be treated in the same way by the same server app.. The file type is usually (but not always) made up from the file extension and 'file'

Most extension entries also have a Content Type variable which reminds you what the file contains. This is not significant here tho it is the same as the content type indicated in the Folder Options dialog in Explorer. For example...

```
Content Type = "image/jpeg"
```

## File Type

The file type is the link between the extension and the server app, and each file type in the system also has an entry under HKEY\_LOCAL\_MACHINE\Software\CLASSES. Eg...

```
jpegfile
  CLSID
  DefaultIcon
  Shell
```

The CLSID is the same as the server app's CLSID entry. The Shell item is similar to the server app's Shell item and includes all the server's commands that are relevant to this file type.

The DefaultIcon entry contains the path to the icon file, usually the server app itself, and the index to the icon in that file. Eg...

```
DefaultIcon = "C:\Program Files\Common Files\Microsoft Shared\
PhotoEd\PhototEd.Exe,4"
```

When a server app is correctly installed you can expect all these registry entries to be created. However if you have created or modified entries using the Folder Options dialog in Explorer, it is likely that the File Type information will be sparse or missing. In particular Explorer seems to forget about the need for the server app's CLSID value!

In summary – the CLSID ties one or more File Types to a server application and each file extension must belong to a File Type to be linked to a server.

NB. There are file extension entries under both HKEY\_LOCAL\_MACHINE\Software\CLASSES and also under HKEY\_CLASSES\_ROOT. These appear to be the same physical registry entry since deleting an extension in one key also deletes it in the other. Similarly creating a new extension in one key also creates it in the other.