Detecting Upload, Lookup and Triggers

A profile starts because:

i) it is scheduled to do so,
ii) an external activity causes it to do so.

An external activity could be:

- a user on the ALCW web site, with permission and access, clicks the trigger icon in that profile's detail page, or uploads a file to that profile in that page, or manually runs the process either in immediate mode or requests a background run,
- the profile is triggered by another running profile

A profile that begins to run, by whatever method, might need to detect 'why' it is running. The following system functions are available to do that:

detect_source(), this is a generalised function that returns one of these values: 'TRIGGERED', 'SCHEDULED','UPLOAD'. This function is not generally used unless the fact that the profile that is running is on a schedule is important.

The following functions are generally used:

detect_upload(), if a file has been uploaded to this profile then this function returns a structure with the path of the file and the originating file name. The return is false if there has been no upload to process.

detect_trigger(), if the process has been triggered by another process (or by a user on the ALCW web site manually triggering the profile) the function returns a structure with details of the triggering profile, along with file details if the triggering profile triggered with a file. The return is false if there has not been a trigger.

detect_lookup(lookup_global), usually used after the profile has been initiated by schedule (although perfectly valid in other circumstances). The function uses the lookup_global to detect a file in a location defined by the lookup. The function returns false if the lookup is invalid or no file is found.

There is an important note here:
If a profile can be initiated by a triggering profile or by an upload, and it is important to know if it is by a trigger then the call to detect_trigger must be called first before detect_upload, otherwise if the triggering profile included a file passed in the trigger then detect_upload will also return the details of that file (as if it was an upload).

So the sequence would look like this:

```
$uploaded = detect_trigger();
if $uploaded is false $uploaded = detect_upload();
if $uploaded is false exit false; // nothing to do

case $uploaded.TriggerByTitle begin
   when “my_trigger_master” begin
         // we have been triggered by a profile called
         // my_trigger_master
         // process the file it sent in $uploaded.FilePath
         // as appropriate
       end
  when “foobar_profile” begin
       // been triggered by foobar_profile, so act as
       // appropriate
       end
  // Not triggered by any profile this knows, or we are
  // running because of upload, the default case
  begin
       // process the file as appropriate in
       // $uploaded.FilePath
  end
end
```

Initiating a trigger to a profile.

The function trigger_process initiates a trigger process. It can be called thus (say, by a profile ‘my_trigger_master’):

```
trigger_process(‘my_slave_profile’);
```

The profile my_slave_profile will collect the following structure after a call to detect_trigger:

```
$uploaded.TriggerByRef = “PF897654321”;
$uploaded.TriggerByTitle = “my_trigger_master”;
```

Note there are no entries for FilePath or FileName, there would be if the master profile called trigger_process the following way:

```
trigger_process(“my_slave_profile”, $file_path, $filename);
```

Then receiving profile will have the structure:

```
$uploaded.TriggerByRef = “PF897654321”;
$uploaded.TriggerByTitle = “my_trigger_master”;
$uploaded.FilePath = “ some file path “;
$uploaded.FileName = “ originating file name “;
```

Note that the structure is similar to that received by detect_upload, which is why the detect_trigger should be called first if it is important to detect the difference between an upload and a trigger.