AFD Robot

.NET API Guide

June 2014

Table of Contents

1. Introduction

AFD Robot API makes using address management and bank validation in .NET easy, minimising your work to get an integration up and running quickly, while also providing full flexibility to customise it to your requirements. Regardless of if you are using it with a Windows Form, ASP .NET or a backend application our .NET class library makes integration with your application simple.

If you wish to integrate directly into a WinForms application you may also like to consider looking at the .NET API WinForms Guide.

2. Setup

To use AFD Robot API you first need to install an AFD product or have access to an Evolution server. You can download data-restricted evaluation products from our website at:

http://www.afd.co.uk/evaluate.asp

To use AFD Robot API in .NET simply add a Reference to your project to our class library (AFD.API.dll). To do this right click the References item under your project in Solution Explorer select "Add Reference", and use the Browse Tab to browse to the library file.

3. Integrating into any .NET Application

Robot API can be easily used in any .NET application including ASP .NET applications, WPF/Silverlight, backends etc. and you can determine if and how to display results on any interface you provide.

3.1. Creating an Instance of the Robot Class

Add the following line to the top of your form's .cs file to reference the AFD.API:

```
    using AFD.API;
```

To create an instance of the Robot class add the following line to your code:

```
Robot robot = new Robot(RobotType.Address);
```

(You can also use RobotType.Bank or RobotType.Email if wanting to carry out Bank or Email Validation). If you wish to mix Address, Bank and/or Email validation simply create multiple instances of the Robot class.

If using an evolution server rather than a locally installed product set the Authentication property to an instance of the Authentication class for your server, e.g.:

```
robot.Authentication = new Authentication("http://myserver:81", "333333", "password");
```

3.2. Task Reference for Robot Class

3.2.1. Address Lookup (Postcode and Fast-Find)

To find an address record from the postcode or any fragment of the address use the Find method on an instance of the Robot class with the lookup string you wish to use and one of the following operations:

Operation.FastFindLookup – Looks up the supplied string which can be a postcode or any fragment of the address.

Operation.PostcodeLookup – Looks up a supplied postcode

Operation.PostcodePropertyLookup – Looks up a supplied postcode which optionally may also include a property, e.g. “279, B11 1AA”

This will return:
> 0 The number of results found
< 0 No results have been found (-2) or an error has occurred (use LastErrorText to obtain a human readable message for the error).

You can then iterate through the robot.Records collection to process each result in your application. This collection has the method getField(string) which you can use to return any of the fields for the result.

For example:

```
// Create instance of the Robot class
Robot robot = new Robot(RobotType.Address);

// Lookup the records
int results = robot.Find("b11 1aa", Operation.FastFindLookup);

// Check for error
if (results < 0) {
    MessageBox.Show(robot.LastErrorText);
    return;
}

// Iterate through results
foreach (AFD.API.Record record in robot.Records) {
    // Obtain the fields you require - some examples:
    string list = record.GetField("List");
    string organisation = record.GetField("Organisation");
    string property = record.GetField("Property");
    string street = record.GetField("Street");
    string locality = record.GetField("Locality");
    string town = record.GetField("Town");
    string postcode = record.GetField("Postcode");
    // display a string for the result
    // - replace with a function to process the result in your application?
    MessageBox.Show(list);
}
```

The number of results returned for FastFind operations will be limited by the MaxRecords Property of the Robot class instance. When using installed data it may also be limited by the Timeout Property. For evolution the server's pre-configured MaxSearchTime may be a limiting factor.

3.2.2. Address Search

To carry out a search for an Address:

- Call clear to Clear any current address fields.
- Use the setField method to set the value of each field you wish to search
- Call Find with Operation.Search (the Lookup parameter will be ignored)

The Find function will return:

> 0 The number of results found

< 0 No results have been found (-2) or an error has occurred (use LastErrorText to obtain a human readable message for the error).

You can then iterate through the robot.Records collection to process each result in your application. This collection has the method getField(string) which you can use to return any of the fields for the result.

For example:

```
// Create instance of the Robot class
Robot robot = new Robot(RobotType.Address);

// Set the fields to Search
robot.Clear();
robot.SetField("street", "Commercial Street");
robot.SetField("town", "Birmingham");

// Search for the records
int results = robot.Find(null, Operation.Search);

// Check for error
if (results < 0) {
    MessageBox.Show(robot.LastErrorText);
    return;
}

// Iterate through results
foreach (AFD.API.Record record in robot.Records) {
    // Obtain the fields you require - some examples:
    string list = record.GetField("List");
    string organisation = record.GetField("Organisation");
    string property = record.GetField("Property");
    string street = record.GetField("Street");
    string locality = record.GetField("Locality");
    string town = record.GetField("Town");
    string postcode = record.GetField("Postcode");
    // display a string for the result
    // - replace with a function to process the result in your application?
    MessageBox.Show(list);
}
```

The number of results returned for Search operations will be limited by the MaxRecords Property of the Robot class instance. When using installed data it may also be limited by the Timeout Property. For evolution the server's pre-configured MaxSearchTime may be a limiting factor.

3.2.3. Bank Lookup and Search

Bank records can be looked up or searched for in exactly the same way as Address records. The difference being that the Robot class must be instantiated with a parameter of RobotType.Bank rather than RobotType.Address.

The applicable operations are:

Operation.FastFindLookup – Looks up the supplied string which can be a sortcode, bank/branch name, location, STD Code or BIC.

Operation.Search – Search for results matching the search fields specified using the SetField method.

An example of a Bank Lookup:

```
// Create instance of the Robot class
Robot robot = new Robot(RobotType.Bank);

// Lookup the records
int results = robot.Find("560036", Operation.FastFind);

// Check for error
if (results < 0) {
    MessageBox.Show(robot.LastErrorText);
    return;
}

// Iterate through results
foreach (AFD.API.Record record in robot.Records) {
    // Obtain the fields you require - some examples:
    string list = record.GetField("List");
    string sortcode = record.GetField("SortCode");
    string ownerBankFullName = record.GetField("OwnerBankFullName");
    string branchTitle = record.GetField("FullBranchTitle");
    string location = record.GetField("Location");
    string directDebits = record.GetField("BACSDirectDebits");
    // display a string for the result
    // - replace with a function to process the result in your application?
    MessageBox.Show(list);
}
```

3.2.4. Bank Account Validation

To validate an account number use the setField method to set the value of the sort code and account number (alternatively the IBAN can be used)

You may also wish to set the clearing system if you want to restrict validation to the UK (BACS), or Irish (IPSO) systems only.

Call Find with Operation.ValidateAccount (the Lookup parameter will be ignored)

Following successful validation a single record will be returned. It is recommended you use the sortcode and account number returned rather than that you supplied as non-standard length account numbers will be transcribed for you.

An example of a Bank Account Validation:

```
// Create instance of the Robot class
Robot robot = new Robot(RobotType.Bank);

// Set the Account Details to validate
robot.Clear();
robot.SetField("SortCode", "774814");
robot.SetField("AccountNumber", "24782346");
// RollNumber can also be set if applicable, IBAN can also be validated instead

// Validate the Account Details
int results = robot.Find(null, Operation.AccountValidate);

// Check for error, e.g. invalid account number
if (results < 0) {
    MessageBox.Show(robot.LastErrorText);
    return;
}

// Success will return a single result
string sortcode = robot.GetField("SortCode");
string accountNumber = robot.GetField("AccountNumber");
string iban = robot.GetField("IBAN");
string rollNumber = robot.GetField("RollNumber");
string typeOfAccount = robot.GetField("TypeOfAccount");
string clearingSystem = robot.GetField("ClearingSystem");
MessageBox.Show(“Account Number is Valid");
```

3.2.5. Card Number Validation

To validate a debit or credit card number use the setField method to set the value of the card number and optionally expiry date. (The expiry date is checked that it is in range for the current date if provided).

Call Find with Operation.ValidateCard (the Lookup parameter will be ignored).

Following successful validation a single record will be returned.

An example of Card Number Validation:

```
// Create instance of the Robot class
Robot robot = new Robot(RobotType.Bank);

// Set the field to Validate
robot.Clear();
robot.SetField("cardnumber", "4694782385016585");
robot.SetField("expirydate", "10/17");  // This field is optional

// Validate the Card Details
int results = robot.Find(null, Operation.CardValidate);

// Check for error, e.g. invalid account number
if (results < 0) {
    MessageBox.Show(robot.LastErrorText);
    return;
}

// Success will return a single result
string cardType = robot.GetField("CardType");
MessageBox.Show(“Valid: ‟ + cardType);
```

3.2.6. Email Validation

To validate an email address from an instance of the robot class instianted with RobotType.Email simply call the Find method with the email address to validate and Operation.EmailValidate.

An example of Email Address Validation:

```
// Create instance of the Robot class
Robot robot = new Robot(RobotType.Email);

// Validate the Card Details
int results = robot.Find("support@afd.co.uk", Operation.EmailValidate);

// Check for error, e.g. invalid Email Address
if (results < 0) {
    MessageBox.Show(robot.LastErrorText);
    return;
}

// Email Address is Valid
MessageBox.Show("Email Address is Valid");
```

3.2.7. Statelessly Retrieving Previous Records

If you have returned results following an address or bank lookup or search and require to fetch one of those results again you can do this with a Retrieve operation. An example when you might wish to do this if is returning a list of results to the user, wanting to minimise data passed and then need to retrieve the full record when a user selects an individual record.

To do this you need to store the key (use getField("Key") to obtain this) when processing the original record. Please note that this Key is unique in a particular dataset so can be used statelessly or across multiple servers as long as the same version of the data is still in use. It is not unique across future versions of the data and therefore should not be stored in a database as a unique reference.

To retrieve a record call Find with the key for the record specified in the lookup field and an operation of Operation.Retrieve.

A single result will be returned which you can then process.

This will return:
1 The single result was found
< 0 No results have been found (-2) or an error has occurred (use LastErrorText to obtain a human readable message for the error).

For example:

```
// Create instance of the Robot class
Robot robot = new Robot(RobotType.Address);

// Lookup the records
int results = robot.Find("B11 1AA1001", Operation.Retrieve);

// Check for error
if (results < 0) {
    MessageBox.Show(robot.LastErrorText);
    return;
}

// Obtain the fields you require - some examples:
string organisation = robot.GetField("Organisation");
string property = robot.GetField("Property");
string street = robot.GetField("Street");
string locality = robot.GetField("Locality");
string town = robot.GetField("Town");
string postcode = robot.GetField("Postcode");
```

3.2.8. Calling off results individually

For ease of use the Find method used in the previous sections returns all results at once until a pre-determined timeout or maximum record count is reached (if applicable). When used with an evolution server there is little point to doing anything else as results have to result from the same web request.

However in some cases when using an installed product, you may wish to return results one-by-one as they are called off or provide your own mechanism for determining when to abort a search (for example based on the actual results that come back).

In such scenarios rather than calling Find, you can call the FindFirst and FindNext methods. FindFirst has identical parameters to Find but rather than returning the results in the Records collection, it instead returns one result at a time (FindNext takes no parameters).

You do also need to be aware that FindFirst and/or FindNext will sometimes return a value of 0 indicating a record or sector break which is provided to give the opportunity to cancel long searches and at which point no new result will be returned.

An example of an Address Fast-Find, identical that given in section 4.2.1 but using findFirst and FindNext is as follows:

```
// Create instance of the Robot class
Robot robot = new Robot(RobotType.Address);

// Lookup the records
int result = robot.FindFirst("b11 1aa", Operation.FastFindLookup);
```

```
// Check for Error, but note if 0 has been returned it is still possible no results
will be found
if (result < 0) {
    MessageBox.Show(robot.LastErrorText);
    return;
}

// Process and call off each result
long results = 0;
while (result >= 0) {
    if (result > 0) {
        // obtain some of the fields for the result to use in your application
        string list = robot.GetField("List");
        string organisation = robot.GetField("Organisation");
        string property = robot.GetField("Property");
        string street = robot.GetField("Street");
        string locality = robot.GetField("Locality");
        string town = robot.GetField("Town");
        string postcode = robot.GetField("Postcode");
        // display a string for the result
        // - replace with a function to process the result in your application?
        MessageBox.Show(list);
        results++;
    }
    else if (result == 0) {
      // sector or record break - take the chance to allow user to cancel?
    }
    // set your own condition(s) to abort the search
    if (results > 500) break;
    result = robot.FindNext();
}
if (results > 0) {
    // success
}
else {
    MessageBox.Show("No Results Found");
}
```

3.3. Function Reference for Robot Class

The following functions are available in the Robot Class:

3.3.1. Clear

void Clear()

This clears the fields, necessary prior to setting criteria for a search (not necessary for fast-finds and postcode lookups).

3.3.2. GetField

string GetField(string)

This property returns the value of the specified field name for the current record. When using Find to return multiple results at once you should use the GetField method of a Record in the Records collection property instead to retrieve fields from any record returned.

3.3.3. Find

int Find(string Lookup, Operation, optional SkipOptions)

Carries out an operation using the AFD API. For a lookup, card or email validation the Lookup parameter specifies the lookup string (for example a postcode, sortcode, card number or email address). The Operation specifies the operation to carry out.

The following Operations are supported:

Robot Type	**Operation**	**Description**
Address	FastFindLookup	Looks up an address fragement specified by the Lookup parameter. This could be a postcode or fields from an address.
	PostcodePropertyLookup	Looks up an address from the specified postcode and optionally property (e.g. “277, B11 1AA”) specified by the Lookup parameter.
	PostcodeLookup	Looks up an address from the specified postcode.
	Search	Searches for an address specified by criteria set by calling setField for the appropriate fields prior to calling this function (lookup parameter is ignored and can be null)
	Retrieve	Retrieves a previously returned record. The required records key field should be specified as the lookup parameter.
Bank	FastFindLookup	Looks up a sort code or searches for a bank from the fragment given as specified by the Lookup parameter.
	Search	Searches for a bank branch specified by critieria set by calling setField for the appropriate fields prior to calling this function (lookup parameter is ignored and can be null).
	Retrieve	Retrieves a previously returned record. The required records key field should be specified as the lookup

		parameter.
	AccountValidate	Validates an account number. Use setField to set both the sortcode and account number or alternatively the IBAN prior to calling this function. The lookup parameter is ignored.
	CardValidate	Validates a card number. The lookup parameter should be set to the card number to validate.
	EmailValidate	Validates an email address. The lookup parameter should be set to the email address to validate.

For address functions you can optionally set the SkipOptions to a member of the SkipOptions enumeration to skip records in large searches. These options are as follows:

Skip Option	**Description**
None	Default – Returns all matching records (unless a timeout or maximum number of records is reached).
Address	Only returns the first matching record per address (or household). Only has any effect with Names & Numbers.
Postcode	Only returns the first matching record for each postcode.
Sector	Returns the first matching record per Sector (this is the portion of the postcode prior to the space and the first digit afterwards, e.g. “B11 1”).
Outcode	Returns the first matching record per Outcode (this is the portion of the postcode prior to the space, e.g. “B11”);
Town	Returns the first matching record per Royal Mail Post Town.
Area	Returns the first matching record per Postcode Area (this is the letters at the start of the postcode, e.g. “B” or “AB”).

This function returns the number of matching records returned, or a value < 0 if an error has occurred. Use the LastErrorText property to obtain a human readable version of an error.

3.3.4. FindFirst

int FindFirst(string Lookup, AFD.API.Operation Operation)
This method is identical in parameters to Find. The difference being that it will obtain the first result only (or sometimes a record or sector break in a long search). You can then call FindNext() to obtain subsequent records.

In general it is recommended you use Find to obtain results as this simplifies calling off results and in-built support for the MaxRecords and, for installed data, Timeout properties to restrict long searches to help prevent your application becoming unresponsive. However in cases were you are using locally installed data and wish to have full control over this process this method gives you that flexibility.

Some examples were you might use this would be if you wish to return results to the user as they are called off rather than at the end of a long search, or you wish to be able to determine when to abort a search based on the results themselves or user intervention then this method gives you that flexibility.

The return codes are the same as for Find with the addition of 0 which indicates a record or sector break, when returned no result has been returned (You will need to call FindNext to continue the search). This allows the opportunity to cancel long searches after a predetermined timeout etc. if required.

3.3.5. FindNext

int FindNext()

This method is used following a call to FindFirst to repeatedly call off subsequent records. It will return 1 on success, 0 on a record or sector break (indicating no new result has been returned but giving the opportunity to cancel or provide user responsiveness) or -6 to indicate the end of a search.

3.3.6. InsertRecord

int InsertRecord(int)

This function is not relevant when not using with an instance of Windows Forms and will always return false in such cases.

3.3.7. LastErrorText

string LastErrorText()

This returns human readable text for the last error to occur (will return an empty string if the last operation was successful). While you may wish to display your own error text for some errors it is recommended that you fall-back to using this for any error code you have not handled yourself.

3.3.8. SetField

bool SetField (string FieldName, string FieldValue)

Sets the value of the specified field for searching.

3.4. Properties Reference for Robot Class

DialogOptions and Mappings are not documented in this section as they are ignored unless Robot is instantiated for use specifically with a Windows Form (see Section 3 for futhur details).

3.4.1. AddressFields

Set the number of address fields to use when returning free address fields with GetField (e.g. address1, address2, etc.). The default is 6, setting this to the same number as the fields you have causes Robot to squeeze fields in as required.

3.4.2. IncludeOrganisation

Boolean value indicating if the Organisation is included in free address field (i.e. if you obtain address1, address2, etc.)

3.4.3. LastErrorText

Returns human readable descriptive text for the last operation. Returns an empty string if no error occurred (result code >= 0).

3.4.4. LastResult

Returns the result code for the last operation. While be < 0 if an error occurred or > 0 if successful.

3.4.5. MaxRecords

Sets the Maximum number of records to return from any lookup or search (default 500).

3.4.6. Records

Following an option calling Find this is a collection of the results (Record objects). Each Record in the collection has a getField method used to obtain individual fields for that result.

3.4.7. Timeout

Sets the maximum time in seconds to spend searching for records (Default is 30 seconds). This helps prevent very long searches making an application unresponsive or a very vague search taking too long when it could instead be refined. This is applicable to installed products only. For evolution the server will be pre-configured with a MaxSearchTime which has the same effect.

3.4.8. UpperCaseTown

Indicates that the Town will always be returned in upper case (default and preferred by Royal Mail for address labels)

3.5. Record Class

The Record class cannot be instantiated on its own. A collection of Record objects is returned by the Records Property of the Robot class following an operation calling Find`.

The class has a single method GetField. This property returns the value of the specified field name for the record.

3.6. Collections

3.6.1. RobotType

This collection is used on instantiating the Robot class to specify the type of Robot used. The values are Address for Address Management (e.g. Address Fastfind, Search, etc.), Bank for BankFinder (e.g. Account or Card Validation) or Email for Email Validation.

You cannot mix types in one instance of the robot class, but you can have multiple instances of the class of different types called in the same function in your code

3.6.2. Operation

This collection is used to specify the operation carried out. Section 4.3.3 provides a table with the valid Operations for each RobotType.

3.7. Return Codes

The possible return codes, available as the return code from FindFirst or the LastResult property, are as follows:

3.7.1. Success Codes

These are the possible codes returned:

Value	Description
0	The search/lookup has not completed but may take some time and so is returning to give the user the option to cancel a long search.
1	The function was successful and a matching record has been returned.
2	This applies only to Bankfinder account number validation and indicates that the function was successful and the account number should be taken as valid. However, as account numbers on this sortcode cannot be validated you may wish to double check it is correct.

3.7.2. Error Codes

These specify the possible errors returned from any API function:

Value	Description
-1	The field specification string specified is invalid. This shouldn’t be returned under normal circumstances.
-2	No records matching your lookup or search criteria were found.
-3	The record number provided (e.g. when re-retrieving an item from a list box) is invalid.
-4	An error occurred attempting to open the AFD data files. Check they are correctly installed.

-5	An error occurred reading the data. Likely to be due to corrupt data so software may need to be re-installed.
-6	End of Search (when the last result has already been called off – indicates there are no more results to return).
-7	Indicates there is an error with the product registration. Normally due to it having expired. Run the Welcome program to re-register the software.
-8	Occurs if you attempt to search for a Name and Organisation at the same time. Also occurs with Postcode Plus if the UDPRN field is searched for at the same time as any other field.
-99	Indicates that the user clicked the cancel button if the DLL internal list box was used.
-12	The sort code specified for an account number validation does not exist.
-13	The sortcode specified for an account number validation is invalid.
-14	The account number specified for an account number validation is invalid.
-21	The sort code and account number given are for a building society account which also requires a roll number for account credits. No roll number has been supplied or is incorrect for this building society.
-22	The International Bank Account Number provided is in an invalid format
-23	The IBAN provided contains a country that is not recognised as valid
-24	Both an IBAN and Account Number was provided and these details do not match.
-15	The expiry date specified for a card validation is invalid.
-16	The card has expired
-18	The card number specified for a card validation is invalid.
-19	The card number specified is a Visa card which can be used in an ATM only.
-20	While the card number appears to be a valid one, the card is not of any of the known types and is therefore unlikely to be acceptable for payment.

4. Evolution Server Details

The AFD.API.Authentication Object is used to specify authentication details to use robot with an evolution server. If you do not assign one to the Authentication property of the Robot class then a default instance is created which uses local installed product.

We strongly recommend that the details passed to create an instance of the Authentication object are stored as modifiable configuration parameters in your application, so that details can be changed in the future without needing to alter code.

There are three initialisers for the Authentication object

Authentication()
This creates a version to work with installed data only. You will require a copy of an AFD product installed on each machine your software runs on. This is the fastest and most efficient method and is particularly ideal for offline use.

Authentication(Serial, Password)
This creates a version to work with AFD's server. You pass it your serial number and password.

Authentication(Server, Serial, Password)
This creates a version to work with your own server. The Server parameter should be a string including the protocol, server and (if not 80) the port so as to give wide compatibility. For example http://server:81

5. ASP .NET Webpage

With ASP .NET webpages you can easily use Robot with the Robot class as described in Section 4 of this manual.

If you wish to use auto-configuration with your webpage you may wish to consider using Robot Web rather than Robot API as Robot Web uses JavaScript and AJAX to lookup addresses in-line from your webpage.

If using Robot API in an ASP .NET webpage an example of obtaining address results to add to a list box is as follows:

```
protected void ButtonLookup_Click(object sender, EventArgs e)
{
   // Create an instance of the Robot class for Address operations
   AFD.API.Robot robot = new AFD.API.Robot(AFD.API.RobotType.Address);

   // Lookup the postcode or fast-find string in the TextLookup TextBox
   int results = robot.Find(TextLookup.Text,
           AFD.API.Operation.FastFindLookup);

   // Add Items to ListBoxAddress For The User To Select From
   ListBoxAddress.AutoPostBack = true;
   foreach (AFD.API.Record record in robot.Records) {
        string ListText = record.GetField("list");
        string ListValue = record.GetField("key");
        ListItem item = new ListItem(ListText, ListValue);
        ListBoxAddress.Items.Add(item);
   }
}
```

On selecting an item from a List Box the record can be re-looked up to provide full address details for submission, an example of this:

```
        protected void ListBoxAddress_SelectedIndexChanged(object sender, EventArgs e)
        {
           // Create an instance of the Robot class for Address operations
           AFD.API.Robot robot = new AFD.API.Robot(AFD.API.RobotType.Address);

           // Lookup the selected item
           int results = robot.Find(ListBox1.SelectedValue,
AFD.API.Operation.Retrieve);

           // Assign the address to fields on the Form
           TextBoxOrganisation.Text = robot.GetField("organisation");
```

```
    TextBoxProperty.Text = robot.GetField("property");
    TextBoxStreet.Text = robot.GetField("street");
    TextBoxLocality.Text = robot.GetField("locality");
    TextBoxTown.Text = robot.GetField("town");
    TextBoxPostcode.Text = robot.GetField("postcode");
}
```

6. Appendices

Appendix A. Address Management Product Fields

This currently includes Postcode, Plotter, Postcode Plus & Names & Numbers.

■ Field returned by this product and fully searchable
□ Field returned by this product, but not searchable.

Note: that the alternative address formats provided do share some of the same fields where there data is identical, but you should not mix and match other fields between the different formats as this could lead to address corruption. For example with Standard Address Fields the Street or Locality field could include a street number, whereas with Raw PAF Fields the number would be in the separate Number field.

Field Name	**Default Size**	**Description**	**Postcode**	**Plotter**	**Postcode Plus**	**Names & Numbers**
Lookup	255	Specify postcode (or zipcode) and fast-find lookup string's here for lookup operations.	■	■	■	■
Key	255	Provides a key which can be used to easily retrieve the record again, e.g. when a user clicks on an item in the list box.	■	■	■	■
List	512	Provides a list item formatted to be added to a list box for this record.	□	□	□	□
Product	40	Indicates the product name used [10]	□	□	□	□
Occupant Fields						
Name	120	Full name (includes title, first name, middle initial and surname).				□
Gender	6	The gender (M or F) of the resident if known.				■
Forename	30	The first name of the resident				■
MiddleInitial	6	The middle initiate of the resident.				■
Surname	30	The surname/last name of the resident.				■
OnEditedRoll	6	Indicates if the resident is on the edited electoral roll (i.e. they have not opted out). Set to Y if they are on the Edited Roll, N if not, blank for Organisation and other records). To search set to #Y to return only records on the electoral roll, #N only for those not on the electoral roll or !N for all				■

		records including Organisations but excluding those not on the Edited Roll.				
DateOfBirth	10	The residents date of birth if known (electoral roll attainers in the last 10 years only).				■
Residency	6	Gives time in years that the occupant has lived at this address.				■
HouseholdComposition	106	Describes the household composition of the selected address [6]				■
Organisation	120	Full business name (includes any department)			■	■
Property	120	Property (building-includes any sub-building).			■	■
Street	120	Delivery Street (includes any sub-street)	■	■	■	■
Locality	70	Locality (sometimes a village name – in ZipAddress used for Urbanization)	■	■	■	■
Town	30	Postal Delivery Town (or City)	■	■	■	■
Postcode	10	The Royal Mail Postcode for this address (or ZipCode)	■	■	■	■
OrganisationName	60	Business Name			■	■
Department	60	Department Name			■	■
Sub Building	60	Sub Building Name			■	■
Building	60	Building Name			■	■
Number	10	House Number			■	■
DependentThoroughfare	60	Sub-Street Name	■	■	■	■
Thoroughfare	60	Street Name	■	■	■	■
DoubleDependentLocality	35	Sub-Locality Name	■	■	■	■
DependentLocality	35	Locality Name (Urbanization in ZipAddress)	■	■	■	■
Town	30	Postal Delivery Town (City)	■	■	■	■
Postcode	10	The Royal Mail Postcode for this address (or Zipcode)	■	■	■	■
Identifier	8	Provides a unique identifier for the address (the Royal Mail UDPRN)			■	■
BuildDate	10	Provides the build date, which can be used as the start date, entry date, and update date fields for BS7666.			□	□
Administrator	20	Provides the administrator of the gazetteer (AFD).			□	□
Language	5	Provides the language (ENG)			□	□
Department	60	The name of a department within an organization where required.			■	■
Organization	60	The Organization Name			■	■
SubUnit	60	Sub-Unit of a building where needed			■	■
BuildingName	60	Building Name where present			■	■
BuildingNumber	10	Building Number, including 17A, 17-19, etc.			■	■
SubStreet	60	Sub-Street where needed			■	■
DeliveryStreet	60	Designated Street Name	■	■	■	■
SubLocality	60	Sub-Locality where required	■	■	■	■
DeliveryLocality	60	Locality name (or Urbanization)	■	■	■	■
DeliveryTown	30	Postal Town name (or City)	■	■	■	■

Code	10	The Postcode (or ZipCode)	■	■	■	■
Postal County	30	Royal Mail supplied postal county	■	■	■	■
AbbreviatedPostalCounty	30	Royal Mail approved abbreviation is used where available for the postal county	■	■	■	■
OptionalCounty	30	Postal counties including optional ones for most addresses which would otherwise not have a county name.	■	■	■	■
AbbreviatedOptionalCounty	30	Royal Mail approved abbreviation is used where available for the optional county	■	■	■	■
TraditionalCounty	30	The traditional county name for this postcode	■	■	□	■
AdministrativeCounty	30	The administrative county name for this postcode	■	■	□	■
Outcode	4	The Outcode portion of the Postcode (the portion before the space)	■	■	■	■
Incode	3	The Incode portion of the Postcode (the portion after the space).	■	■	■	■
DPS	2	The Delivery Point Suffix which along with the postcode uniquely identifies the letterbox.			□	■
PostcodeFrom	8	Used with Postcode field to provide a range for searching. Also returns any changed postcode from a lookup.	□	□	■	■
PostcodeType	6	L for Large User Postcode, S for Small User.	□	□	□	■
MailsortCode	5	Used for obtaining bulk mail discounts.	□	□	□	■
UDPRN	8	Royal Mail Unique Delivery Point Reference Number assigned to this letter box.			■	■
JustBuilt	10	AFDJustBuilt - Contains the date of inclusion on PAF for properties thought to be recently built. The date is stored numerically in descending format in the form YYYYMMDD. YYYY is the year, MM is the month and DD is the day. For example 20080304 is 04/03/2008.			■	■
Phone Number Related Fields						
Phone	20	STD Code or Phone Number	■[3]	■[3]	■[3]	■
Geographical Fields						
GridE	10	Grid Easting as a 6 digit reference		□	■	■
GridN	10	Grid Northing as a 6/7 digit reference		□	■	■
Latitude	10	Latitude representation of Grid Reference in Decimal Format (WGS84)		□	■	■
GBGridE	10	UK Based Grid Easting as a 6 digit reference. Always returns the UK based grid even for Northern Ireland addresses.		□	■	■
GBGridN	10	UK Based Grid Northing as a 6/7 digit reference.		□	■	■
NIGridE	10	Irish Grid Based Grid Easting as a 6 digit reference. Always returns the Irish base grid even for mainland UK addresses.		□	■	■

NIGridN	10	Irish Grid Based Grid Northing as a 6/7 digit reference.		□	■	■
Longitude	10	Longitude representation of Grid Reference in Decimal Format (WGS84)		□	■	■
Miles	6	Distance from supplied grid reference			■	■
Km	6	Distance from supplied grid reference			■	■
UrbanRuralCode	2	Provides a code which indicates if an area is mainly urban or rural and how sparsely populated those areas are. [11]			□	■
UrbanRuralName	60	Provides a description which goes along with the UrbanRuralCode.			□	■
SOALower	9	Lower level Super Output Area (Data Zone in Scotland, Super Output Area in Northern Ireland)			□	■
SOAMiddle	9	Middle level Super Output Area (Intermediate Geography in Scotland, not applicable for Northern Ireland).			□	■
SubCountryName	20	Provides the devolved or non-UK country name (e.g. England, Scotland, Wales etc.)			□	■
WardCode	9	Code identifying the electoral ward for this postcode			□	■
WardName	50	Name identifying the electoral ward for this postcode			□	■
AuthorityCode	9	Local/Unitary Authority for this Postcode (same as the start of the ward code).			□	■
Authority	50	Local / Unitary Authority for this postcode			□	■
ConstituencyCode	9	Parliamentary Constituency Code for this postcode			□	■
Constituency	50	Parliamentary Constituency for this postcode			□	■
DevolvedConstituencyCode	9	Devolved Constituency Code for this postcode (currently covers Scotland)			□	■
DevolvedConstituencyName	50	Devolved Constituency Name for this postcode (currently covers Scotland)			□	■
EERCode	9	Code identifying the European Electoral Region for this postcode			□	■
EERName	40	Name identifying the European Electoral Region for this postcode			□	■
LEACode	3	Code identifying the Local Education Authority for this postcode			□	■
LEAName	50	Name identifying the Local Education Authority for this postcode			□	■
TVRegion	30	ISBA TV Region (not TV Company)			□	■
Postcode Level Property Indicator Fields						
Occupancy	6	Indication of the type of occupants of properties found on the selected postcode [4]	□	□	□	■
OccupancyDescription	30	Description matching the Occupancy [4]	□	□	□	□
AddressType	6	Indication of the type of property level data to	□	□	□	■

		capture to have the full address for a property on the selected postcode. [5]				
AddressTypeDescription	55	Description matching the Address Type [5]	□	□	□	□
NHSCode	6	National Health Service Area Code			□	■
NHSName	50	National Health Service Area Name			□	■
PCTCode	9	National Health Service Clinical Commissioning Group Code for England (Local Health Board Code in Wales, Community Health Partnership in Scotland, Local Commissioning Group in Northern Ireland, Primary Healthcare Directorate in the Isle of Man)			□	■
PCTName	50	Name matching the PCT Code field			□	■
CensationCode	10	Censation Code assigned to this Postcode	□	□	□	■
CensationLabel	50	Provides a handle for the Censation Code	□	□	□	■
Affluence	30	Affluence description	□	□	□	■
Lifestage	100	LifeStage description	□	□	□	■
AdditionalCensusInfo	200	Additional information from the Census.	□	□	□	■
Business	100	Provides a description of the type of business				■
SICCode	10	Standard Industry Classification Code for an organisation record.				■
Size	6	Gives an indication of the number of employees of an organisation at this particular office. [7]				■
LocationType	6	The type of Business Location, e.g. Head Office or Branch Office				■
BranchCount	6	The number of branches for this business				■
GroupID	6	An ID of the Group were a business is part of a wider group				■
ModelledTurnover	15	The modelled annual turnover for the business				■
NationalSize	6	Gives an indication of the number of employees of an organisation covering all sites. [7]				■
AliasLocalities	4	Returns the number of alias records present for the postcode sector in which this result resides.			□	□
AliasLocality	35	Returns an alias (non-postal) locality that resides in the postcode sector that this address is contained in. Note that many postcode sectors have multiple alias localities and as such you can include this field multiple times to return multiple localities.			□	□
Advanced / Premium Fields						
DataSet	10	With Postcode Plus and Welsh data can be set to 'Welsh" to obtain the Welsh language version of an address in Wales where available. If not set then the English language version will be returned. With TraceMaster this indicates an historic				■

		dataset to use [9]				
CouncilTaxBand	6	Provides the Council Tax Band for the selected property. *Requires Names & Numbers*				■

Notes:

[3] STD Code Only – No Phone Number present

[4] Possible Occupancy values and descriptions are as follows (information in brackets not part of the description):

1. Large User Organisation (Single Organisation on this postcode)
2. Small User Organisation (All the properties on this postcode are likely to be businesses)
3. Mostly Organisations (Most of the properties on this postcode are organisations)
4. Mixed (This postcode contains a mixture of business and residential addresses)
5. Mostly Residential (Most of the properties on this postcode are residential)
6. Residential (All the properties on this postcode are likely to be residential)

[5] Possible Address Type values and descriptions are as follows (information in brackets not part of the description):

1. Numbered (Only a property number needs to be captured)
2. Numbered and Named (This postcode contains a mixture of properties needing a property number and those needing a property name including properties such as 16b)
3. Numbered and Named, Likelihood of Multiple Occupancy (This postcode contains a mixture of properties needing a property number and those needing a property name. Some of the properties on this postcode are likely to contain multiple occupants, e.g. flats).
4. Named (This postcode only contains properties needing a property name).
5. Non-Standard Address Format (This refers to addresses which do not have a street field at all, or have multiple street names on the same postcode. This also includes addresses with numbered localities (no street but a house number which goes in with the locality field). It is in-effect a warning to be careful in capturing the property information as it is not in one of the most common address formats).
6. PO Box (This postcode has a PO Box number)
7. No Property Information (Addresses on this postcode have no property information - i.e. capture an Organisation or Resident name only)

[6] The household composition field includes both a number and description and can have any of the following values.

1. 1 Male and 1 Female occupant with different surnames
2. 1 Male and 1 Female occupant with the same surname (married couples)
3. Mixed household
4. More than 2 persons with the same surname (e.g. older families).
5. 1 Male Occupant Only
6. 1 Female Occupant Only
7. More than 7 persons (e.g. old peoples home).

[7] The Size property can have any of the following values:

A. 1 to 9 employees
B. 10 to 19 employees
C. 20 to 49 employees
D. 50 to 99 employees
E. 100 to 199 employees

F. 200 to 499 employees
G. 500 to 999 employees
H. 1000+
(If blank then this is unknown or not applicable).

[8] The phone match type will be set to F if the phone number has been matched to the full name of this resident, or S if just to the surname. This can be useful for identifying the bill payer among multiple residents.

[9] DataSet property when used with the Names & Numbers TraceMaster product can currently be any of the following years: 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005 or Current (for the current data). Only one year can be specified at a time and searches/lookup's will fail if the specified year has not been installed. New years are automatically accessible when they become available if installed with no change required to the DLL or your application.

[10] The Product field can have any of the following values:
AFD Postcode
AFD Postcode Plotter
AFD Postcode Plus
AFD Names & Numbers
AFD Names & Numbers TraceMaster

[11] The Urban Rural Code differs from England and Wales, Scotland and Northern Ireland. The possible codes and there meanings are as follows:

England & Wales
1. Urban (Sparse): Falls within Urban settlements with a population of 10,000 or more and the wider surrounding area is sparsely populated
2. Town and Fringe (Sparse): Falls within the Small Town and Fringe areas category and the wider surrounding area is sparsely populated.
3. Village (Sparse): Falls within the Village category and the wider surrounding area is sparsely populated.
4. Hamlet and Isolated Dwelling (Sparse): Falls within the Hamlet and Isolated Dwelling category and thee wider surrounding area is sparsely populated.
5. Urban (Less Sparse): Falls within urban settlements with a population of 10,000 or more and the wider surrounding area is less sparsely populated.
6. Town and Fringe (Less Sparse): Falls within the Small Town and Fringe areas category and the wider surrounding area is less sparsely populated.
7. Village (Less Sparse): Falls within the village category and the wider surrounding area is less sparsely populated.
8. Hamlet and Isolated Dwelling (Less Sparse): Falls within the Hamlet & Isolated Dwelling category and the wider surrounding area is less sparsely populated

Scotland
S1. Large Urban Area: Settlement of over 125,000 people.
S2. Other Urban Area: Settlement of 10,000 to 125,000 people.
S3. Accessible Small Town: Settlement of 3,000 to 10,000 people, within 30 minutes drive of a settlement of 10,000 or more.
S4. Remote Small Town: Settlement of 3,000 to 10,000 people, with a drive time of 30 to 60 minutes to a settlement of 10,000 or more.

S5. Very Remote Small Town: Settlement of 3,000 to 10,000 people, with a drive time of over 60 minutes to a settlement of 10,000 or more.
S6. Accessible Rural: Settlement of less than 3,000 people, within 30 minutes drive of a settlement of 10,000 or more.
S7. Remote Rural: Settlement of less than 3,000 people, with a drive time of 30 to 60 minutes to a settlement of 10,000 or more.
S8. Very Remote Rural: Settlement of less than 3,000 people, with a drive time of over 60 minutes to a settlement of 10,000 or more.

Northern Ireland
A - E (Urban):
A. Belfast Metropolitan Urban Area
B. Derry Urban Area
C. Large Town: 18,000 and under 75,000 people
D. Medium Town: 10,000 and under 18,000 people
E. Small Town: 4,500 and under 10,000 people
F - H (Rural):
F. Intermediate Settlement: 2,250 and under 4,500 people
G. Village: 1,000 and under 2,250 people
H. Small Village, Hamlet or Open Countryside: Less than 1,000 people

[12] The record type will be one of the following:

General Delivery
Highrise
Firm
Street
PO Box
Rural Route/Highway Contract
Multi-Carrier Route

Appendix B. BankFinder Fields

■ Field returned by this product and fully searchable

□ Field returned by this product, but searchable only through the Search Text field only.

Note: that the alternative address formats provided do share some of the same fields where there data is identical, but you should not mix and match other fields between the different formats as this could lead to address corruption. For example with Standard Address Fields the Street or Locality field could include a street number, whereas with Raw PAF Fields the number would be in the separate Number field.

Field Name	**Default Size**	**Description**	**BankFinder**
General Fields			
Lookup	255	Specify sort code, postcode and fast-find lookup string's here for lookup operations.	■
ClearingSystem	25	Clearing system for this record [3]	□
Key	40	Provides a key which can be used to easily retrieve the record again, e.g. when a user clicks on an item in the list box.	■
List	512	Provides a list item formatted to be added to a list box for this record.	□
Product	40	Provides the product name used [10]	□
SearchText	255	Specify text to search for within any of the BankFinder fields	■
General Bank Fields			
SortCode	6	Bank's Sortcode	■
BankBIC	8	Bank BIC Code [1]	■
BranchBIC	3	Branch BIC Code [1]	■
SubBranchSuffix	2	Allows a branch to be uniquely identified where there is a cluster of branches sharing the same Sort Code [1]	□
ShortBranchTitle	27	The official title of the branch	■
FullBranchTitle	105	Extended title for the institution	■
CentralBankCountryCode	2	The ISO Country code for beneficiary banks in other countries	□
CentralBankCountryName	20	The country name corresponding to the ISO code given.	□
SupervisoryBodyCode	1	Indicates the supervisory body for an institution that is an agency in any of the clearings. [2]	□

SupervisoryBodyName	50	The name of the supervisory body [2]	□
DeletedDate	10	Specifies the date the branch was closed if it is not active	□
BranchType	20	The branch type - Main Branch, Sub or NAB Branch, Linked Branch	□
MainBranchSortCode	6	Set for linked branches in a cluster. It identifies the main branch for the cluster. For IPSO records this is set to the branch that would handle transactions for this sortcode when the branch has been amalgamated with another.	■
Location	60	Where present helps indicate the physical location of the branch.	■
BranchName	35	Defines the actual name or place of the branch	■
AlternativeBranchName	35	An alternative name or place for the branch where applicable.	■
OwnerBankShortName	20	Short version of the name of the Owning Bank	■
OwnerBankFullName	70	Full version of the name of the Owning Bank	■
OwnerBankCode	4	The four digit bank code of the Owning Bank [1]	□
Standard Address Fields *(Formatted as an address would appear on an envelope)*			
Organisation	120	Owner Bank Full Name	■
Property	65	Bank Postal Address: Property (Building)	□
Street	60	Bank Postal Address: Street	□
Locality	60	Bank Postal Address: Locality	□
Town	30	Bank Postal Address: Town	■
County	30	Bank Postal Address: County (Optional)	□
Postcode	8	The Royal Mail Postcode for this address	■
Raw PAF Fields *(Formatted closer to how they appear on Raw PAF, useful if your database stores fields this way)*			
OrganisationName	60	Owner Bank Full Name	■
SubBuilding	60	Bank Postal Address: Sub-Building Name	□
Building	60	Bank Postal Address: Building Name	□
Number	10	Bank Postal Address: House Number	□
DependentThoroughfare	60	Bank Postal Address: Sub-Street Name	□
Thoroughfare	60	Bank Postal Address: Street Name	□
Double DependentLocality	35	Bank Postal Address: Sub-Locality Name	□
DependentLocality	35	Bank Postal Address: Locality Name	□
Town	30	Bank Postal Address: Postal Delivery Town	■
County	30	Bank Postal Address: County (Optional)	□
Postcode	8	The Royal Mail Postcode for this address	■
Alternative Postcode Fields *(Can be used in-place of the Postcode field to provide it as separate parts)*			
Outcode	4	The portion of the postcode before the space	■
Incode	3	The portion of the postcode after the space	■
Phone Number Related Fields			

Phone	20	Phone Number for this bank	■
Fax	20	Fax Number for this bank (IPSO only)	□
BACS Related Fields *(Not applicable to IPSO Records)*			
BACSStatus	5	Indicates the BACS Clearing Status [4]	□
BACSStatusDescription	60	Provides a description for the status [4]	□
BACSLastChange	10	Date on which BACS data was last amended	□
BACSClosedClearing	10	Indicates the date the branch is closed in BACS clearing if applicable.	□
BACSRedirectedFromFlag	1	Set to R if other branches are redirected to this sort code.	□
BACSRedirectedToSortCode	6	This specifies the sort code to which BACS should redirect payments addressed to this sort code if applicable.	□
BACSSettlementBankCode	4	BACS Bank Code of the bank that will settle payments for this branch.	□
BACSSettlementBankShortName	20	Short form name of the settlement bank	□
BACSSettlementBankFullName	70	Full form name of the settlement bank	□
BACSSettlementBankSection	2	Numeric data required for BACS to perform it's settlement.	□
BACSSettlementBankSubSection	2	Numeric data required for BACS to perform it's settlement.	□
BACSHandlingBankCode	4	BACS Bank Code of the member that will take BACS output from this branch.	□
BACSHandlingBankShortName	20	Short form name of the handling bank	□
BACSHandlingBankFullName	70	Full form name of the handling bank	□
BACSHandlingBankStream	2	Numeric code defining the stream of output within the Handling Bank that will be used or payments to this branch.	□
BACSAccountNumbered	1	Set to 1 if numbered bank accounts are used	□
BACSDDIVoucher	1	Set to 1 if Paper Vouchers have to be printed for Direct Debit Instructions.	□
BACSDirectDebits	1	Set to 1 if branch accepts Direct Debits	□
BACSBankGiroCredits	1	Set to 1 if branch accepts Bank Giro Credits	□
BACSBuildingSocietyCredits	1	Set to 1 if branch accepts Building Society Credits.	□
BACSDividendInterestPayments	1	Set to 1 if branch accepts Dividend Interest Payments.	□
BACSDirectDebitInstructions	1	Set to 1 if branch accepts Direct Debit Instructions.	□
BACSUnpaidChequeClaims	1	Set to 1 if branch accepts Unpaid Cheque Claims.	□
CHAPS Related Fields *(Not applicable to IPSO Records)*			
CHAPSPStatus	1	Indicates the CHAPS Sterling clearing Status [5]	□
CHAPSPStatusDescription	80	Provides a description for the status [5]	□
CHAPSPLastChange	10	Date on which CHAPS Sterling data was last amended	□
CHAPSPClosedClearing	10	Indicates the date the branch is closed in CHAPS Sterling clearing if applicable.	□
CHAPSPSettlementBankCode	3	CHAPS ID of the bank that will settle payments for this	□

		branch,	
CHAPSPSettlementBankShortName	20	Short form of the name of the settlement bank	□
CHAPSPSettlementBankFullName	70	Full form of the name of the settlement bank	□
CHAPSEStatus	1	Indicates the CHAPS Euro clearing Status [6]	□
CHAPSEStatusDescription	80	Provides a description for the status [6]	□
CHAPSELastChange	10	Date on which CHAPS Euro data was last amended	□
CHAPSEClosedClearing	10	Indicates the date the branch is closed in CHAPS Euro clearing if applicable.	□
CHAPSEEuroRoutingBICBank	8	Specifies the SWIFT closed user group Bank BIC to which CHAPS Euro payments for this branch should be routed.	□
CHAPSEEuroRoutingBICBranch	3	Specifies the SWIFT closed user group Branch BIC to which CHAPS Euro payments for this branch should be routed.	□
CHAPSESettlementBankCode	3	CHAPS ID of the bank that will settle payments for this branch.	□
CHAPSESettlementBankShortName	20	Short form of the name of the settlement bank	□
CHAPSESettlementBankFullName	70	Full form of the name of the settlement bank	□
CHAPSEReturnIndicator	1	Set to R if this is the branch to which CHAPS Euro payments should be sent.	□
C&CCC Related Fields *(Not applicable to IPSO Records)*			
CCCCStatus	1	Indicates the C&CCC clearing Status [7]	□
CCCCStatusDescription	40	Provides a description for the status [7]	□
CCCCLastChange	6	Date on which C&CCC data was last amended	□
CCCCClosedClearing	30	Indicates the date the branch is closed in C&CCC clearing if applicable.	□
CCCCSettlementBankCode	3	BACS generated code of the bank that will settle payments for this branch.	□
CCCCSettlement BankShortName	20	Short form of the name of the settlement bank	□
CCCCSettlement BankFullName	70	Full form of the name of the settlement bank	□
CCCCDebitAgencySortCode	50	When the Status field is set to 'D' this specifies where cheque clearing is handled for this branch.	□
CCCCReturnIndicator	6	Set if this is the branch that other banks should return paper to. It will only be set for a sort code of a Member.	□
Validation Related Fields			
AccountNumber	12	The account number to validate (set along with the sort code field for account number validation).	■
TypeOfAccount	1	The type of account field required for transmitting data to BACS when the account number has been translated.	□
RollNumber	20	For some building society credit accounts a roll number is required. This can be specified here for validation.	■

IBAN	50	The International Bank Account Number. This contains the sort code and account number in a standardised format for cross-border transactions.	■
BuildingSocietyName	70	For building society accounts requiring a roll number this will contain the name of the receiving building society as this sometimes differs from the bank branch that the payment passes through.	□
CardNumber	20	Used to specify a card number to validate	■
ExpiryDate		Optional field to specify an expiry date to validate along with the card number.	■
CardType	30	Indicates the card type following validation [8]	□

Notes:

[1] Does not apply to records in the IPSO (Irish Payment Services Organisation) clearing system.

[2] The supervisory body code and name can be any of the following:
A. Bank of England
B. Building Society Commission
C. Jersey, Guernsey or Isle of Man authorities
D. Other

[3] The clearing system property can have one of the following values

United Kingdom (BACS) – For branch records for the UK clearing system
Ireland (IPSO) – For branch records on the Irish Payment Services Organisation Clearing System
Both UK and Irish – Returned by Account Number Validation only when a branch is on both systems.

Note, that you should only accept account numbers validated on the Irish system if you can clear through both the Irish (IPSO) system as well as the UK (BACS) system.

[4] Possible values for the BACS Status and Description fields are as follows:
M. Branch of a BACS Member
A. Branch of an Agency Bank
I. Member of the Irish Clearing Services (IPSO)
Does not accept BACS Payments

[5] Possible values for the CHAPS Sterling Status and Description fields are as follows:
M. Direct Branch of a CHAPS £ Member that Accepts CHAPS £ Payments
A. Branch of an Agency Bank that Accepts CHAPS £ Payments
I. Indirect Branch of a Member or Agency Bank that Accepts CHAPS £ Payments
Does not accept CHAPS £ Payments

[6] Possible values for the CHAPS Euro Status and Description fields are as follows:
D. Direct Branch of a CHAPS € Member that Accepts CHAPS € Payments
I. Indirect Branch of a Member or Agency Bank that Accepts CHAPS € Payments
Does not accept CHAPS € Payments

[7] Possible values for the C&CCC Status and Description fields are as follows:

M.	Branch of a C&CCC Member
F.	Full Agency Bank Branch
D.	Debit Agency Branch Only
	Not Part of the C&CCC Clearing

[8] Possible values for the card type field are as follows:
MasterCard
Visa
American Express
Visa Debit
Electron
Visa Purchasing
UK Maestro
International Maestro
Solo and Maestro
JCB
Charities Aid Foundation
MasterCard Debit

[10] The Product field would have the value ‘AFD BankFinder’.