

[print](#) | [close](#)

## APIs by Example: Print File Field Description

[System iNetwork Programming Tips Newsletter](#)

[Carsten Flensburg](#)

Carsten Flensburg

Thu, 09/09/2004 (All day)

In this installment of APIs by Example, Carsten Flensburg demonstrates the Retrieve Database File Description (QDBRTVFD) API. This API can be used to retrieve information about a database, which is similar to the information you can get with the DSPFD CL command.

The API is demonstrated in the form of a report program that reads information about the file and prints it. You could, of course, modify this program to use this information for your own purposes. For example, you might be writing an RPG program that needs to know what positions of a record a particular group of fields is in. This API can give you that information, or you may want to look up the column headings for fields in a file so that you can print them on your own report.

The output of the API can be characterized as a series of data structures, each one "pointed to" by another data structure. To take them all in, you have to use a technique that Carsten refers to as "structure jumping."

Structure jumping involves moving forward in memory by a variable amount using pointer logic.

The following will walk you through the process of reading the key fields for the file from the results of the QDBRTVFD API:

- a) Navigate to the manual page for the QDBRTVFD API in the Information Center by selecting Programming -> APIs -> APIs by Category -> Database and File -> Retrieve Database File Description.
- b) Scroll down to the section entitled "FILDO100 Format (File Definition Template (FDT) header)."
- c) A few lines below the heading for this section is a diagram that shows all of the data structures that are returned when using the FILDO100 format and how they are connected to each other.
- d) On the diagram, you can see that the "key specification array (Qdb\_Qdbfk)" that contains a list of all of the key fields in a file is pointed to by a field in the "file scope array (Qdb\_Qdbfb)." The file scope array is pointed to by the "file definition header (Qdb\_Qdbfh)."
- e) Call the API to get its returned value. The file definition header is the first part of the value returned by the API. To ensure that enough memory is available for the resulting values, start with a small amount of memory, such as 64k, with the %alloc() BIF. Then call the API. If the API could not fit everything into the memory that you provided, increase the space that you provide using the %realloc() BIF. Carsten does this in his program with the following code:

```
ApiRcvSiz  = 65535;  
pQdb_Qdbfh = %Alloc( ApiRcvSiz );  
Qdb_Qdbfh.Qdbfyavl = 0;
```

```

DoU  Qdb_Qdbfh.Qdbfyavl  ApiRcvSiz;
      ApiRcvSiz = Qdb_Qdbfh.Qdbfyavl;
      pQdb_Qdbfh  = %ReAlloc( pQdb_Qdbfh: ApiRcvSiz );
      EndIf;

      RtvDbfDsc( Qdb_Qdbfh
                  : ApiRcvSiz
                  : FilRtnQ
                  : 'FILD0100'
                  : PxDbfNamQ
                  : PxRcdFmt
                  : '0'
                  : '*LCL'
                  : '*EXT'
                  : ERRRC0100
                  );

      EndDo;

```

f) All of the data that the API returns has now been loaded into the memory that you allocated. However, you have to use structure jumping to read that memory correctly. After running the above code, only the Qdb\_Qdbfh data structure is loaded. The next thing you need is the file scope array. If you scroll down the manual page to the section entitled "File Scope Array (Qdb\_Qdbfd)," it says that you can locate this section with the offset called Qdbfos, which is found in the Qdb\_Qdbfh data structure.

g) You use pointer logic to find the file scope array. Since the Qdb\_Qdbfb data structure is based on the pQdb\_Qdbfb pointer in Carsten's program, he accesses the file scope array with the following pointer logic:

```
pQdb_Qdbfb = pQdb_Qdbfh + Qdb_Qdbfh.Qdbfos;
```

h) Scroll back up the manual page to the section entitled "File Definition Header (Qdb\_Qdbfh)." At offset 8 of this data structure, there's a field called Qdbfhflg. This is a field that is 16 bits long. Each bit corresponds to a different file attribute. The sixth bit of this field tells you whether a file has a keyed access path or not. If the sixth bit is on, the file has a keyed access path. If not, it doesn't. In the code, Carsten makes sure that this bit is on before attempting to read the keys from the file. If it is set on, he calls the GetKeyFlds subroutine, as follows:

```

      If  tstbts( %Addr( Qdb_Qdbfh.Qdbfhflg ): 6 ) = 1;
          ExSr  GetKeyFlds;
      EndIf;

```

i) In the GetKeyFlds subroutine, the key fields are loaded into an array.

As you learned by looking at the diagram, the key specification array can be found using a field in the Qdb\_Qdbfb data structure. More information about this is described in the section entitled "Key Specification Array (Qdb\_Qdbfk)" in the manual. It tells you that the offset that you need to jump to is in a field called Qdbfksof in the file scope array data structure.

This Qdbfksof field only points you to the first entry in the key fields array. If the file has more than one key field, you can find the additional fields by jumping forward in memory by the size of the Qdb\_Qdbfk data structure.

Carsten illustrates this process in the following code from the GetKeyFlds subroutine:

```
pQdb_Qdbfk = pQdb_Qdbfh + Qdb_Qdbfb.Qdbfksof;
```

```
For KeyIdx = 1 To Qdb_Qdbfb.Qdbfbgky;
```

```
    KeyFld(KeyIdx) = Qdb_Qdbfk.Qdbfkfld;
```

```
    If KeyIdx
```

More information about memory offsets, as they are used in APIs, was explained in the article "Getting Started with APIs, Part 2" in the May 20, 2004, issue of this newsletter and its follow-up article in the May 27, 2004, issue. You can read these articles at the following links:

<http://www2.systeminetwork.com/resources/clubtech/index.cfm?fuseaction=ShowNewsletterIssue&ID=18647>

<http://www2.systeminetwork.com/resources/clubtech/index.cfm?fuseaction=ShowNewsletterIssue&ID=18725>

More information about allocating memory to be used with APIs and another example of memory offsets can be found in the article "APIs by Example: List Job Object Locks" that was published in the July 15, 2004 newsletter. You can read that article at the following link:

<http://www2.systeminetwork.com/resources/clubtech/index.cfm?fuseaction=ShowNewsletterIssue&ID=19006>

The Print File Field Description program demonstrates the Retrieve Database File Description (QDBRTVFD) API. It is documented in the Information Center at the following link:

<http://publib.boulder.ibm.com/series/v5r2/ic2924/info/apis/qdbrtvfd.htm>

You can download the source code for this article from

<http://www2.systeminetwork.com/noderesources/code/clubtechcode/PrtFileFl>

The above source code was written by Carsten Flensburg. If you have any questions, you can contact Carsten at <mailto:flensburg@novasol.dk>.



**Source URL:** <http://iprodeveloper.com/rpg-programming/apis-example-print-file-field-description>