Recording information sent between 4D Client and 4D Server

By Jean-Yves Fock-Hoon, QA Manager, 4D, Inc.

Technical Note 06-03

Overview

This Technical Note will demonstrate you how to record and analyze requests exchanged between 4D Client and 4D Server.

Purpose

In 4D version 2004, the SET DATABASE PARAMETER command has been considerably improved. One of these improvements is the introduction of a new parameter, ID 28, (4D Server Log Recording). This parameter can be used in a Client/Server configuration and requires an additional numerical value as parameter.

- If the value is 0, 4D Server will stop recording.
- If the value is greater than 0, the value will be appended to the name of the request log file created. 4D Server will record any information related to any incoming requests in that request log file. The log file will be a text file placed next to the Structure file.

Once the file reaches a size of 10 MB, it is closed and a new file is generated, with an incremented sequence number. If a file of the same name already exists, it is replaced.

If you edit one of these files, you can read a collection of values, cryptic to most people. This technical Note will show you how to manipulate and read that file.

Structure of the file

If you edit this log file in a text editor, it looks like this:

1 nth log opened on Tuesday, January 10, 2006 03:17:09 PM

time	pid	uid	cid	request	bytes in	bytes out	duration	request name		
01/10/06, 15:17:09			4	26268080		26291080	189	24	11	0
01/10/06, 15:17:09			4	26268080		26291080	189	24	11	0
01/10/06, 15:17:09			4	26268080		26291080	189	24	11	0
01/10/06, 15:17:09			4	26268080		26291080	189	24	11	0

The file starts with a line that states the date and time of the creation of the file. The second line corresponds to the title of these columns. Other following lines would be values recorded for each incoming request.
As previously stated, it is difficult to understand these values if you do not understand the titles. Let's check them right Now.

The first column is time. It contains the date and time of the incoming request. In our example, date is in MM/JJ/YY, separated by a comma with the time in 24 hour format.

PID means Process ID. It is the Internal Process ID. This information cannot be tracked by the developer since it is internal to 4D. It's an internal process ID which is different from the process ID that the developer can see from the Interface or Runtime Explorer.

UID is the User ID. It is also an internal value for 4D Server. It is a number assigned to any user connecting to the Server. This is not the User ID defined by the 4D user login account such as Designer or Administrator. It is an internal value that you can also retrieve in the On Server Open Connection database method ($1 parameter).

CID is the Connection ID. This is also an internal value for 4D Server. It allows 4D Server to which connection this process belongs to. This value can also be retrieved in the On Server Open Connection database method with the parameter $2.

Request is the Request ID that will tell us what the purpose of the request is. This is the ID that we need to look at since it will tell us what exactly 4D Client is performing with 4D Server. The ID is unique and is not self-explanatory. You will find a brief description for each possible request at the end of this technical note.

Bytes In is be the size of the incoming packets, size of the received request. It is interesting to see if 4D Client is sending big packets or Not. For example simple requests should translate into very small incoming packets. Uploading records can generate bigger incoming packets.

Bytes Out is the size of the answer to that request. For each incoming request, 4D Server needs to respond and this value is the size of the response.
Typically, the response is big when the server is transferring information to the Client, such as downloading a big record.

Duration is the processing time in milliseconds for the request. It starts after the request is received, including parameters and indicates the time lapsed until the reply of the server is sent out. In most cases, the value should be equal to 0 since each request usually takes less than one

millisecond. Longer processing times occur when too many processes need to be handled, the network is flooded or the request is time consuming, such as a big SELECTION TO ARRAY.

Request name is, by default, empty. This is the name of the request. It can be retrieved if a specific resource has been installed in the resource file of 4D Server. Unfortunately, this is not very practical since you would have to recreate the resource each time an upgrade is performed. A quick workaround would be to store the name of these requests inside your structure file or data file and not worry about this resource anymore. You will find the list of all process IDs at the end of this technical note in appendix 1.

How to use the file

We can now understand the contents of that file but we still need to know how to use it. This feature is useful for optimization purposes only. This feature was first designed by and for 4D engineers to optimize the execution of 4D commands. Nevertheless, this option can also be useful for 4D developers if they want to reduce the number of requests between Client and Server and monitor the actual traffic their application creates.

Why would a 4D developer need to monitor low-level communication? The network is often the limiting factor in a Client/Server environment. If the latency is high, your clients will be slower and can also subject to more timeouts. If the latency is very low, such as on a LAN, adverse effects the clients can also be affected if too many of them are communicating with the Server at the same time.

Once a request has been executed, it is written to the log file; so keep in mind that, if the server receives too many requests, your whole system can slow-down because of 4D Server writing every request to the log. Accessing the disk will definitely affect performance on a ‘busy’ server. This is why it is not recommended to keep that feature always on but rather to use it carefully.

You can analyze the traffic that your method can generate. Knowing the number of packets that 4D Client will send for each of your methods can be very helpful. You may then optimize your code to reduce the number of packets. There is no direct relation between this and speed. In most cases, if you are sending fewer requests, you will use less network resources and less internal execution. Chances are greater to see your code running faster. However, it is possible that by trying to reduce the network traffic, you could end up with code that can run a little bit slower. That would be the price to pay in order to allow other processes to be faster or to allow 4D Server to handle more requests. This case is rare but it does not mean that it will never happen.

If you have many clients connected to the server and are worried about performance issues, it can be a good idea to have a look at all of the requests. By sorting these requests on the Process ID and Connection ID, you can see the name of the requests and guess which process this could be. A quick look can tell you if this can be optimized or if it is normal for that type of request can be executed that many times.

The Demonstration database

The Demonstration database is fairly simple and is provided as a way to analyze the request log file. As we have seen, we can just execute the SET DATABASE PARAMETER command with that selector and 4D Server will start to log all requests.

A text file is generated and you can edit it any text editor application. However the text file does not contain the name of these requests and it would be interesting to perform some statistics on those figures too. This is where our small database comes into play.

1- Start the database with 4D Server and connect with 4D Client.
2- Go to the Custom menu environment.
3- Select Request Log File from the Demonstration menu. A small dialog will be displayed with 3 buttons:
- Record: This turns the parameter ON or OFF. Click on this button to start recording all requests.
- Clear: This button executes a stored procedure on the Server to delete all request log files.
- View: Once we're recorded the file, click on that button to view it.

As we know, we can have multiple log files. The ID that we're using will be 1. You can change that value if needed but this is not really important. The most important thing to remember is that you can have multiple files for that ID since a new file will be created when the current file reaches 10 MB.

Let's assume that we are trying to measure our traffic with multiple clients. Therefore, 3 log files will be generated in our example.
With a small file, it would be easy to load all data into arrays and display them in a dialog with a listbox for example. But in our case, it would be unwise to load the 30 MB into arrays. This is why the database will save our data into the [Requests] table.

For visibility, the View button will display a dialog where you can view a tab control and different displays.
- Raw Data: This is a listbox where we can see all requests. Since we still can't display 30 MB in arrays, the data will be divided into sets. The number of records per set can be defined in the same dialog.

Change the number of records per set to increase or decrease the number of sets but just keep in mind that an increased value will require more memory since these are arrays. You can sort your columns and click on the four buttons at the bottom left in order to navigate thru these sets. When navigating within sets, the sorting order is kept, thanks to the use of named selections. This can also be a good example of how to display huge selections with listboxes.

- Mnemonics: This display will generate some statistics based on mnemonics, i.e. the name of these requests. When importing a log file, the name of the request will be pulled from the Mnemonics table according to the Request ID retrieved from the log file. The dialog will generate statistics from the Requests table by using the quick report editor. An xml file will be generated by the M_GenerateXMLStats method. We would just have to read that file and store our values into arrays. We can see the number of calls that a type of request has been called and the average and sum for the duration of the call, number of bytes in and out. This information might tell us if an improvement or optimization can be plausible.

- Processes: This will follow the same technique used for mnemonics. The M_GenerateXMLStats2 method will generate an xml file thru the quick report editor. A sum on Duration, Bytes In and Bytes Out will be computed per connection and process IDs. The result will be loaded in arrays and displayed it the current listbox. We can see if a process requires a lot of time or data transfer. It can tell us which process deserves more investigation if values are suspicious.
- Graphs: This part allows you to display 5 types of graphs.
    - Bytes per Request: This is a 2D Column chart that will show the sum of Bytes in and Bytes Out per type of request.
    - Bytes per time: This is a 2D Line chart that shows the sum of all bytes In and Out in the time.
    - All Durations: This is a 2D Line chart that shows the sum of all durations for all requests in the time.
    - % per request: This is a pie chart, showing the percentage of all requests based on the type of request.
    - Duration per Request: This is a 2D Column chart that will show the sum of all durations per type of request.

Summary

In this technical note, we saw that enabling the request log feature is very easy. We explained where to retrieve the log files and how to read them. Those steps are very easy. In a future technical note, we will explain how to evaluate those figures.

Appendix 1: List of ID and Request Name

ID	Mnemonics Short description	Internal Request
1	Process_EndOf End of a process from client	Yes
2	Rec_load Load record	No
3	Rec_LoadAndSelect Load record and reduce current selection to this record	Yes
4	Rec_Save Save record	No
5	Rec_GetNumInSelec Get record number inside selection	No
6	Sel_AddRec Add record to selection	No
7	Sel_Dim Set size of selection (init. of selection)	Yes
8	Rec_Delete Delete record	No
9	Sel_RemoveRec Remove record from selection	No
10	Rec_Unload Unload record	No
11	Trans_Start Start transaction	No
12	Trans_Validate Validate transaction	No
13	Trans_Cancel Cancel transaction	No
14	Sem_Set Set semaphore	No
15	Sem_Clear Clear semaphore	No
16	Rec_DeleteSel_Language Delete selected record (programming language)	No
17	Sel_AllRecords All records	No
18	Rec_RecInTable Count records in table	No
19	Set_Convert2Sel Convert set into selection	Yes
20	Sel_Convert2Set Convert selection into set	Yes
21	Rec_SelectedRecNumber Position of record in current selection	No
22	Sel_ReduceToCurrentRec Reduce current selection to current record	No
23	Struct_GetNbTablesAndFields Get number of tables & number of records per table	Yes
24	PRes_Get Get pseudo resource	Yes
25	PRes_Add Add pseudo resource	Yes
26	PRes_Write Write pseudo resource	Yes

ID	Mnemonics Short description	Internal Request
27	Obsolete_PRes_ Count Obsolete	Yes
28	Obsolete_PRes_ CountKind Obsolete	Yes
29	PRes_Remove Remove pseudo resource	Yes
30	PRes_GetInd Get pseudo resource	Yes
31	Obsolete_PRes_GetKind Obsolete	Yes
32	PRes_GetMap Get map of internal 4D resources at the first connection of a 4D Client (execute once only per 4D Client)	Yes
33	PRes_GetUniqueID Get new unique pseudo resource ID	Yes
34	PRes_GetResFileNum Get internal resource file reference number	Yes
35	Search Search	No
36	Sort Order by	No
37	Sel_CacheSelection Cache of current selection (send part of cache to client. Avoid sending a big selection through the network)	Yes
38	FlushCache Flush cache	No
39	Rec_DeleteSel_UserMode Delete selected record (User Mode)	No
40	Sel_RemoveRecordFrom Remove record in a selection (reduce selection, remove in selection)	Yes
41	Pooling Client keep alive (Interpreted: 8 seconds, Compiled: 30 seconds)	Yes
42	PRes_LoadAndLock Load & Lock a pseudo 4D resource	No
43	PRes_Unload Unload a pseudo resource on server	Yes
44	Auth_Challenge Send password	Yes
45	PRes_WhoLock Who lock a pseudo 4D resource	Yes
46	Auth_GetUserInfo Get user information	No
47	Ind_Drop Drop index	No
48	Ind_Create Create index	No
49	Sel_MoveToRec Goto in selection, without loading the record	Yes
50	Struct_GetTableDesc Get table description	Yes
51	Struct_CountTables Count tables	No
52	OPEN_GetAllFields Load a record and send all fields (Used by 4D Open)	No

ID	Mnemonics Short description	Internal Request
53	OPEN_UnlockRecord Unload all records from a table (4D Open)	No
54	OPEN_ModifyRecord Modify record - all fields (4D Open)	No
55	OPEN_NewRecord New record (4D Open)	No
56	OPEN_TrueGetRes Not used (4D Open)	No
57	Obsolete_TrueGetIndRes Obsolete	No
58	Obsolete_TrueGetResInfo Obsolete	No
59	Obsolete_TrueWriteRes Obsolete	No
60	Plug_GetList Get list of plug-ins	Yes
61	Obsolete_Plug_GetListExt Get list of externals (Obsolete)	No
62	Obsolete_GetCatList Obsolete	No
63	PRes_GetByName Get a pseudo resource by name	Yes
64	OPEN_GetRelatedFields Get a record and related records (4D Open)	No
65	OPEN_GetFieldNumberFromTables Get fields number from all asked tables (4D Open)	Yes
66	Struct_AddTable Create table	No
67	Plug_AddTable Add table by plug-in	No
68	Obsolete_ChangeCurrentTable Obsolete	No
69	Proc_GetList Get process list incl. related information	Yes
70	Obsolete_lockfile Obsolete	No
71	Obsolete_unlockfile Obsolete	No
72	Auth_ChallengeInfo Indicate to 4D Client if password should be used	Yes
73	Auth_Test Test password	No
74	User_GetList Get list of users	No
75	User_WritePassword Write modified password	No
76	Auth_TestPassword Test user password	Yes
77	User_WriteGroups Write groups	No
78	User_TestRights Test user rights	No
79	SearchByKey_Internal Query by index - Unique (Used by link)	Yes
80	User_WriteGroupsDesigner Write groups	No

ID	Mnemonics Short description	Internal Request
81	User_GetUsersAndGroups Get list of users & groups	Yes
82	User_WriteUsersAndGroups Write list of users & groups	Yes
83	Sel_Copy Copy selection	No
84	Sel_Use Use selection	No
85	Sel_Delete Delete selection	No
86	Sel_Move Move selection	Yes
87	Obsolete_Sel_Replace Obsolete	No
88	Rec_SequenceNumber Sequence Number	No
89	Rec_LoadAndSendData Load record and send data (fields)	No
90	Rec_LoadForModifyDisplaySelection Load record, called by Modify Selection, Display Selection or sub form.	Yes
91	Sel_SelectionToArray Selection To Array	No
92	Sel_ArrayToSelection Array To Selection	No
93	Sel_RelateMany Relate Many	No
94	OPEN_DistinctValues Distinct Values (4D Open)	No
95	Table_ReadOnly Read Only	No
96	Table_ReadWrite Read Write	No
97	SearchInSelection Query Selection	No
98	Obsolete_AddRecToSet Obsolete	No
99	OPEN_GetListNamesTables Get list of table name (4D Open)	No
100	Obsolete_Set_Empty Obsolete	No
101	User_NbConnected Number of connected users	No
102	User_NbUserProcess Number of user process	No
103	Math_Sum SUM (on field)	No
104	Math_Average Average (on field)	No
105	Math_Min Minimum (on field)	No
106	Math_Max Maximum (on field)	No
107	File_ReadFromDiskAndSendToClient Read file from disk and send it to a client	Yes
108	File_SendToServerAndWriteOnDisk Write a file on disk sent from a client	Yes

ID	Mnemonics Short description	Internal Request
109	Plug_ReadData Communication from a plugin (on client side) and server, To read on server	Yes
110	Plug_WriteData Communication from a plug-in (on client side) and server, To write on server	Yes
111	Bck_SetCurrentLogFile Set current log file	No
112	Bck_FullBackup Execute backup	No
113	Bck_IncBackup Increment backup	No
114	Set_ConvertListOfRecIntoSet Convert a selection from a displayed list into a set	Yes
115	Ind_CancelCreationRequest Creation of index request cancelled	No
116	Bck_SendInfoToBackup Send information to backup	No
117	Struct_GetTimestamp Read timestamp from resource 4D4D, to update files .res and .rex	Yes
118	Obsolete_PRes_CountKindStruct Obsolete	No
119	Obsolete_getindtypestruct Obsolete	No
120	Obsolete_countresstruct Obsolete	No
121	Struct_SendForkToClient Send fork side to a client	Yes
122	Sel_Join Join	No
123	Sel_Reduce Reduce selection	No
124	Ind_Scan Scan index	No
125	Rec_WhoLock_Language LOCKED ATTRIBUTES (programming language)	No
126	CurrentDateFromServer Current date (*)	No
127	CurrentTimeFromServer Current Time (*)	No
128	Sel_AutoLink Automatic links	No
129	Sel_CheckKind Check if current selection is a selection and not a set	Yes
130	Struct_SendAvailableAutoLink2Server Send a list of tables available for automatic links to server	Yes
131	Sel_AddRec Add record to selection	No
132	User_UpdatePluginsRightsAccess Update plugins authorisation rights	Yes
133	File_TestFilePath Verify if a file exists on server	Yes
134	Rec_SequenceNumberManagement Sequence Number Management	Yes
135	Rec_Lock Lock record	No

ID	Mnemonics Short description	Internal Request
136	Rec_Push Push record	No
137	Rec_Pop Pop record	No
138	Obsolete_searchanddelta Obsolete	No
139	Obsolete_setindex Obsolete	No
140	Obsolete_liaisonfatale Not used	Yes
141	OPEN_GetRelatedFieldsFormatted Get related fields of a selection incl. format (4D Open)	No
142	Trans_GetNbNewRecInside Get number of created records during a transaction	Yes
143	Plug_Get4DXCatInfo Get a description of folders Mac4DX, Win4DX and Plugins	No
144	File_SendFromServerToClient Send a file from server to client	Yes
145	Plug_CheckPluginsRightAccess Check and give access rights for plugins	No
146	Proc_ExecuteOnServer Execute on server	No
147	Proc_FindByName Process number	No
148	Proc_SetProcessVar Set process variable from client to a process on server	No
149	Proc_GetProcessVar Get process variable from a process on server to client	No
150	Search_SetLimit SET QUERY LIMIT	No
151	Search_SetDestination SET QUERY DESTINATION	No
152	Set_IsIn Is in set tests whether or not the current record for the table is in set.	No
153	Set_RecordsIn Records in set	Yes
154	Set_Create CREATE SET	No
155	Set_CreateEmpty CREATE EMPTY SET	No
156	Set_Add ADD TO SET	No
157	Set_Send Send a set to server	Yes
158	Set_Get Get a set from server	Yes
159	Set_Delete CLEAR SET	No
160	Set_Use USE SET	No
161	Set_Operation Operation on set (Difference, Intersection, Union, etc)	Yes
162	Set_Copy COPY SET	No

ID	Mnemonics Short description	Internal Request
163	Struct_UpdateInformation Modify an object in design mode. Update request on other 4D Clients	No
164	Set_GetList Get list of all sets	Yes
165	Sel_GetList Get list of all selection	Yes
166	Sel_DistinctValues Distinct Values	No
167	Set_FindTableFrom Return table from a set	Yes
168	PRes_GetIDByName Get ID of a pseudo resource by name	Yes
169	PRes_GetNameByID Get name of a pseudo resource by ID	No
170	PRes_WriteName Write name of a pseudo resource by ID	Yes
171	Array_RecNumsToSelOrSet Array longint containing rec.number to selection or set	No
172	Struct_SetDBParam Set database parameter	No
173	Proc_RegisterClient Register client	No
174	Proc_UnregisterClient Unregister client	No
175	Proc_GetListRegisteredClients Get Registered Clients	No
176	Proc_ExecuteOnClient Execute on Client	No
177	Proc_GetRequestProcToExec Check if a client has a method to be executed (execute on client)	Yes
178	Obsolete_UnlockRecords Obsolete	No
179	Plug_ValidateUserInfo Management of password for plugins	Yes
180	Plug_CheckUserInfo Management of password for plugins	Yes
181	Plug_GetListOfGroupsAndUsers Management of password for plugins	No
182	Plug_IsAdmin Management of password for plugins	No
183	Array_SelToRecNums Selection to array longint	No
184	Search_QueryWithArray Find in array	No
185	Struct_GetDataFile get name of current data file	No
186	Plug_SerialKey Serial key (SDK)	No
187	User_RegistrationProperties User name & company name (product registration)	Yes
188	User_NbCurrentUsers Number of current users vs. number of licences available	Yes
189	Rec_LoadForModifyDisplaySelection_New Load record to use with display selection, modify selection or sub form	Yes

ID	Mnemonics Short description	Internal Request
190	Struct_IsDatabaseLocked	
	Is data file locked?	No
191	User_IsDemo	
	Run in demo mode?	Yes
192	Struct_GetDBType	
	Compiled or interpreted mode?	No
193	Struct_GetLastOpeningMode	
	Last opening mode (compiled or interpreted)	Yes
194	PRes_GetListPseudoResources	
	Get list of files containing pseudo resources (.4DB, .4DA,etc.)	No
195	Struct_Update4DClient	
	Automatic update from 4D Client	No
196	Auth_Challenge2	
	see sendID (optimized)	No
197	Bck_GetLastBackupInformation	
	Get information from last backup	No
198	Bck_GetLastRestoreInformation	
	Return information from last restore	No