New Log Parser

By Timothy Aaron Penner, Senior Technical Services Engineer, 4D Inc.

Technical Note 17-20

Table of Contents

Abstract

Logs are an integral part of understanding the performance of an application and for troubleshooting. However, correlating multiple logs of different types together can be difficult. Understanding the format of the logs is a crucial first step, but having a tool to parse the logs can make this process much simpler. This Technical Note details the contents of the Request Log and Debug Logs and introduces a Log Parser that correlates the data together (v16R5+).

Introduction

This Technical Note talks about using the Debug Log and Request Log for performance profiling and troubleshooting purposes. The contents of each file are discussed and a new field added to the Request Log in v16R5 is explored. This Technical Notes includes a log parser that utilizes the new field that was added to the Request Log in v16R5. It allows the user to easily correlate the two logs together and view both the Debug Log data and the Request Log data together, organized by process.

Log File Definitions

The new log parser included in this Technical Note is used to correlate the Debug Log data with the Request Log data (v16R5+). It is important to have an understanding of the underlying logs that are being used before introducing the actual log parser.

The following sections cover these logs types.

Request Log

The request log records standard requests carried out by the 4D Server machine or the 4D remote machine that executed the command.

The request log is started with the following 4D code:

```
SET DATABASE PARAMETER(4D Server log recording;1)
```

The request log is disabled with the following 4D code:

```
SET DATABASE PARAMETER(4D Server log recording;0)
```

When enabled, the Request Log feature writes to two files:

- 4DRequestLog.txt
- 4DRequestLog_ProcessInfo.txt

The standard requests are logged to the 4DRequestLog.txt file while the 4DRequestLog_ProcessInfo file contains information about each process that is logged.

The following sections detail the file contents for each of these two files.

4DRequestLog.txt

This log file contains all of the standard requests that are carried out by the 4D Server.

The content of the file uses this format for each line of data:

Column	Field Name	Description
1	Sequence_number	Unique and Sequential operation number in the logging session
2	Time	Date and time using “MM/DD/YY, HH:MM:SS” format
3	System id	System ID
4	Component	Component signature (e.g. ‘4SQLS’ or ‘dbmg’)
5	Process_info_index	Corresponds to the “index” field in the 4DRequestsLog_ProcessInfo.txt file and permits linking a request to a process
6	Request	Request ID in C/S or message string for SQL requests or LOG EVENT messages
7	Bytes_in	Number of bytes received
8	Bytes_out	Number of bytes sent
9	Duration	Time took in milliseconds to perform action
10	Task_kind	Preemptive or cooperative (e.g. ‘p’ or ‘c’)

The actual *Request* (column 6) is usually a numeric value. This Technical Note includes a JSON formatted definition list that can also be updated via the internet. This is discussed further later on in this document in the *New Log Parser / Updating the Request ID’s via the web* section.

4DRequestLog_ProcessInfo.txt

This log file contains an index of information for each of the processes listed in the 4DRequestsLog.txt file. This file can be thought of as a Table of Contents to help organize the information from the 4DRequestsLog.txt file.

For each process, the following fields are logged:

Column	Field Name	Description
1	Sequence_number	Unique and Sequential operation number in the logging session
2	Time	Date and time using “MM/DD/YY, HH:MM:SS” format
3	Process_info_index	Corresponds to the “index” field in the 4DRequestsLog_ProcessInfo.txt file and permits linking a request to a process
4	CDB4DBaseContext	DB4D component database context UUID
5	System id	System ID
6	Server_process_id	Process ID on Server
7	Remote_process_id	Process ID on Client
8	Process_name	Process Name
9	cID	Identifier of 4D connection
10	uID	Identifier of 4D Client
11	IP	Client IPv4/IPv6 address
12	Host_name	Client hostname
13	User_name	User login name on client
14	Connection_uuid	UUID identifier of process connection
15	Server_process_unique_id	Unique process ID on server

Debug Log

The debug log records each event occurring at the 4D programming level such as method, commands, and plugins. Each event is logged to the file prior to executing the operation.

The debug log has two formats available; standard and tabular. The new log parser included with this Technical Note only works with the Tabular format but this document talks about both formats.

When the debug is logging it creates a file named 4DDebugLog.txt inside of the Logs folder of the database. A new log file is created each time the log reaches 10 MB.

4DDebugLog.txt (standard)

The standard format is the old format that was used in versions prior to v14. This mode provides a basic view of events. The contents of the file is human readable, there is no program needed to parse the information.

This format is very useful when troubleshooting a crash because scrolling to the bottom of this log will reveal in plain text the last operation the server had performed.

To start the debug log using the standard format issue the following 4D code:

```
SET DATABASE PARAMETER(Debug Log Recording;2)
```

To stop the debug log issue the following 4D code:

```
SET DATABASE PARAMETER(Debug Log Recording;0)
```

The contents of the log when using the standard mode uses this format:

Column	Description
1	Unique and Sequential operation number in the logging session
2	Elapsed time in milliseconds since log startup
3	Process ID (p=xx) and unique process id (puid=xx)
4	Stack Level
5	Can be command name, method name, message, task start/stop info, plugin, event or callback, connection uuid
6	Time taken for logging operation in milliseconds (different from 2nd

	column)

Note: The new log parser included with this Technical Note only works with the Tabular format.

4DDebugLog.txt (tabular)

The tabular format is the new format that was first available in v14; the contents of the file are more compact yet it includes additional information when compared to the standard format. The file is not human readable, it is expected that the information will be parsed by a computer.

To start the debug log using the standard format issue the following 4D code:

```
SET DATABASE PARAMETER(Debug Log Recording;2+4)
```

To stop the debug log issue the following 4D code:

```
SET DATABASE PARAMETER(Debug Log Recording;0)
```

The contents of the log when using the tabular mode uses this format:

Column	Description
1	Unique and Sequential operation number in the logging session
2	Elapsed time since log startup in “hh:mm:ss:ms” format (can be preceded by a day counter. For example, if the log was started 3 days ago, the time could be “3+11:58:23:163”)
3	Process ID
4	Unique process ID
5	Stack level
6	May represent (depending on the type of entry logged in 8th column): • A language command ID (when type=1) • A method name (when type=2) • A combination of pluginIndex:pluginCommand (when type=4, 5, 6, or 7). May contain something like ‘3:2’ • A Task Connection UUID (when type=8) • Or may contain a ‘starting sequence number’ when closing a stack level (this should correspond to the sequence number of the current action’s start)

7	Parameters passed to the commands, methods, or plugins
8	Log operation type: 1: Command 2: Method 3: Message (send by LOG EVENT command only) 4: Plugin Message 5: Plugin Event 6: Plugin Command 7: Plugin Callback 8: Task When a value is negative, it means that it is the closing stack level counterpart.
9	Form event if any; empty in other cases (suppose that column is used when code is executed in a form method or script)
10	Elapsed time (in micro seconds) of the current logged action; only for the closing stack levels

Given the following log data:

```
-- Startup on: Monday, October 30, 2017 12:52:38 PM --
1       12:52:38:336 5       69      0       CallMethod     "data" 2       0
2       12:52:38:336 5       69      1       41      "data" 1       0
3       12:52:39:937 5       69      1       2              -1      0       1600906
4       12:52:39:937 5       69      0       1              -2      0       1600963
```

Can be read like this:

1. Sequence 1 was process 5 or unique process 69 at stack level 0.
   The operation type is 2 meaning that the event was a Method call.
   The actual call was: CallMethod("data")

2. Sequence 2 was process 5 or unique process 69 at stack level 1.
   The operation type is 1 meaning that the event was a Command call.
   The command number was 41 which corresponds to ALERT.
   The actual call was: ALERT("data")

3. Sequence 3 was process 5 or unique process 69 at stack level 1.
   The operation was -1 meaning it was the closing of a command call.
   The sequence number listed in the data was 2 meaning this closure corresponds to sequence #2, the previous call to ALERT.

This closing line shows the duration it took for the event at sequence #2 to complete.

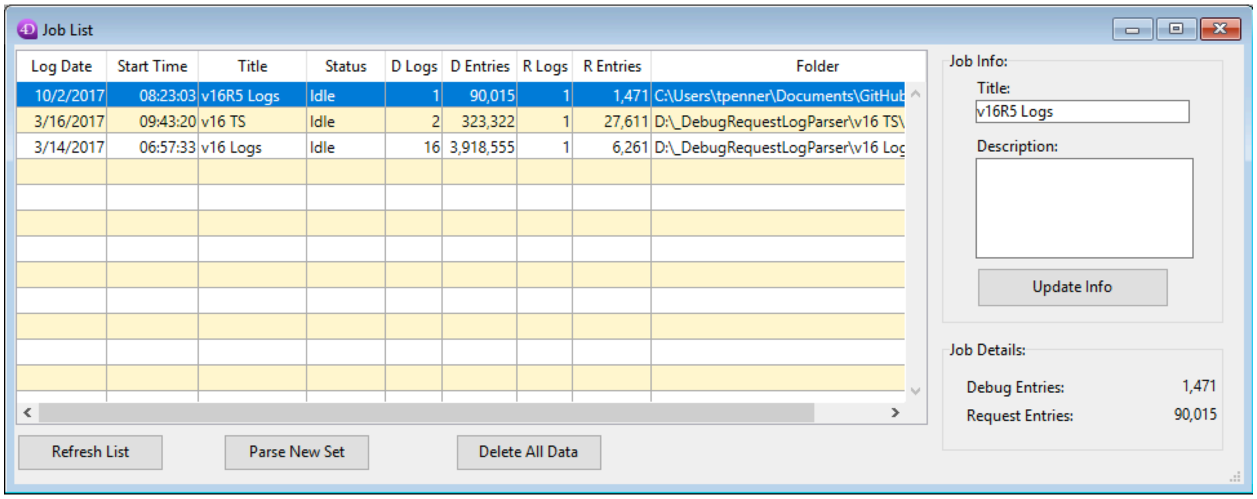
4. Sequence 4 was process 5 or unique process 69 at stack level 0. The operation was -2 meaning it was the closing of a method call. The sequence number listed in the data was 1 meaning this closure corresponds to sequence #1, the previous call to CallMethod("data"). This closing line shows the duration it took for the event at sequence #1 to complete.

New Log Parser

This Technical Note includes a sample database, the New Log Parser. When the database opens it automatically checks the GitHub repository for an updated set of Request Log definitions and then displays the Job List dialog. The Job List dialog contains all of the parsed sets of logs (known as 'Jobs'). Double clicking on a Job from this list will open the Job in the Log Viewer UI.

Job List

The Job List dialog displays all of the sets of logs that have been parsed into the current data file. From this UI the number of Debug Logs, Request Logs, as well as the number of log entries for each log type is displayed.



Log Date	Start Time	Title	Status	D Logs	D Entries	R Logs	R Entries	Folder
10/2/2017	08:23:03	v16R5 Logs	Idle	1	90,015	1	1,471	C:\Users\tpenner\Documents\GitHub
3/16/2017	09:43:20	v16 TS	Idle	2	323,322	1	27,611	D:_DebugRequestLogParser\v16 TS\
3/14/2017	06:57:33	v16 Logs	Idle	16	3,918,555	1	6,261	D:_DebugRequestLogParser\v16 Log

Job Info:
Title: v16R5 Logs
Description:

Job Details:
Debug Entries: 1,471
Request Entries: 90,015

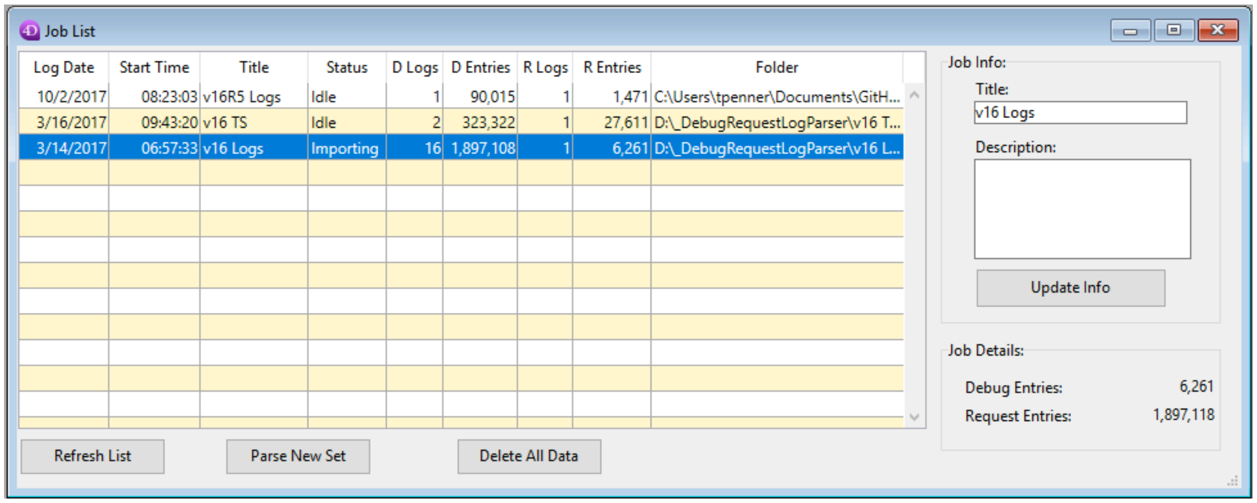
Selecting a Job from the list updates the Title and Description information on the side of the window. While a job is selected the Title and Description can be updated. To save the changes click on the *Update Info* button.

Double clicking on a row will open that Job in the Log Viewer.

Parsing a set of logs

Parsing a set of logs can be accomplished by either using the *Parse New Set* button on the Job List dialog or from the File menu using the *Parse Logs Folder* menu item. Activating this feature will bring up a folder selection dialog; select the folder that contains either Debug Logs, Request Logs, or both. The folder will immediately start to be imported into the data file.

While the import is taking place the Jobs List dialog is updated showing the progress. The Status column for the job being imported will display “Importing” while the others display “Idle”. The number of log entries is updated every second showing the overall progress of the import.


Job List

Log Date	Start Time	Title	Status	D Logs	D Entries	R Logs	R Entries	Folder
10/2/2017	08:23:03	v16R5 Logs	Idle	1	90,015	1	1,471	C:\Users\tpenner\Documents\GitH...
3/16/2017	09:43:20	v16 TS	Idle	2	323,322	1	27,611	D:_DebugRequestLogParser\v16 T...
3/14/2017	06:57:33	v16 Logs	Importing	16	1,897,108	1	6,261	D:_DebugRequestLogParser\v16 L...

Job Info:
Title: v16 Logs
Description:

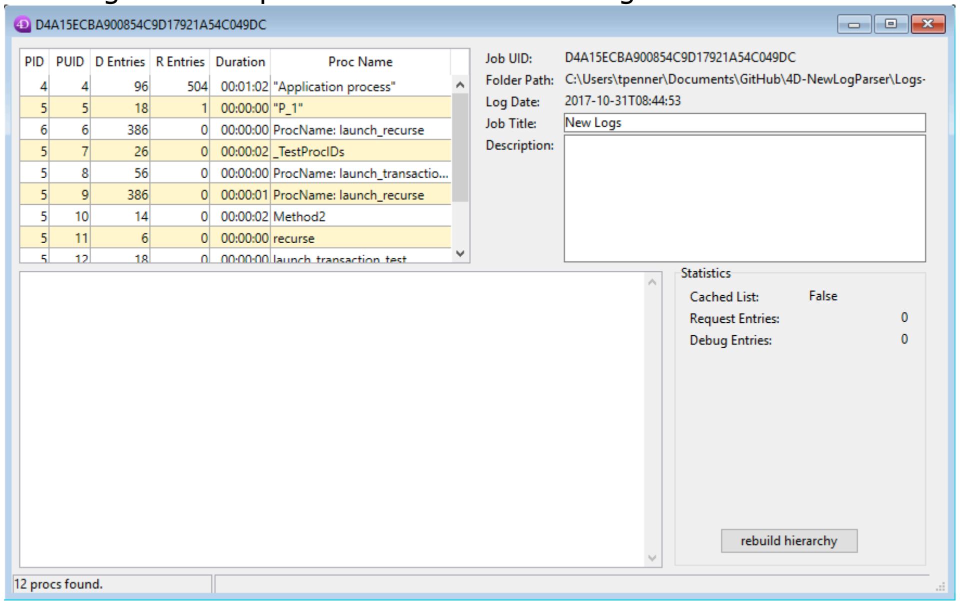
Job Details:

Debug Entries:	6,261
Request Entries:	1,897,118

The Job cannot be opened in the Log Viewer while the status says Importing.

Log Viewer

The Log Viewer opens after double clicking on a Job from the Job List dialog:


D4A15ECBA900854C9D17921A54C049DC

PID	PUID	D Entries	R Entries	Duration	Proc Name
4	4	96	504	00:01:02	"Application process"
5	5	18	1	00:00:00	"P_1"
6	6	386	0	00:00:00	ProcName: launch_recurse
5	7	26	0	00:00:02	_TestProcIDs
5	8	56	0	00:00:00	ProcName: launch_transactio...
5	9	386	0	00:00:01	ProcName: launch_recurse
5	10	14	0	00:00:02	Method2
5	11	6	0	00:00:00	recurse
5	12	18	0	00:00:00	launch_transaction_test

Job UID:	D4A15ECBA900854C9D17921A54C049DC
Folder Path:	C:\Users\tpenner\Documents\GitHub\4D-NewLogParser\Logs-
Log Date:	2017-10-31T08:44:53
Job Title:	New Logs
Description:	

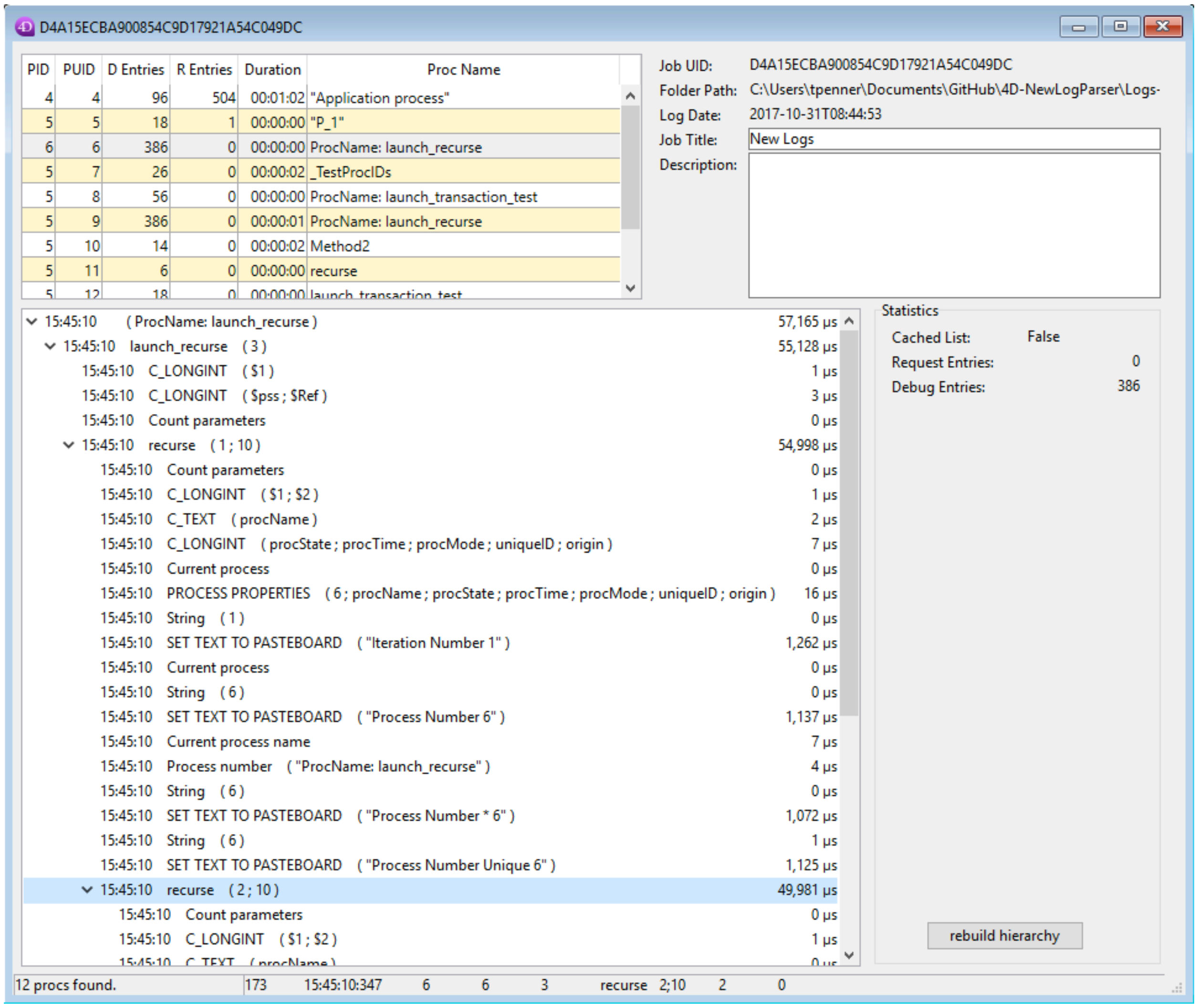
Statistics

Cached List:	False
Request Entries:	0
Debug Entries:	0

12 procs found.

In the upper left corner of this dialog is the list of processes. Starting at the left, the columns are; Process ID (pid), Unique Process ID (puid), the number of debug log entries (d entries), the number request log entries (r entries), the duration of the process (duration), and finally the Process Name (proc name).

Selecting a process from the list will build a hierarchical list from the process information. This hierarchical list is archived in the database for better performance but it can be rebuilt by clicking on the *rebuild hierarchy* button in the bottom right of the window.



D4A15ECBA900854C9D17921A54C049DC

PID	PUID	D Entries	R Entries	Duration	Proc Name
4	4	96	504	00:01:02	"Application process"
5	5	18	1	00:00:00	"P_1"
6	6	386	0	00:00:00	ProcName: launch_recurse
5	7	26	0	00:00:02	_TestProcIDs
5	8	56	0	00:00:00	ProcName: launch_transaction_test
5	9	386	0	00:00:01	ProcName: launch_recurse
5	10	14	0	00:00:02	Method2
5	11	6	0	00:00:00	recurse
5	12	18	0	00:00:00	launch_transaction_test

Job UID:	D4A15ECBA900854C9D17921A54C049DC
Folder Path:	C:\Users\tpenner\Documents\GitHub\4D-NewLogParser\Logs-
Log Date:	2017-10-31T08:44:53
Job Title:	New Logs
Description:	

```
15:45:10   ( ProcName: launch_recurse )                                              57,165 µs
  15:45:10   launch_recurse   ( 3 )                                                  55,128 µs
    15:45:10   C_LONGINT   ( $1 )                                                     1 µs
    15:45:10   C_LONGINT   ( $pss ; $Ref )                                            3 µs
    15:45:10   Count parameters                                                      0 µs
    15:45:10   recurse   ( 1 ; 10 )                                               54,998 µs
      15:45:10   Count parameters                                                    0 µs
      15:45:10   C_LONGINT   ( $1 ; $2 )                                              1 µs
      15:45:10   C_TEXT   ( procName )                                                2 µs
      15:45:10   C_LONGINT   ( procState ; procTime ; procMode ; uniqueID ; origin )  7 µs
      15:45:10   Current process                                                     0 µs
      15:45:10   PROCESS PROPERTIES   ( 6 ; procName ; procState ; procTime ; procMode ; uniqueID ; origin )   16 µs
      15:45:10   String   ( 1 )                                                       0 µs
      15:45:10   SET TEXT TO PASTEBOARD   ( "Iteration Number 1" )                 1,262 µs
      15:45:10   Current process                                                     0 µs
      15:45:10   String   ( 6 )                                                       0 µs
      15:45:10   SET TEXT TO PASTEBOARD   ( "Process Number 6" )                   1,137 µs
      15:45:10   Current process name                                                7 µs
      15:45:10   Process number   ( "ProcName: launch_recurse" )                      4 µs
      15:45:10   String   ( 6 )                                                       0 µs
      15:45:10   SET TEXT TO PASTEBOARD   ( "Process Number * 6" )                 1,072 µs
      15:45:10   String   ( 6 )                                                       1 µs
      15:45:10   SET TEXT TO PASTEBOARD   ( "Process Number Unique 6" )            1,125 µs
      15:45:10   recurse   ( 2 ; 10 )                                             49,981 µs
        15:45:10   Count parameters                                                  0 µs
        15:45:10   C_LONGINT   ( $1 ; $2 )                                            1 µs
        15:45:10   C_TEXT   ( procName )                                              0 µs
```

Statistics

Cached List:	False
Request Entries:	0
Debug Entries:	386

```
12 procs found.    173    15:45:10:347    6    6    3    recurse  2;10    2    0
```

On the right side of this window is a statistics field that includes the number of Request Entries and Debug Entries found for the selected process and whether or not the currently displayed hierarchicial list is cached or not. If the list, it can be rebuilt by clickintg on the *rebuild hierarchy* button.

Updating the Request ID's via the web

The Request Log uses a numerical value to represent what operation took place. These numerical values combined are referred to as Request ID's. The database has a table of definitions but this information can also be updated from the web.

To update the local database run the ***ImportRequestIDs*** project method. This method downloads the latest requestIDs.txt file from GitHub and saves a copy locally.

This method is automatically called when starting up with a blank data file.

Note: The GitHub repository containing the Request ID definitions is
https://github.com/4D/4d-request-log-definitions

Conclusion

This Technical Note talked about using the Debug Log and Request Log for troubleshooting purposes. The contents of each file were discussed and the new field added to the Request Log in v16R5 was explored. A log parser was included that, thanks to this new field added in v16R5, allows the user to view both the Debug Log data and the Request Log data in line with each other organized by process. This should provide the developer with the tools necessary to start analyzing their debug/request logs and also provide the information necessary for the developer to extend the parser to better suit their needs.

Additional resources

Description of Logs:
http://livedoc.4d.com/4Dv16-R5/help/Title/en/page8942.html
Docs -> Design Reference -> Appendixes -> Appendix E: Description of Logs