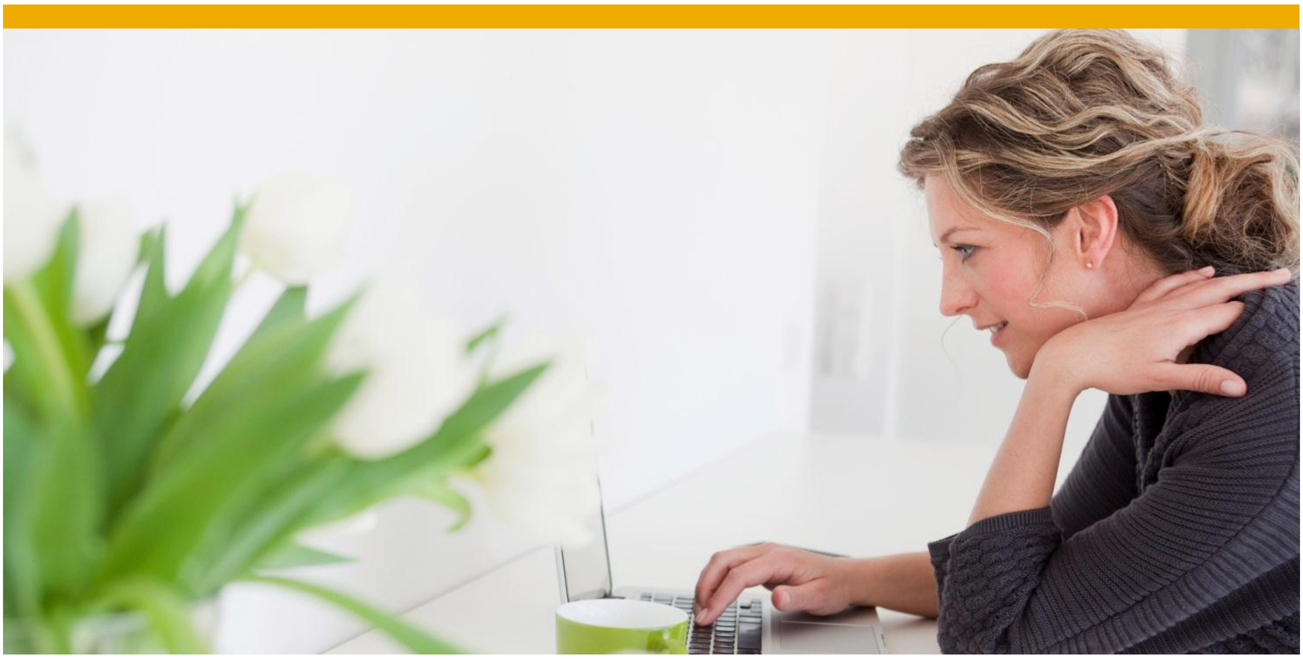


Performance Measurement in Android

This document and code samples provided here is valid for product Sybase Unwired Platform (SUP) - Online Data Proxy (ODP), release version 2.2.

This document describes how to get and understand performance measurement logs from an Android OData application during different intervals added in OData Client SDK as well as using Performance Library as standalone library. This document also describes about how to enable the performance logs in OData SDMParser.

Suhas Narasimhan
(Associate Developer, Android Platform, ODP, Mobile Platform IND)

TABLE OF CONTENTS

INTRODUCTION

This document includes the details and code snippets which will help the application developers to analyze the CPU time lapsed and memory consumed between various intervals by enabling the performance logs for the applications developed using OData Client SDK.

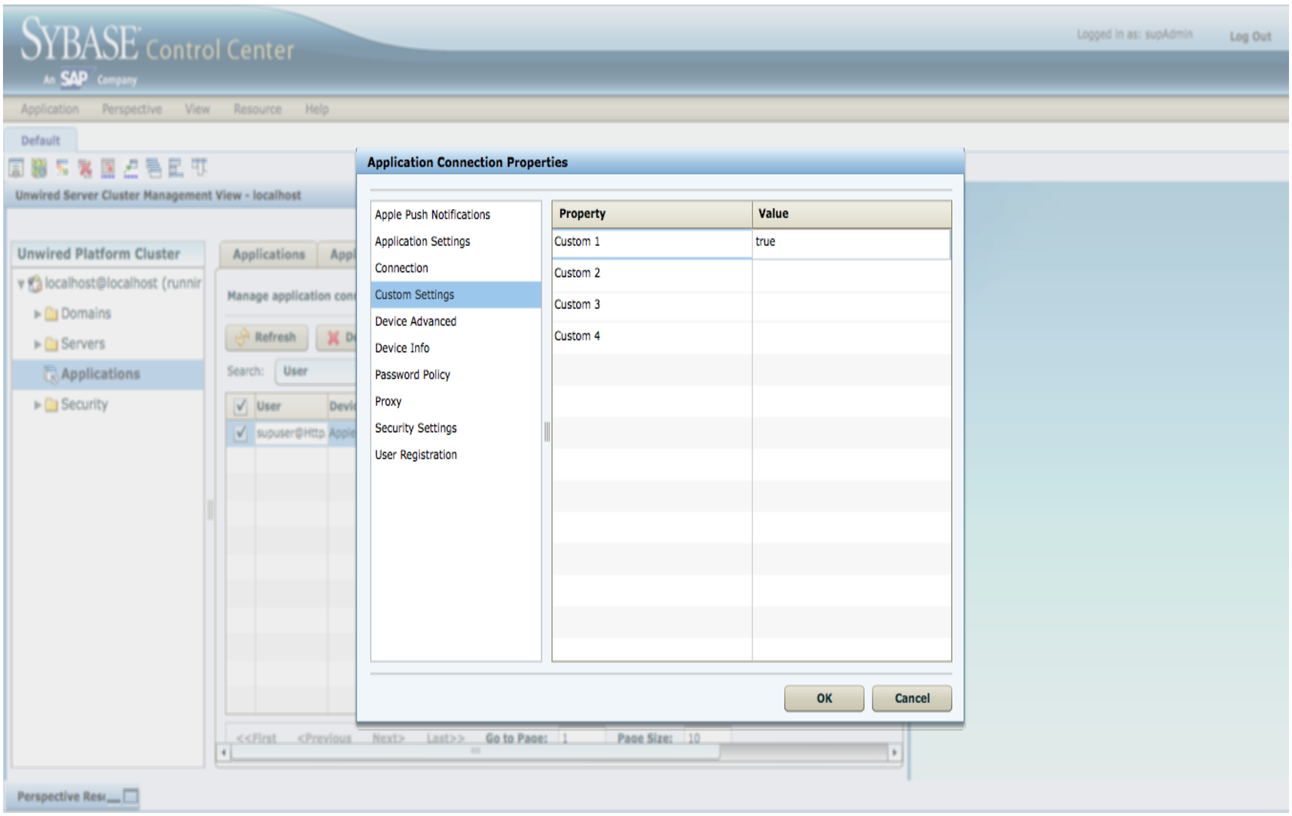
By using the performance libraries application developers can instrument their own coding with the help of the libraries while at the same time reducing time and effort needed for the implementation. With this, the compliance to the product standard, i.e, performance for mobile applications can also be checked.

This document details the steps involved in enabling performance logging for an OData application which uses the SUPProxyClient library via the SCC and how to analyze the logs. The document also explains how to use the performance library as a standalone library for measuring various performance data for any android application. Application developers can also use provided API to enable the logs in OData SDMParser to get the CPU time lapsed and memory used during parsing.

HOW TO ENABLE AND ANALYSE PERFORMANCE LOGS

On SUPProxyClient Library

Performance logs are already added in OData Client SDK for certain intervals such as Network time, Request/Response, End to End etc which can be viewed on console/logs.txt file by setting *Custom1* property under Custom Settings of Application Connection Properties in SCC to “***true***” (case sensitive) as shown in below screenshot. Setting the property “Custom1” to true will enable the performance logging in the application which is using the SUPProxyClient Library.



Property	Value
Custom 1	true
Custom 2	
Custom 3	
Custom 4	

Sample Console Performance logs

While logging the performance measurements, we first log the various KPIs names under the tag “Headers” and then the various values for these KPIs are logged under the tag “Readings”.

Network Timings:

```
12-06 23:40:27.418:
NETWORK:
RR-Headers:
NAME;ELAPSED_TIME;CPU_TIME;START_TIMESTAMP;STOP_TIMESTAMP;TRAFFIC_SENT;TRAFFIC_RECEIVED;PACKETS_SENT;PACKETS_RECEIVED;TRAFFIC_TOTAL;REQUEST_BYTE_COUNT;RESPONSE_BYTE_COUNT

12-06 23:40:27.428: D/ODP(347):
NETWORK:
RR-Readings:TtO56f1M-NETWORK:RR;2334;12;514310;516644;0;0;3;3;0;;346;0
```

The above logs are the performance measured for the amount of time taken by the network during a request-response.
ELAPSED_TIME: 2334 represents that the total time taken by the network during the request response to 2334 milliseconds.

RR Timings:

```
12-06 23:40:27.498: D/ODP(347):
RR-Headers:
NAME;ELAPSED_TIME;CPU_TIME;START_TIMESTAMP;STOP_TIMESTAMP;TRAFFIC_SENT;TRAFFIC_R
ECEIVED;PACKETS_SENT;PACKETS_RECEIVED;TRAFFIC_TOTAL;
12-06 23:40:27.508: D/ODP(347):
RR-Readings:TtO56f1M-RR;2724;239;514002;516726;0;0;27;30;0;
```

The above logs are the performance measured for the amount of time taken by the application for request-response including the iMO component.
ELAPSED_TIME: 2742 represents that the total time taken by the application for request-response including the iMO component is 2742 milliseconds

iMO Timings:

```
Here to obtain the amount of time taken by the iMO component can be deduced by
using the RR timings that are logged as below:

iMO loop time = RR_Elapsed_Time – Network_RR_Elapsed_Time
iMO loop time = 2742 – 2334
              = 408 milliseconds
Hence, the iMO component has consumend 408 milliseconds in this request-reponse.
```

End-to-End Timings:

```
12-06 23:40:27.518: D/ODP(347):
E2E-Headers:
NAME;MEMORY_MAX;MEMORY_ALLOCATED;ELAPSED_TIME;CPU_TIME;START_TIMESTAMP;STOP_TIME
STAMP;
E2E-Readings:TtO56f1M-E2E;4167;247;2873;376;513969;516842;
```

The above measured performance logs indicate the total time taken for the entire request-response including the time taken by the Android ODP SDK.

“ELAPSED_TIME: 2873” represents that the total time taken for the entire request-response including the time taken by the Android ODP SDK is 2873 milliseconds

OData Client SDK Timings:

Here to obtain the amount of time taken by the ODP SDK component can be deduced by using the E2E timing and RR timing that are logged as below:

OData Client SDK time = E2E_Elapsed_Time - RR_Elapsed_Time
OData Client SDK time = 2873 - 2742
                                    = 131 milliseconds
Hence, the OData Client SDK component has consumed 131 milliseconds in this request-response.

Using Performance Library as Standalone Library

Each user interaction of your test case will be reported as a single named entity, for which different KPIs like end-to-end response time will be measured.

In case, a developer wants to use the standalone performance library. For each user interaction, several measurement intervals can be instrumented. Currently the instrumentation needs to be done by the developer. As a developer, you are responsible to start and stop a certain interval (e.g. "http request") at the proper point in time in your code. The library only makes sure that the performance counters are measured.

Reporting of the data for each interaction step (incl. the intervals) can be done on the log output (logcat) or the data can be written to a log file which can be downloaded to view the measurement data.

Application developers can directly use “perflib.jar” as standalone library to get the Performance logs in their Android application to track the various parameters such as time taken or memory consumed for various method calls or for the end-to-end run of the application.

- Add following code snippet:

```
/*** Code to add before any interval ****/

// Initialize the object of PerformanceAgent Class

PerformanceAgent pa = PerformanceAgentImpl.getInstance();
pa.setContext(getApplicationContext());
pa.setApplicationName("com.sap.demo.travelapp");
//com.sap.performance.android.testapp -> to be set to your application


PerformanceAgent pa = PerformanceAgentImpl.getInstance();
     pa.startInterval("Interval_Name",
 com.sap.performance.android.lib.intervals.IntervalsType.HttpRequest);

 /*** Code to add after  interval ****/
    pa.stopInterval("Interval_Name");

     /*** Print the output on console ****/
    log.p("Tag_Name", "Headers:"+
pa.getInterval("Interval_Name").getIntervalDescriptionAsString());

    log.p("Tag_Name", "Readings:"+
pa.getInterval("Interval_Name").getIntervalDataAsString());
```

Sample Console Performance Logs for an Interval

```
12-06 23:40:49.258:
Headers:NAME;MEMORY_MAX;MEMORY_ALLOCATED;ELAPSED_TIME;CPU_TIME;START_TIMESTAMP;S
TOP_TIMESTAMP;
12-06 23:40:49.268:
Readings: Interval_Name;4487;35;3364;1457;531562;534926;
```

ENABLE PERFORMANE LOGS IN ODATA SDMPARSER

To enable the performance Logs in SDMParser, application developer can follow below steps:

- Initialize the SDMParser

```
SDMLogger logger = new SDMLogger();

SDMPreferences pref = new SDMPreferences(getApplicationContext(), logger);

logger.setLogLevel(ISDMLogger.DEBUG);
logger.logToAndroid(true); // Logs are written to the console

parser = new SDMParser(pref, logger);
```

- To enable the performance log for parser, add following code snippet :

```
parser.enableParserPerformanceLog(true, getApplicationContext());
```

Logs can be seen on console if the logToAndroid(true) is called, otherwise the logs are written to the SDM Logger log file in the device.

Sample Console Performance Logs for SDM Parser

```
12-06 23:40:31.288: I/SDMParser(347): Parsing Time -
Headers:NAME;ELAPSED_TIME;CPU_TIME;START_TIMESTAMP;STOP_TIMESTAMP;
12-06 23:40:31.288: I/SDMParser(347): Parsing Time -
Readings:parseSDMODataServiceDocumentXML;68;68;520470;520538;

12-06 23:40:39.917: I/SDMParser(347): Parsing Time -
Headers:NAME;ELAPSED_TIME;CPU_TIME;START_TIMESTAMP;STOP_TIMESTAMP;
12-06 23:40:39.927: I/SDMParser(347): Parsing Time -
Readings:parseSDMODataSchemaXML;373;289;528795;529168;

12-06 23:40:50.918: I/SDMParser(347): Parsing Time -
Headers:NAME;ELAPSED_TIME;CPU_TIME;START_TIMESTAMP;STOP_TIMESTAMP;
12-06 23:40:50.918: I/SDMParser(347): Parsing Time -
Readings:parseSDMODataEntriesXML;785;757;539376;540161;
```

Here in the above performance logs, the “Readings” tag is followed by the name of the API and the time taken by the API.

www.sap.com