Review of integration points

SAS & Hadoop intersect in many ways:

- SAS can treat Hadoop just as any other data source, pulling data FROM Hadoop, when it is most convenient;

- SAS can work directly IN Hadoop, leveraging the distributed processing capabilities of MapReduce.

- SAS can work directly WITHIN Hadoop, lifting data on HDFS into a SAS advanced analytics in-memory environment;
Review of SAS’ technology direction

SAS/Access to Hadoop - Extract data from Hadoop into SAS

FROM
Hive
Impala

Embedded Process - Push *some* SAS processing to Hadoop with Map Reduce

IN
Score A
Code A

DQ

Within
HPA
LASR

In-Memory Analytics - Use Hadoop for Storage persistence, workload mgmt. and commodity computing.
Access engines overview

SAS/Access to Hadoop, Hawq or Impala - Push *some* of SAS’ processing to Hadoop
SAS Accelerators for Hadoop overview

SAS/Embedded Process - Push SAS processing to Hadoop with Map Reduce

SAS/Scoring Accelerator for Hadoop
SAS/Data Loader for Hadoop
• SAS/Code Accelerator for Hadoop
• SAS/Data Quality Accelerator for Hadoop

```sas
proc ds2;
/* thread ~ equiv to a mapper */
thread map_program;
method run();
set dbmslib.intab;
/* program statements */
end; endthread; run;
/* program wrapper */
data hdf.data_reduced;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
dcl thread map_program map_pgm;
```

Copyright © 2015. SAS Institute Inc. All rights reserved.
In-memory analytics overview

SAS ANALYTIC HADOOP ENVIRONMENT

In-Memory Analytics – Process in Memory, use Hadoop for Storage persistence and commodity computing

WEB CLIENTS

APPLICATIONS

Visual Analytics
Visual Statistics
In-Memory Statistics
Visual Scenario Designer
HPA

SAS® LASR ANALYTIC SERVER

SAS® IN-MEMORY
SAS® IN-MEMORY
SAS® IN-MEMORY
SAS® IN-MEMORY
SAS® IN-MEMORY

HADOOP

ERP
SCM
CRM
Images
Audio and Video
Machine Logs
Text
Web and Social
What’s coming for Hadoop in July with 9.4 M3
Major Hadoop themes for SAS 9.4 M3

SAS 9.4M3

YARN

Simpler Install & Config

Access And Files

Performance

DS2

Expanded Distribution Support
Tighter YARN integration

- Continue the momentum of 9.4M2 where LASR & HPA first became YARN aware.

- SAS EP now runs in same process as the MapReduce JVM – more tightly coupled resource
  - Fully integrated with YARN Resource Manager
  - SAS EP can now write to MR job logs through put statements
  - Performance improvements over M2
Easier deployment

• SAS Deployment wizard can now gather required jars
• Integration of SAS EP with Hadoop distributor administration tools to simplify SAS install/configuration
  • integration with Cloudera Manager and Ambari
  • Packages and parcels
  • No root required – SUDO only
  • Does *not* require two way SSH keys setup
Access engine improvements

- SAS/ACCESS to Hadoop
  - Support for BINARY and DECIMAL data types
  - Implicit Pass-through improvements
    - READ_METHOD=HDFS honored
  - CTRL-C Query Interrupt
  - Improved error messaging
- SAS/Access to HAWQ including Proc pushdown support
- SAS/Access to Impala adds Proc pushdown support
File type support

- Increased ability to read and write Hadoop file types using SAS EP
  - Full EP support for Parquet, Avro, JSON, ORC and compressed Sequence files.
- SerDe’s to make reading SAS proprietary file types easy for Hadoop Community e.g. SPDE, SASHDAT
- PROC SQOOP - GA
Expanded distribution support

- Expand SAS’ Distribution Support (SAS/Access, EP, HPA, LASR)
  - Near Parity between CDH / HWX and MapR, Pivotal and IBM
  - New HAWQ access Engine

- Expand Hadoop Pushdown Processing
  - Data Step / DS2 – Merge, Proc Transpose
  - SPDE on HDFS
  - Improved WHERE pushdown: AND, OR, NOT, parenthesis, range operators and in-lists all supported
  - Parallel write support can improve write performance up to 40%
New products with SAS 9.4 M3

- **SAS Grid**
  - SAS Grid Manager for YARN
  - Hadoop support is an alternative, not a replacement for LSF

- **SPDS**
  - SPDS supports Hadoop (5.2)
  - Ability to read, write and update SPD Server tables stored in Hadoop
  - Kerberos only
An Introduction to DS2 on Hadoop

Doug Green
What is DS2?

• DATA step like distributed processing for Hadoop and other MPP platforms (think do loops, arrays, statements, by-group processing, functions etc.)

• Appropriate for advanced data manipulation and data modelling applications especially those that are difficult or impossible to achieve through SQL (e.g. transposing data)

• Object orientated programming environment

• Runs in the MapReduce/YARN framework on Hadoop

• Portable across platforms
The SAS Embedded Process

A portable, lightweight execution container for SAS code that makes SAS portable and deployable on a variety of platforms

1. Data Lifting
2. Data Preparation
3. Data Quality
4. Scoring

OPTIONS DS2ACCEL=ANY DSACCEL=ANY;
proc ds2;

/*---
**MAP
*/
thread map_program / overwrite=yes;
method run();
set Hadoop.&source;
/* program statements */
end;
endthread;

/*-----
**REDUCE
*/
data hadoop.&target overwrite=yes);
dcl thread map_program MapReduce;
method run();
set from MapReduce;
end;
enddata;
run;
quit;
A simple example for Hadoop
DS2 syntax framework for Hadoop

1. Hadoop libname
2. SAS Options
3. Create thread program
4. DS2 logic
5. Call thread program

```sas
/*HIVE libname */
libname hadoop hadoop SUBPROTOCOL=hive2 READ_METHOD=HDFS schema=sukdam user=sukdam pwd="/18A8002]E043FE4757B4CE074DC2458F2E9204D532827D8D2A0DA252
server="XXXXXXXXXXXXX" port=10001 ;

let source=HADOOP_SOURCE_TABLE;
let target=HADOOP_TARGET_TABLE;

OPTIONS DS2ACCEL=ANY DSACCEL=ANY;
proc ds2 ;
/*---
**MAP PHASE
*/
thread map_program / overwrite=yes;
method run();
get Hadoop.Source;
/* DS2 program statements */
end;
endthread;
/*------
**REDUCE PHASE (If by Statement used)
*/
data hadoop.Target overwrite=yes);
dcl thread map_program MapReduce;
method run();
get from MapReduce;
end;
enddata;
run;
quit;
```
An important option!

- For any DS2 thread program to run in Hadoop the following SAS option must be set:

```sas
21
22 options DS2ACCEL=ANY;
23
```
data test;
  input i j x;
  datalines;
  1 1 123
  1 1 3245
  1 2 23
  1 2 543
  1 3 87
  1 3 90
  2 1 88
  2 1 86
;
/* When the first observation in each BY-Group is read, the variables JSUB and */
/* FREQ are initialized to zero and with each subsequent observation in the */
/* BY-Group, FREQ is incremented by one and JSUB is incremented by the value of */
/* X. When the last observation in the BY-Group is read, AVER is created by */
/* dividing JSUB by FREQ to determine the average value for the group. */

data jsubtot (keep=i j freq aver);
  set test;
  by i j;
  retain jsub freq;
  if first.j then do;
    jsub=0;
    freq=0;
  end;
  jsub + x;
  freq + 1;
  if last.j then do;
    aver=jsub/freq;
    output;
  end;
run;
proc print;
run;
DS2 equivalent for Hadoop

```
proc ds2;
  thread compute / overwrite=yes;
  declare double jsub freq aver;
  retain jsub freq;
  keep i j freq aver;
  method run();
  set hdp.test;
  by i j;
  if first.j then do;
    jsub=0;
    freq=0;
  end;
  jsub + x;
  freq + 1;
  if last.j then do;
    aver=jsub/freq;
    output;
  end;
endthread;
data hdp.jsubtot (overwrite=yes);
  declare thread compute t;
  method run();
  set from t;
  end;
enddata;
run;
quit;
```
DS2 equivalent for Hadoop

```sas
proc ds2;
    thread compute / overwrite=yes;
    declare double jsub freq aver;
    retain jsub freq;
    keep i j freq aver;
    method run();
    set hdp.test;
    by i j;
    if first.j then do;
        jsub=0;
        freq=0;
    end;
    jsub + x;
    freq + 1;
    if last.j then do;
        aver=jsub/freq;
        output;
    end;
endthread;
data hdp.jsubtot (overwrite=yes);
    declare thread compute t;
    method run();
    set from t;
    end;
enddata;
run;
quit;
```

DS2 is a SAS procedure and is therefore invoked through SAS procedure syntax.
To run in-database, a thread program must be used. The SAS Code Accelerator enables you to publish a DS2 thread program and execute that thread program in parallel inside Hadoop.
DS2 equivalent for Hadoop

```
proc ds2;
    thread compute / overwrite=yes;
    declare double jsub freq aver;
    retain jsub freq;
    keep i j freq aver;
    method run();
    set hdp.test;
    by i j;
    if first.j then do;
        jsub=0;
        freq=0;
    end;
    jsub + x;
    freq + 1;
    if last.j then do;
        aver=jsub/freq;
        output;
    end;
end;
endthread;
data hdp.jsubtot (overwrite=yes);
    declare thread compute t;
    method run();
    set from t;
    end;
enddata;
run;
quit;
```
DS2 equivalent for Hadoop

```sas
proc ds2;
  thread compute / overwrite=yes;
    declare double jsub freq aver;
    retain jsub freq;
    keep i j freq aver;
    method run();
    set hdp.test;
    by i j;
    if first.j then do;
      jsub=0;
      freq=0;
    end;
    jsub + x;
    freq + 1;
    if last.j then do;
      aver=jsub/freq;
      output;
    end;
  endthread;
data hdp.jsubtot (overwrite=yes);
  declare thread compute t;
  method run();
  set from t;
  end;
enddata;
run;
quit;
```

DS2 has new data types, more akin to an RDBMS, and should be explicitly declared. E.g. VARCHAR, DOUBLE, INT, BIGINT etc.
DS2 equivalent for Hadoop

DROP/KEEP/RETAIN/RENAME are only valid in global scope. i.e. outside of a method programming block.
**DS2 equivalent for Hadoop**

```plaintext
proc ds2;
    thread compute / overwrite=yes;
    declare double jsup freq aver;
    retain jsup freq;
    keep i j freq aver;
    method run();
    set hdp.test;
    by i j;
    if first.j then do;
        jsup=0;
        freq=0;
        end;
    jsup + x;
    freq + 1;
    if last.j then do;
        aver=jsup/freq;
        output;
        end;
end;
endthread;
data hdp.jsubtot (overwrite=yes);
    declare thread compute t;
    method run();
    set from t;
    end;
enddata;
run;
quit;
```

Method `run()` is a system method – will execute in an implicit loop for every row of the input data. Other system methods are `init()` & `term()`
DS2 equivalent for Hadoop

```
proc ds2;
    thread compute / overwrite=yes;
    declare double jsu sub freq aver;
    retain jsu sub freq;
    keep i j freq aver;
    method run();
    set hdp.test;
    by i j;
    if first.j then do;
        jsu sub = 0;
        freq = 0;
        end;
        jsu sub + x;
        freq + 1;
    if last.j then do;
        aver = jsu sub / freq;
        output;
        end;
    endthread;
data hdp.jsubtot (overwrite=yes);
    declare thread compute t;
    method run();
    set from t;
    end;
    enddata;
run;
quit;
```

This block of code is identical to the original data step program.
DS2 equivalent for Hadoop

A BY statement is required to generate Hadoop REDUCE tasks. Without a BY statement, only MAP tasks are generated.
DS2 equivalent for Hadoop

```sas
proc ds2;
    thread compute / overwrite=yes;
    declare double jsub freq aver;
    retain jsub freq;
    keep i j freq aver;
    method run();
    set hdp.test;
    by i j;
    if first.j then do;
        jsub=0;
        freq=0;
    end;
    jsub + x;
    freq + 1;
    if last.j then do;
        aver=jsub/freq;
        output;
    end;
endthread;
data hdp.jsubtot (overwrite=yes);
    declare thread compute t;
    method run();
    set from t;
    end;
enddata;
run;
quit;
```

End statement to close the run() method.
DS2 equivalent for Hadoop

```
proc ds2;
  thread compute / overwrite=yes;
    declare double jsub freq aver;
    retain jsub freq;
    keep i j freq aver;
    method run();
    set hdp.test;
    by i j;
    if first.j then do;
      jsub=0;
      freq=0;
    end;
    jsub + x;
    freq + 1;
    if last.j then do;
      aver=jsub/freq;
    output;
    end;
  endthread;
  data hdp.jsubtot (overwrite=yes);
    declare thread compute t;
    declare thread compute t;
    method run();
    set from t;
    end;
    enddata;
run;
quit;
```

Endthread statement to close the thread program.
DS2 equivalent for Hadoop

```
proc ds2;
   thread compute / overwrite=yes;
      declare double jsub freq aver;
      retain jsub freq;
      keep i j freq aver;
      method run();
      set hdp.test;
      by i j;
      if first.j then do;
         jsub=0;
         freq=0;
      end;
      jsub + x;
      freq + 1;
      if last.j then do;
         aver=jsub/freq;
      output;
      end;
endthread;
data hdp.jsubtot (overwrite=yes);
   declare thread compute t;
   method run();
   set from t;
   end;
enddata;
run;
quit;
```

Now we reference the output dataset to be created on Hadoop
Explicitly declare the thread program and specify a name that identifies an instance of the thread.
DS2 equivalent for Hadoop

```
proc ds2;
  thread compute / overwrite=yes;
  declare double jsub freq aver;
  retain jsub freq;
  keep i j freq aver;
  method run();
  set hdp.test;
  by i j;
  if first.j then do;
    jsub=0;
    freq=0;
  end;
  jsub + x;
  freq + 1;
  if last.j then do;
    aver=jsub/freq;
    output;
  end;
endthread;

data hdp.jsubtot (overwrite=yes);
  declare thread compute t;
  method run();
  set from t;
  end;
enddata;
run;
quit;
```

Use method run() to allow the program to read from the thread program.
DS2 equivalent for Hadoop

```sas
proc ds2;
    thread compute / overwrite=yes;
    declare double jsub freq aver;
    retain jsub freq;
    keep i j freq aver;
    method run();
    set hdp.test;
    by i j;
    if first.j then do;
        jsub=0;
        freq=0;
    end;
    jsub + x;
    freq + 1;
    if last.j then do;
        aver=jsub/freq;
        output;
    end;
    endthread;
data hdp.jsubtot (overwrite=yes);
    declare thread compute t;
    method run();
    set from t;
    end;
enddata;
run;
quit;
```

Read the thread program by referencing the thread identifier
DS2 equivalent for Hadoop

```sas
proc ds2;
    thread compute / overwrite=yes;
    declare double jsub freq aver;
    retain jsub freq;
    keep i j freq aver;
    method run();
    set hdp.test;
    by i j;
    if first.j then do;
        jsub=0;
        freq=0;
    end;
    jsub + x;
    freq + 1;
    if last.j then do;
        aver=jsub/freq;
        output;
    end;
    endthread;
data hdp.jsubtot (overwrite=yes);
    declare thread compute t;
    method run();
    set from t;
    end;
    enddata;
run;
quit;
```

End statement to close the run() method.
The enddata statement marks the end of a data statement.
The RUN statement submits the DS2 statements.

DS2 equivalent for Hadoop

```sas
proc ds2;
    thread compute / overwrite=yes;
    declare double jsub freq aver;
    retain jsub freq;
    keep i j freq aver;
    method run();
    set hdp.test;
    by i j;
    if first.j then do;
        jsub=0;
        freq=0;
    end;
    jsub + x;
    freq + 1;
    if last.j then do;
        aver=jsub/freq;
        output;
    end;
endthread;
data hdp.jsubtot (overwrite=yes);
    declare thread compute t;
    method run();
    set from t;
    end;
enddata;
run;
quit;
```
DS2 equivalent for Hadoop

As DS2 is a SAS procedure we must explicitly quit it.
The SAS Log

83 proc ds2;
84 NOTE: Connection string:
85 NOTE: DRIVER=DS2;CONCPTS= (DRIVER=FEDSQL;CONCPTS= (DRIVER=base;CATALOG=WORK;SCHEMA=
86 (NAME=WORK;PRIMARYPATH=/tmp/SAS_work/485850004F88.ukval-01.suk.sas.com));
87 (DRIVER=FEDSQL;SERVER=grnhadoop-01.suk.sas.com;UID=sukm;PWD=); PORT=18080;SUBPROTOCOL=v1v2); ; DB_CONFIG=tcp/IP/SAS_work/485850004F88.ukval-01.suk.sas.com
88 (NAME=LODWORK;PRIMARYPATH=/home/sukm/WebWork)); (DRIVER=base;CATALOG=SASDATA;SCHEMA=
89 (NAME=SASDATA;PRIMARYPATH=/data/SAS/config/Level1/SASapp/Data)); (DRIVER=base;CATALOG=SASAPP;SCHEMA=
90 (NAME=STPASQP;PRIMARYPATH=/data/SAS/software/SASFoundation/9.4/samples/IntelTech)); (DRIVER=base;CATALOG=WALB;SCHEMA=
91 (NAME=WALB;PRIMARYPATH=/data/SAS/config/Level1/SASapp/Data/valib)); (DRIVER=base;CATALOG=WAPS;SCHEMA=
92 (NAME=WAPS;PRIMARYPATH=/data/SAS/software/SASFoundation/9.4/maps)); (DRIVER=base;CATALOG=WAPSQIPK;SCHEMA=
93 (NAME=WAPSQIPK;PRIMARYPATH=/data/SAS/software/SASFoundation/9.4/mapsqipk)); (DRIVER=base;CATALOG=SASUSER;SCHEMA=
94 (NAME=SASUSER;PRIMARYPATH=/home/sukm/ds2/sasuser.v94));)
95 thread compute / overwrite=yes;
96 declare double jsub freq aver;
97 retain j sub freq aver;
98 keep i j freq aver;
99 method run();
100 set hop.test;
101 by i j;
102 if first.j then do;
103 jsub=0;
104 freq=0;
105 end;
106 jsub + x;
107 freq + 1;
108 if last.j then do;
109 aver=jsub/freq;
110 output;
111 end;
112 end;
113 endthread;
114 data hdo.jsubtot (overwrite=yes);
115 declare thread compute t;
116 method run();
117 set from t;
118 end;
119 enddata;
120 run;
121 NOTE: Created thread compute in data set work.compute.
122 NOTE: Running THREAD program in-database
123 NOTE: Running DATA program in-database
124 NOTE: Execution succeeded. No rows affected.
125 quit;
126 NOTE: PROCEDURE DS2 used (Total process time):
What’s happening on the Hadoop cluster?

<table>
<thead>
<tr>
<th>Logs</th>
<th>ID</th>
<th>Name</th>
<th>Status</th>
<th>User</th>
<th>Maps</th>
<th>Reduces</th>
<th>Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>📚</td>
<td>1431102899342_0270</td>
<td>SAS Map/Reduce Job</td>
<td>RUNNING</td>
<td>sukdmg</td>
<td>50%</td>
<td>50%</td>
<td>root.sukdmg</td>
</tr>
</tbody>
</table>
Running data step in Hadoop with 9.4 M2

- Only one input and output data set and both must be on Hadoop
- Only a subset of the full DATA step syntax is currently available for parallel execution.
  - Data step logic is converted to DS2 under the covers
- Only functions and formats that are supported by the DS2 language compile successfully.
  - E.g. No LAG or DIF functions
- SAS Statements not currently supported:
  - BY (or FIRST. and LAST. variables)
  - CONTINUE
  - DISPLAY
  - FILE
  - INFILE
  - INPUT
  - LEAVE
  - MERGE
  - MODIFY
  - OUTPUT
  - PUT
  - REMOVE
  - RENAME
  - REPLACE
  - RETAIN
  - UPDATE
  - WHERE
  - WINDOW
An important option!

• For any Data Step program to run in Hadoop the following SAS option must be set:

```
15 16  options DSACCEL=ANY;
17
```
Data Step Example log

```sas
68  /*Turn on SAS DS1 processing via the SAS EP*/
69  options msglevel=1;
70  options DSACCEL=ANY;
71
72  proc delete data=hdfsdemo.scored_big;
73  run;

NOTE: Deleting HDFSDEMO.SCORED_BIG (memtype=DATA).
NOTE: PROCEDURE DELETE used (Total process time):
    real time     0.18 seconds
    cpu time     0.11 seconds

75
76  data hdfsdemo.scored_big;
77  set hdfsdemo.intra;
79  /* Execute the score code. */
80  if sum > 1000 then score=1;
81  run;

NOTE: Attempting to run DATA Step in Hadoop.
NOTE: Data Step code for the data set “HDFSDEMO.SCORED_BIG” was executed in the Hadoop EP environment.

Hadoop Job (HDP_JOB_ID), job_1431102899342_0220, SAS Map/Reduce Job,
http://gbrhadoop1-01.suk.sas.com:8088/proxy/application_1431102899342_0220/

Hadoop VersionUser
real time  1:27.45
2.5.0-cdh5.3.1demo
cpu time  0.32 seconds

Started At
May 22, 2015 10:42:23 AM

Finished At
May 22, 2015 10:42:49 AM
```
What’s coming for DS2 for Hadoop with 9.4 M3

• DS2 SET / MERGE
  • Multi-table SET: set a b c;
  • SQL SET: set {select * from A inner join B on A.id = B.id};
  • MERGE: merge A B C; by X;
  • Support for IN=, FIRST. and LAST.
Next steps with DS2

Thank You