

Debugging and Monitoring On-line Validation and Derivation Procedures in OC and RDC



Bill Caulkins

DBMS Consulting

12 October 2010

Validation/Derivation Procedure Focus Group

Session S16



Acknowledgements

- Many thanks to OHSUG for this opportunity to present to the Validation and Derivation Procedures Focus Group.
- Many thanks to the Validation and Derivation Procedures Focus Group Chairs for their receiving and expeditious review of this presentation
- Many thanks to everyone who participated in the development of presentation.



Debugging PL/SQL Code

DBMS_OUTPUT

- Every developer, at one time or another, needs to debug code
- PL/SQL offers DBMS_OUTPUT.PUT_LINE.
 - Messages can be up to 255 characters long;
 - Messages are buffered in user memory
 - No history of messages
 - Messages are not displayed until the current statement completes
 - Can be problematic for long-running jobs
 - All message are created "equal"
 - A severe error message is treated the same as an "I am here" message
- DBMS_OUTPUT can be used to display messages in OC Validation and Derivation procedures



Debugging PL/SQL Code

OPA_TRACE

- Oracle has developed the OPA_TRACE package which can:
 - Be enabled or disabled
 - Write messages to the screen
 - Write messages to a table (OPA_DEBUG)
 - Write messages depending on severity of the message
 - Write messages to a file

- Messages written to a table can be viewed (even from another session) the instant they are inserted
 - Autonomous transactions occur within the OPA_TRACE package
 - COMMITs in this package are separate from COMMITs in the session which is invoking the package
 - No waiting for a process to finish before reviewing some of the results

- OPA_TRACE can be used instead of DBMS_OUTPUT

- By utilizing OPA_TRACE in error handling routines, procedures, and functions and directing the output to an Oracle table, developers and support personnel can identify errors across the user community, and uncover errors which may have gone unnoticed



Agenda

- Simple Example of OPA_TRACE in PL/SQL
- Review opa_debug table
- Review OPA_TRACE's Procedures and Functions
- OPA_TRACE in a Derivation Procedure
- Getting Started with OPA_TRACE



Example using OPA_TRACE in PL/SQL Code

BEGIN

```
opa_trace.debugon;  
opa_trace.tableon;  
<some pl/sql code>  
opa_trace.debugmsg(msgstring);  
<some pl/sql code>
```

Your PL/SQL code with
some debug statements

EXCEPTION

WHEN OTHERS THEN

```
opa_trace.debugmsg(sqlerrm, proc name)
```

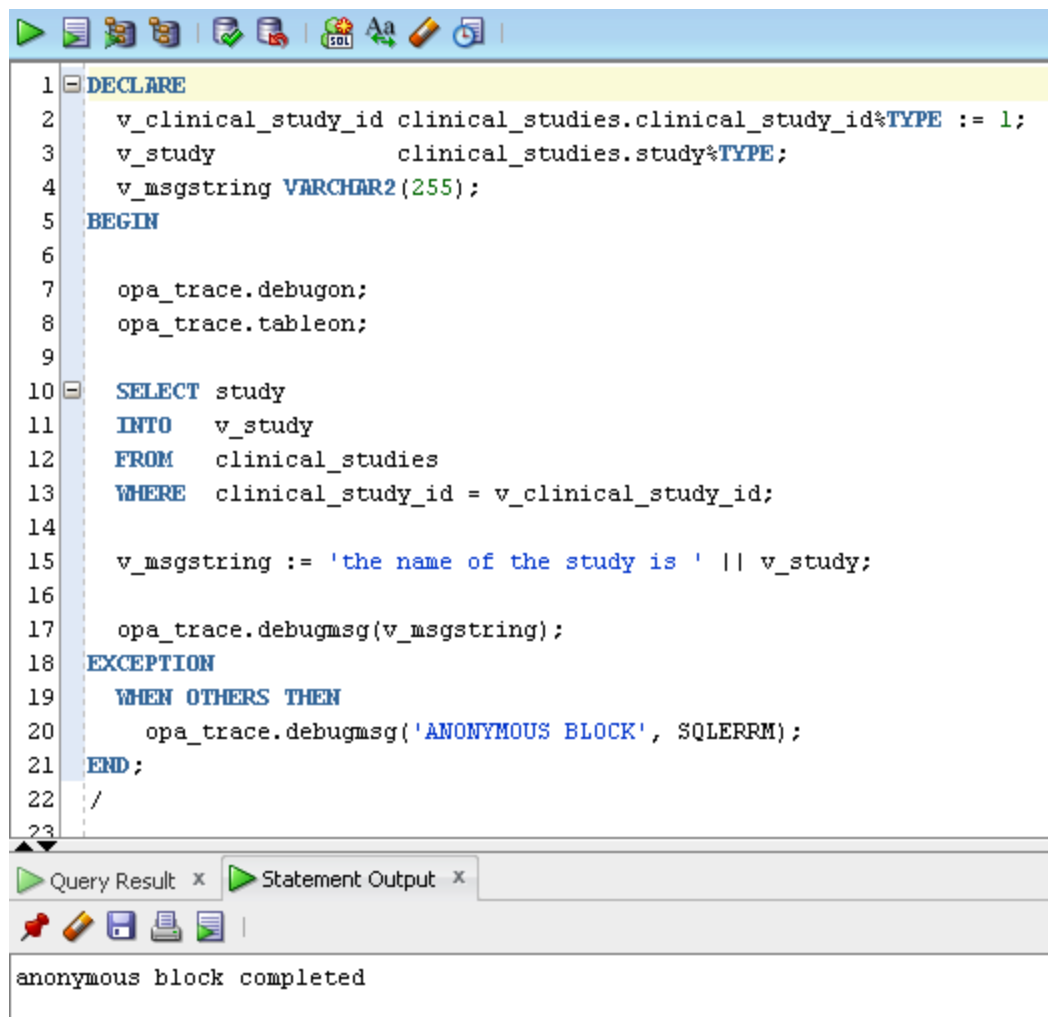
END;

View the messages
created by OPA_TRACE

```
SELECT * FROM opa_debug WHERE sessionid = USERENV('SESSIONID');
```



Example using OPA_TRACE in PL/SQL Code



```
1 DECLARE
2   v_clinical_study_id clinical_studies.clinical_study_id%TYPE := 1;
3   v_study             clinical_studies.study%TYPE;
4   v_msgstring VARCHAR2(255);
5 BEGIN
6
7   opa_trace.debugon;
8   opa_trace.tableon;
9
10 SELECT study
11 INTO   v_study
12 FROM   clinical_studies
13 WHERE  clinical_study_id = v_clinical_study_id;
14
15 v_msgstring := 'the name of the study is ' || v_study;
16
17 opa_trace.debugmsg(v_msgstring);
18 EXCEPTION
19 WHEN OTHERS THEN
20   opa_trace.debugmsg('ANONYMOUS BLOCK', SQLERRM);
21 END;
22 /
23
```

Query Result x Statement Output x

```
anonymous block completed
```



Example using OPA_TRACE in PL/SQL Code

The screenshot shows the Oracle SQL Developer interface. The top toolbar contains various icons for file operations, execution, and editing. The main window displays a SQL query:

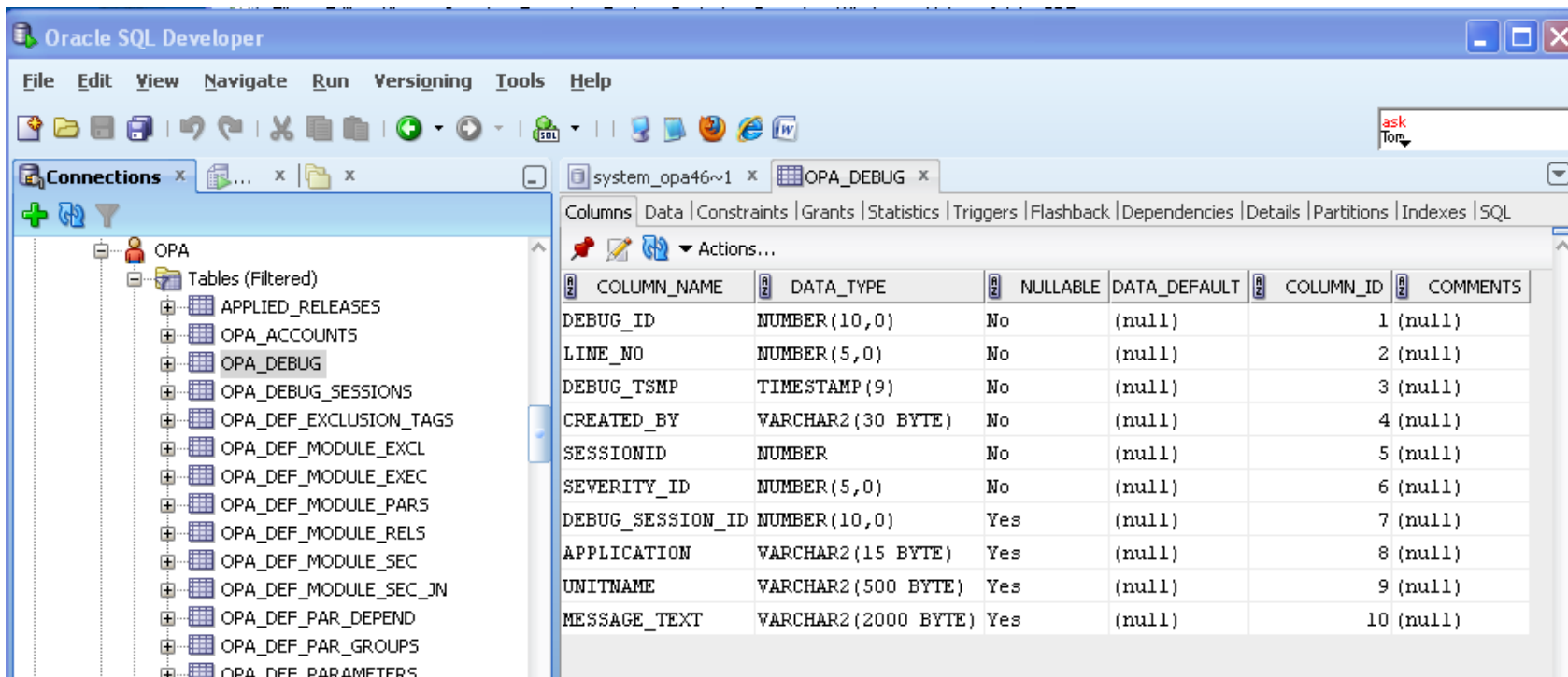
```
24 SELECT *  
25 FROM opa_debug  
26 WHERE sessionid = USERENV('sessionid')
```

Below the query editor, the "Query Result" tab is active, showing the execution status: "All Rows Fetched: 1 in 0.047 seconds". The results are displayed in a table with the following columns and data:

DEBUG_ID	LINE_NO	DEBUG_TSMP	CREATED_BY	SESSIONID	SEVERITY_ID	DEBUG...	APPLICATION	UNITNAME	MESSAGE_TEXT
1	817	0 20-SEP-10...	TMS_LOAD	16630394	820	(null)	OCL	ANONYMOUS BLOCK	ORA-01403: no data found



OPA_DEBUG table



The screenshot shows the Oracle SQL Developer interface. The left pane displays the 'Connections' tree with the 'OPA' schema expanded, showing a list of tables including 'OPA_DEBUG'. The right pane shows the 'Columns' tab for the 'OPA_DEBUG' table, displaying the following structure:

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
DEBUG_ID	NUMBER(10,0)	No	(null)	1 (null)	
LINE_NO	NUMBER(5,0)	No	(null)	2 (null)	
DEBUG_TSMP	TIMESTAMP(9)	No	(null)	3 (null)	
CREATED_BY	VARCHAR2(30 BYTE)	No	(null)	4 (null)	
SESSIONID	NUMBER	No	(null)	5 (null)	
SEVERITY_ID	NUMBER(5,0)	No	(null)	6 (null)	
DEBUG_SESSION_ID	NUMBER(10,0)	Yes	(null)	7 (null)	
APPLICATION	VARCHAR2(15 BYTE)	Yes	(null)	8 (null)	
UNITNAME	VARCHAR2(500 BYTE)	Yes	(null)	9 (null)	
MESSAGE_TEXT	VARCHAR2(2000 BYTE)	Yes	(null)	10 (null)	



OPA_DEBUG table

Oracle SQL Developer

File Edit View Navigate Run Versioning Tools Help

Connections x ... x | system_opa46~1 x | OPA_DEBUG x

Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

Note especially the columns

SEVERITY_ID

UNITNAME

MESSAGE_TEXT

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
DEBUG_ID	NUMBER(10,0)	No	(null)	1 (null)	
LINE_NO	NUMBER(5,0)	No	(null)	2 (null)	
DEBUG_SESSIONID	TIMESTAMP(9)	No	(null)	3 (null)	
CREATED_BY	VARCHAR2(30 BYTE)	No	(null)	4 (null)	
SESSIONID	NUMBER	No	(null)	5 (null)	
SEVERITY_ID	NUMBER(5,0)	No	(null)	6 (null)	
DEBUG_SESSION_ID	NUMBER(10,0)	Yes	(null)	7 (null)	
APPLICATION	VARCHAR2(15 BYTE)	Yes	(null)	8 (null)	
UNITNAME	VARCHAR2(500 BYTE)	Yes	(null)	9 (null)	
MESSAGE_TEXT	VARCHAR2(2000 BYTE)	Yes	(null)	10 (null)	

OPA

Tables (Filtered)

- OPA_ACCOUNTS
- OPA_DEBUG
- OPA_DEF_EXCLUSION_TAGS
- OPA_DEF_MODULE_EXCL
- OPA_DEF_MODULE_EXEC
- OPA_DEF_MODULE_PARS
- OPA_DEF_MODULE_RELS
- OPA_DEF_MODULE_SEC
- OPA_DEF_MODULE_SEC_JN
- OPA_DEF_PAR_DEPEND
- OPA_DEF_PAR_GROUPS
- OPA_DEF_PARAMETERS



OPA_TRACE PACKAGE

- Schema OPA owns package OPA_TRACE
- CONSTANTS
- PROCEDURES/FUNCTIONS
 - Messages
 - Message Destination
 - Others



OPA_TRACE CONSTANTS

- The “debug level” constants provide a level of granularity to differentiate the severity levels of messages
 - ALL_MESSAGES – 0
 - FINEST – 300
 - FINER – 400
 - FINE – 500
 - CONFIG – 700
 - INFO – 800
 - DEBUG_LOW – 820
 - DEBUG_MEDIUM – 850
 - DEBUG_HIGH – 870
 - WARNING – 900
 - SEVERE – 1000

- These values can be placed in, and queried from, the OPA_DEBUG table
 - `SELECT * FROM opa_debug where severity_id = <some value>;`



OPA_TRACE Messages

■ Procedure DebugMsg Overloaded versions

- pMessage
- pMessage, pSeverity
- pUnitName, pMessage
- pUnitName, pMessage, pSeverity
- pApplication, pUnitName, pMessage
- pApplication, pUnitName, pMessage, pSeverity

■ Parameters

- pMessage – content of message
- pSeverity – relative severity – defaults to DEBUG_LOW
- pUnitName – name of program/procedure
- pApplication – name of application



OPA_TRACE Message Destination

Debug Messages are written to either:

- **File**
 - PROCEDURE FileOn
 - pDirectory, pFilename
 - PROCEDURE FileOff
 - FUNCTION Filing (returns Boolean)

- **Table (opa_debug)**
 - PROCEDURE TableOn
 - PROCEDURE TableOff
 - FUNCTION Tabling

- **Screen (set serveroutput on)**



OPA_TRACE Other Parameters

■ Procedure DebugOn

- No parameters
- pBufferSize (optional parameter)

```
BEGIN
    opa_trace.debugon(1000000);
    REM more code...
END;
```

■ Procedure DebugOff

■ Procedure SetSeverity

- pSeverity
 - Sets severity – all messages that have a severity \geq this value will be logged

■ FUNCTION getSeverity



OPA_TRACE – Other Parameters

- **PROCEDURE setMultiLineOn**
 - No argument
 - pLineLength
- **PROCEDURE setMultiLineOff (default)**
- **FUNCTION Debugging (returns boolean)**
- **PROCEDURE registerDebugSession**
 - pAppSessionId
 - Will call tableOn and setMultiLineOn
- **PROCEDURE PurgeDebugTable**
 - No arguments
 - pForUser
 - pForSession
 - pBeforeDate
- **PROCEDURE setDebugErrorMsgOn**
- **PROCEDURE setDebugErrorMsgOff**
- **FUNCTION getDebugErrorMsg**



OPA_TRACE in a Derivation Procedure

- Example: Debug being turned on in a post-details procedure

The screenshot displays a software interface with a menu bar (Action, Edit, Move, Clear, Data, Query, Special, Help, Window) and a toolbar. A Navigator pane on the left shows a tree structure under 'OC OCL 4.6.0' with categories like Admin, Plan, Design, Glib, Definition, DCMs, DCIs, Validation Procs, Derivation Procs, Data Extract View, Copy Groups, Test a Study, Test Data Entry, Test Mass Change, SA for Study, and Definition Reports. The 'Derivation Procs' category is expanded, showing sub-items: Procedures, Prov Proce, Procedures, and Qry Proce.

A dialog box titled 'Maintain Study Derivation Procedure (Study: AS_STUDY1)' is open, showing 'Custom Code for Procedure D_AE'. The 'Custom Code Location' dropdown is set to '* POST-DETAILS'. The code area contains the following text:

```
opa.opa_trace.debugon;  
opa.opa_trace.tableon;  
opa.opa_trace.debugmsg("Starting Post-Details. Patient Position Id is: ' ||  
    rxcpdstd.patients_rec.patient_position_id);
```

Buttons for 'Back', 'Save', and 'OK' are visible at the bottom of the dialog.

A 'Forms' window is also open, displaying a message: '79940: Completed with SUCCESS status - Batch job Id 1604'. It includes a yellow sticky note icon and an 'OK' button.



OPA_TRACE in a Derivation Procedure

- Now let's execute the procedure in test mode

The screenshot shows a software application window titled 'Maintain Study Derivation Procedure (Study: AS_STUDY1)'. The window contains a table with the following data:

Procedure Name	Domain	Version	Status	Order	Exec Context	Des
D_AE	AS_STUDY1	0	P	100	OFF-LINE	Der

Overlaid on top of the table is a smaller dialog box titled 'Forms'. It contains a red question mark icon and the text: 'Ready to execute against TEST data. Run in debug mode?'. Below the text are three buttons: 'Yes', 'No', and 'Cancel'.



OPA_TRACE in a Derivation Procedure

- Messages in opa_debug table

```
4 select * from opa_debug where debug_tmstp > sysdate - 1
```

Query Result x

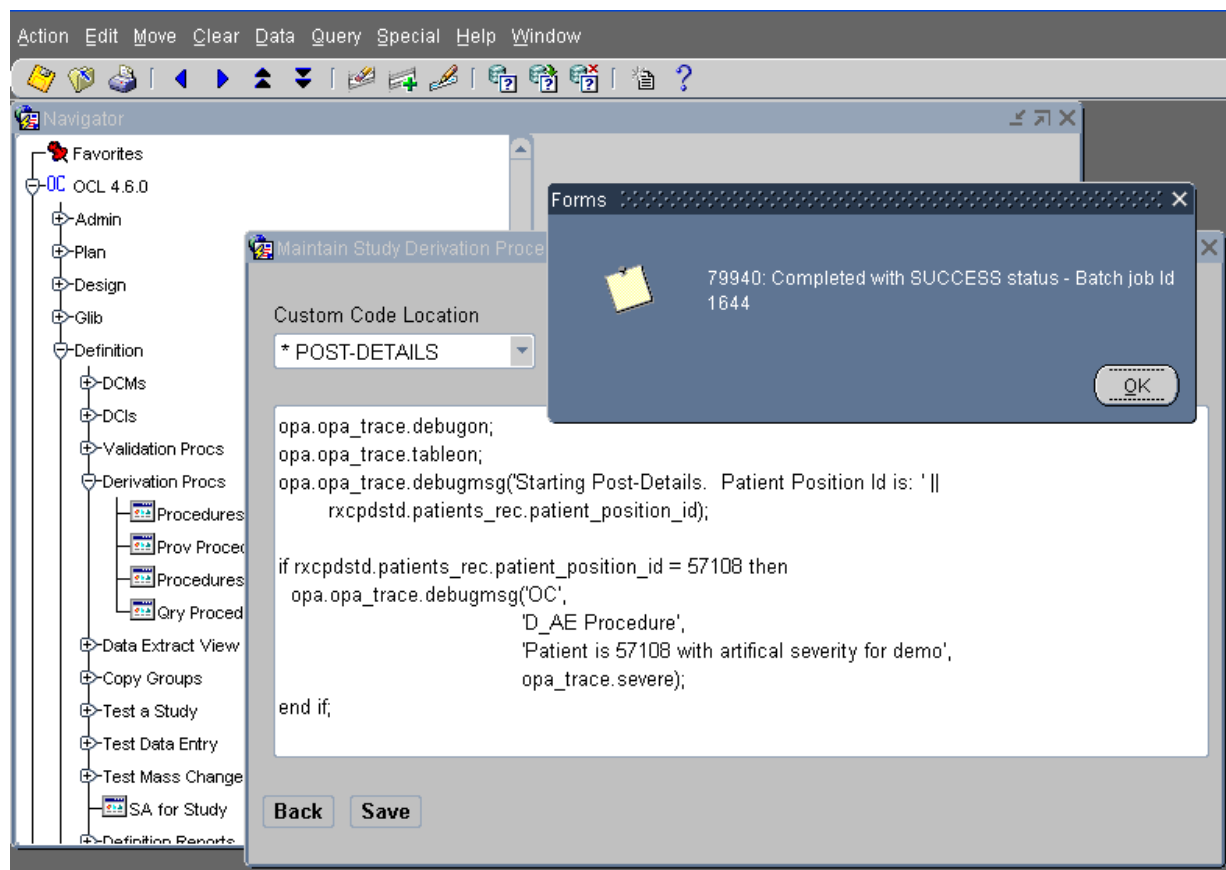
SQL | All Rows Fetched: 8 in 0.047 seconds

EBUG_ID	LIN...	DEBUG_TMSP	CREATE...	SESSIONID	SEVE...	DEBUG_SESSION_ID	APP...	UNI...	MESSAGE_TEXT
1	843	0 23-SEP-10 10.39.44...	OPS\$OPAPPS	18620394	820	(null)	OCL	(null)	Starting Post-Details. Patient Position Id is: 57108
2	844	0 23-SEP-10 10.39.44...	OPS\$OPAPPS	18620394	820	(null)	OCL	(null)	* Mark_Pat.APPLY 1 *
3	845	0 23-SEP-10 10.39.44...	OPS\$OPAPPS	18620394	820	(null)	OCL	(null)	* In Mark_Pat.APPLY, calling ocl_flex_assessment.proces
4	846	0 23-SEP-10 10.39.44...	OPS\$OPAPPS	18620394	820	(null)	OCL	(null)	* Mark_Pat.APPLY 2 *
5	847	0 23-SEP-10 10.39.44...	OPS\$OPAPPS	18620394	820	(null)	OCL	(null)	Starting Post-Details. Patient Position Id is: 57208
6	848	0 23-SEP-10 10.39.44...	OPS\$OPAPPS	18620394	820	(null)	OCL	(null)	* Mark_Pat.APPLY 1 *
7	849	0 23-SEP-10 10.39.44...	OPS\$OPAPPS	18620394	820	(null)	OCL	(null)	* In Mark_Pat.APPLY, calling ocl_flex_assessment.proces
8	850	0 23-SEP-10 10.39.44...	OPS\$OPAPPS	18620394	820	(null)	OCL	(null)	* Mark_Pat.APPLY 2 *



OPA_TRACE in a Derivation Procedure

- Demonstrate a severe message



The screenshot displays the OHSUG software interface. The main window shows a tree view on the left with the following structure:

- OC OCL 4.6.0
 - Admin
 - Plan
 - Design
 - Glib
 - Definition
 - DCMs
 - DCIs
 - Validation Procs
 - Derivation Procs
 - Procedures
 - Prov Proce
 - Procedures
 - Gry Proce
 - Data Extract View
 - Copy Groups
 - Test a Study
 - Test Data Entry
 - Test Mass Change
 - SA for Study
 - Definition Reports

The main window displays the following code:

```
opa.opa_trace.debugon;  
opa.opa_trace.tableon;  
opa.opa_trace.debugmsg("Starting Post-Details. Patient Position Id is: '||  
    rxcpdstd.patients_rec.patient_position_id);  
  
if rxcpdstd.patients_rec.patient_position_id = 57108 then  
    opa.opa_trace.debugmsg("OC",  
        'D_AE Procedure',  
        'Patient is 57108 with artificial severity for demo',  
        opa_trace.severe);  
end if;
```

A dialog box titled "Forms" is open, displaying the following message:

```
79940: Completed with SUCCESS status - Batch job Id  
1644
```

The dialog box has an "OK" button.



OPA_TRACE in a Derivation Procedure

Messages in opa_debug table

Note the Severity 1000

```
4 | select * from opa_debug where debug_tsmp > sysdate - 1
```

Query Result x

All Rows Fetched: 9 in 0.047 seconds

	DEBUG_ID	...	DEBUG_TSMP	CREATED_BY	SESSIONID	SEVERI...	DEBU...	APPLI...	UNITNAME	MESSAGE_TEXT
1	851	0	23-SEP-10 11.37...	OPS\$OPAPPS	18630404	820	(null) OCL	(null)	(null)	Starting Post-Details. Patient Position Id is: 57108
2	852	0	23-SEP-10 11.37...	OPS\$OPAPPS	18630404	1000	(null) OC	D_AE Procedure	(null)	Patient is 57108 with artificial severity for demo
3	853	0	23-SEP-10 11.37...	OPS\$OPAPPS	18630404	820	(null) OCL	(null)	(null)	* Mark_Pat.APPLY 1 *
4	854	0	23-SEP-10 11.37...	OPS\$OPAPPS	18630404	820	(null) OCL	(null)	(null)	* In Mark_Pat.APPLY, calling ocl_flex_assessment.processl
5	855	0	23-SEP-10 11.37...	OPS\$OPAPPS	18630404	820	(null) OCL	(null)	(null)	* Mark_Pat.APPLY 2 *
6	856	0	23-SEP-10 11.37...	OPS\$OPAPPS	18630404	820	(null) OCL	(null)	(null)	Starting Post-Details. Patient Position Id is: 57208
7	857	0	23-SEP-10 11.37...	OPS\$OPAPPS	18630404	820	(null) OCL	(null)	(null)	* Mark_Pat.APPLY 1 *
8	858	0	23-SEP-10 11.37...	OPS\$OPAPPS	18630404	820	(null) OCL	(null)	(null)	* In Mark_Pat.APPLY, calling ocl_flex_assessment.processl
9	859	0	23-SEP-10 11.37...	OPS\$OPAPPS	18630404	820	(null) OCL	(null)	(null)	* Mark_Pat.APPLY 2 *

OPA_TRACE in a Derivation Procedure

One last example

Set the severity level to 'warning'

The screenshot displays the OHSUG software interface. On the left, a Navigator pane shows a tree structure under 'OC OCL 4.6.0', with 'Derivation Procs' expanded to show 'D_AE'. A dialog box titled 'Maintain Study Derivation Procedure (Study: AS_STUDY1)' is open, showing the 'Custom Code for Procedure D_AE' configuration. The 'Custom Code Location' is set to '* POST-DETAILS'. The custom code contains the following OHSUG script:

```

opa.opa_trace.debugon;
opa.opa_trace.tableon;
opa.opa_trace.setseverity(opa_trace.warning);

opa.opa_trace.debugmsg("Starting Post-Details. Patient Position Id is: " ||
  rxcpstd.patients_rec.patient_position_id);

if rxcpstd.patients_rec.patient_position_id = 57108 then
  opa.opa_trace.debugmsg("OC',
    'D_AE Procedure',
    'Patient is 57108 with artificial severity for demo',
    opa_trace.severe);
  
```

At the bottom of the dialog box, there are 'Back' and 'Save' buttons.



OPA_TRACE in a Derivation Procedure

Messages in opa_debug table
After severity was set to "warning"

```
4 select * from opa_debug where debug_tsmp > sysdate - 1
```

Query Result x

All Rows Fetched: 1 in 0.062 seconds

DEBUG_ID	LINE_NO	DEBUG_TSMP	CREATED_BY	SESSIONID	SEVERITY_ID	DEBUG_SESSION_ID	APPLICATION	UNITNAME	MESSAGE_TEXT
1	860	0 23-SEP-10 11.48.11.317000000 PM	OPS\$OPAPPS	18630421	1000	(null)	OC	D_AE Procedure	Patient is 57108 with ar

The other messages are not written to the opa_debug table



Getting Started with OPA_TRACE

- Start now in development. Replace DBMS_OUTPUT.PUT_LINE with OPA_TRACE.DEBUGMSG.
 - Benefit immediately by removing the 255 character limit
 - Ability to write records to tables for viewing during processing (from a different session).
- Develop standards for “WHEN OTHERS” clauses
 - Record any error raised by any user in a table.
 - Daily review of the table may help identify problems you didn't know that you had



Getting Started with OPA_TRACE

- **Establish standards regarding OPA_TRACE**
 - Define severity levels
 - Include package and procedure name for pUnitName argument
 - Include data values in error messages (so that problems can be replicated)
- **Create your own PL/SQL “wrapper” package which can**
 - Enforce standards regarding messages and debugging
 - Ensure severe errors are captured
 - Protect you from changes to OPA_TRACE
- **Advanced Ideas**
 - Provide the ability to turn on debug for a given user or session
 - Provide the ability to “watch” for certain procedures and only turn on debug for them



Conclusions

- OPA_TRACE provides the functionality of DBMS_OUTPUT.
- OPA_TRACE ALSO provides the ability to write messages to an Oracle table and get instantaneous information regarding long-running processes
- Debugging in development is useful – but why stop there?
- Potential to leverage debugging in a production environment to assist users in troubleshooting specific problems without capturing “every thing from every user.”
- Potential to leverage debugging in a production environment to record error messages from custom code for any user or any process



Question and Answers

Sunil G. Singh

singh@clinicalserver.com

+1-860-983-5848

Bill Caulkins

bill.caulkins@clinicalserver.com

+1-646-963-6101 x 707



Biographies

Sunil G. Singh, President & CEO, DBMS Consulting, Inc.

- Sunil is a Global Oracle Health Sciences deployment expert for DBMS Consulting. He has been an active member of the OCUG community since 1996 and is extremely grateful for this opportunity to make these presentations at OCUG 2010.

Bill Caulkins, Director, DBMS Consulting, Inc.

- Bill has over 20 years experience working with the Oracle RDBMS and has worked in the pharmaceutical industry since the early 1990s as an Oracle developer, technical team leader, and project manager.