Smart Triggers/Push Data

Nagaraju & Brian

Value Proposition

- Selectively trigger events based on changes in server data
- Real time ‘push’ notifications help clients avoid polling the server
- Small data flow allows simple small clients to work with many triggered events at once

What are Smart Triggers in JDBC

- Smart Triggers are registered events on the server that you subscribe to from your JDBC client
  - Triggers are based on a SQL statement query that matches changes made to a table
  - SELECT id FROM CUSTOMER WHERE cardBalance > 20000;
- One client can listen to many events from many tables, allowing a wide range of monitoring opportunities
  - Monitor account balances
  - Take action on suspicious behaviors

What does a Smart Trigger Look Like?

- It's designed to be a simple set of classes/interfaces in Java
- Backed by a connection to the sysadmin database
  - Must connect to sysadmin database
- Designed for both simple standalone monitor applications as well as integration into multi-threaded environments
- Leverages the Push Notification feature in the server to do the heavy lifting
- Receives JSON documents when a trigger occurs

Use case: Banking

- Bank accounts
  - I want to be alerted when an account balance drops below zero dollars
  - I don’t want to write SPL or install stored procedures
  - I want to be notified in my client application
  - I don’t want to poll the database for this information or re-query each time a balance changes from the client

Smart Trigger Bank Code

```
public class BankMonitor implements IfmxSmartTriggerCallback {
      public static void main(String[] args) throws SQLException {
            IfxSmartTrigger trigger = new IfxSmartTrigger(args[0]); // pass in JDBC URL to SYSADMIN database
            trigger.timeout(5).label("bank_alert");
            trigger.addTrigger("account", "informix", "bank",
                  "SELECT * FROM account WHERE balance < 0", new BankMonitor());
            trigger.watch(); //blocking call
      }

      @Override
      public void notify(String json) {
            System.out.println("Bank Account Ping!");
            if (json.contains("ifx_isTimeout")) {
                  System.out.println("-- No balance issues");
            }
            else {
                  System.out.println("-- Bank Account Alert detected!");
                  System.out.println("   " + json);
            }
      }
}
```

Demo!

Smart Triggers in Other Languages

- Adding Smart Triggers to the JDBC driver allows other languages to have this support
- Groovy, JavaScript (NodeJS)*, Python*, Scala and more
  - * Using Java bridges these languages can call the JDBC driver in Java
- Native Smart Trigger API's for other drivers under consideration

NodeJS Smart Trigger Example

```
var java = require("java");
java.asyncOptions = {
  syncSuffix: ""
};

java.classpath.push("ifxjdbc.jar");

var smartTrigger = java.newInstanceSync("com.informix.smartTrigger.IfxSmartTrigger",
"jdbc:informix-sqli://localhost:20290/sysadmin:USER=informix;PASSWORD=informix");

smartTrigger.timeout(10);

smartTrigger.open();
smartTrigger.addTrigger("pushtest", "informix", "ewdb", "SELECT * FROM pushtest", "smart-trigger");
smartTrigger.registerTriggers();

var foo = smartTrigger.readTriggerEvent();
console.log(foo);
```

JSON attributes for registering new event conditions

Input attribute name	Description
table	Table name to be registered
owner	Table owner
database	Database name
query	Select statement including projection list and where clause to register for changes in a data set.
label	User defined string to be returned along with event document – useful to differentiate between events when more than one push-data event registered within the the same session
timeout	How long client gets blocked in smartblob read api for new events to be returned by server before returning timeout document.
commit_time	Return event data committed after this transaction commit time.
txnid	8 byte unique id. Higher order 4 bytes: commit work log id, lower order 4 bytes: commit work log position.
max_pending_ops	Maximum number of event records to be kept in the session pending
maxrecs	Maximum number of records to be returned by smartblob api read call.

Example Command:

execute function informix.task('pushdata register', {table:“creditTable",owner:"informix",database:"creditdb",query:"select uid, cardid, carddata from creditTable where carddata.Amount >= 100",label:"card txn alert"})

Event Data JSON Attributes:

Attribute name	Description
operation	Operation type: Insert/Delete/Update
table	Table name
owner	Table owner
database	Database name
label	Optional user specified data for the event condition.
txnid	8 byte unique id. Higher order 4 bytes: commit work log id, lower order 4 bytes: commit work log position.
commit_time	Transaction commit time for the event data.
op_num	Increasing sequence number for the event document within a given transaction. If transaction generate 10 events, then each document returned will have incrementing op_num starting from 1 to 10.
rowdata	Row data in JSON document format. Data is returned in column name as key and column data as value.
before_rowdata	Before row data for “update” operation.
ifx_isTimeout	Document with this attribute is returned with value set to “true” if no events gets triggered within the timeout interval specified by the client.
ifx_warn_total_skipcount	Warning document with this attribute is returned with cumulative number of discarded events due to max_pending_ops attribute threshold.
operation_owner_id	Operation’s session owner userid. Added in 12.10xC10.
operation_session_id	Operation’s session id. Added in 12.10xC10.

Example event data documents

- Sample output for Insert operation:

  {“operation”:"insert",”table”:"creditcardtxns",”owner”:"informix",”database”:"creditdb",”label”:"card txn alert",”txnid”:2250573177224,"operation_owner_id":200,"operation_session_id":5,”commit_time”:1488243530,”op_num”:1,”rowdata”:{“uid”:22,”cardid”:"6666-6666-6666-6666",”carddata”:{"Merchant":"Sams Club","Amount":200,"Date":2017-05-01T10:35:10.000Z } }}

- Sample output for Update operation:

  {“opertion”:"update",table:"creditcardtxns",”owner”:"informix",”database”:"creditdb",”label”:"card txn alert",”txnid”:2250573308360, "operation_owner_id":200,"operation_session_id":5,”commit_time”:1488243832,”op_num”:1,”rowdata”:{uid:21,cardid:"7777-7777-7777-7777",”carddata”:{"Merchant":"Sams Club","Amount":200,"Date":"25-Jan-2017 16:15"} },”before_rowdata”:{“uid”:21,”cardid”:"6666-6666-6666-6666",”carddata”:{"Merchant":"Sams Club","Amount":200,"Date":2017-05-01T10:35:10.000Z } }}

- Sample output for Delete operation:

  {“opertion”:"delete",”table”:"creditcardtxns",”owner”:"informix",”database”:"creditdb",”label”:"card txn alert",”txnid”:2250573287760, "operation_owner_id":200,"operation_session_id":5,”commit_time”:1488243797,”op_num”:1,”rowdata”:{“uid”:22,”cardid”:"6666-6666-6666-6666",”carddata”:{"Merchant":"Sams Club","Amount":200,"Date":2017-05-01T13:35:06.000Z } }}

- Sample output for multi row document when maxrecs input attribute set to greater than 1:

  {[

  {“operation”:"Insert",”table”:"creditcardtxns",”owner”:"informix",”database”:"creditdb",”label”:"card txn alert", “txnid”:2250573309999, "operation_owner_id":200,"operation_session_id":5,”commit_time”:1487781325,”op_num”:1,”rowdata”:{uid:"7",”cardid”:"6666-6666-6666-6666",”carddata”:{"Merchant":"Sams Club","Amount":200,"Date":2017-05-01T15:10:10.000Z } }},

  {“operation”:"insert",table:"creditcardtxns",”owner”:"informix",”database”:"creditdb",”label”:"card txn alert",”txnid”:2250573177224, "operation_owner_id":200,"operation_session_id":5,”commit_time”:1488243530,”op_num”:1,”rowdata”:{“uid”:22,”cardid”:"6666-6666-6666-6666",”carddata”:{"Merchant":"Sams Club","Amount":200,"Date":2017-05-01T16:20:10.000Z } }}

  ]}

Detached Triggers

- Added in 12.10xC10/ 4.10.JC10
- Allows triggers to be detachable
  - Connection can be interrupted
  - Can willfully disconnect the trigger
- Detached triggers continue to process events on the server
  - Events are stored in memory buffers until you reconnect your session
  - Size of memory buffer can be configured
  - Detached triggers can be terminated by a DBA
    - *execute function task('pushdata delete', '{session_id:"12"}') ;*
    - *execute function task('pushdata delete', '{delete_all:"1"}');*

API Example

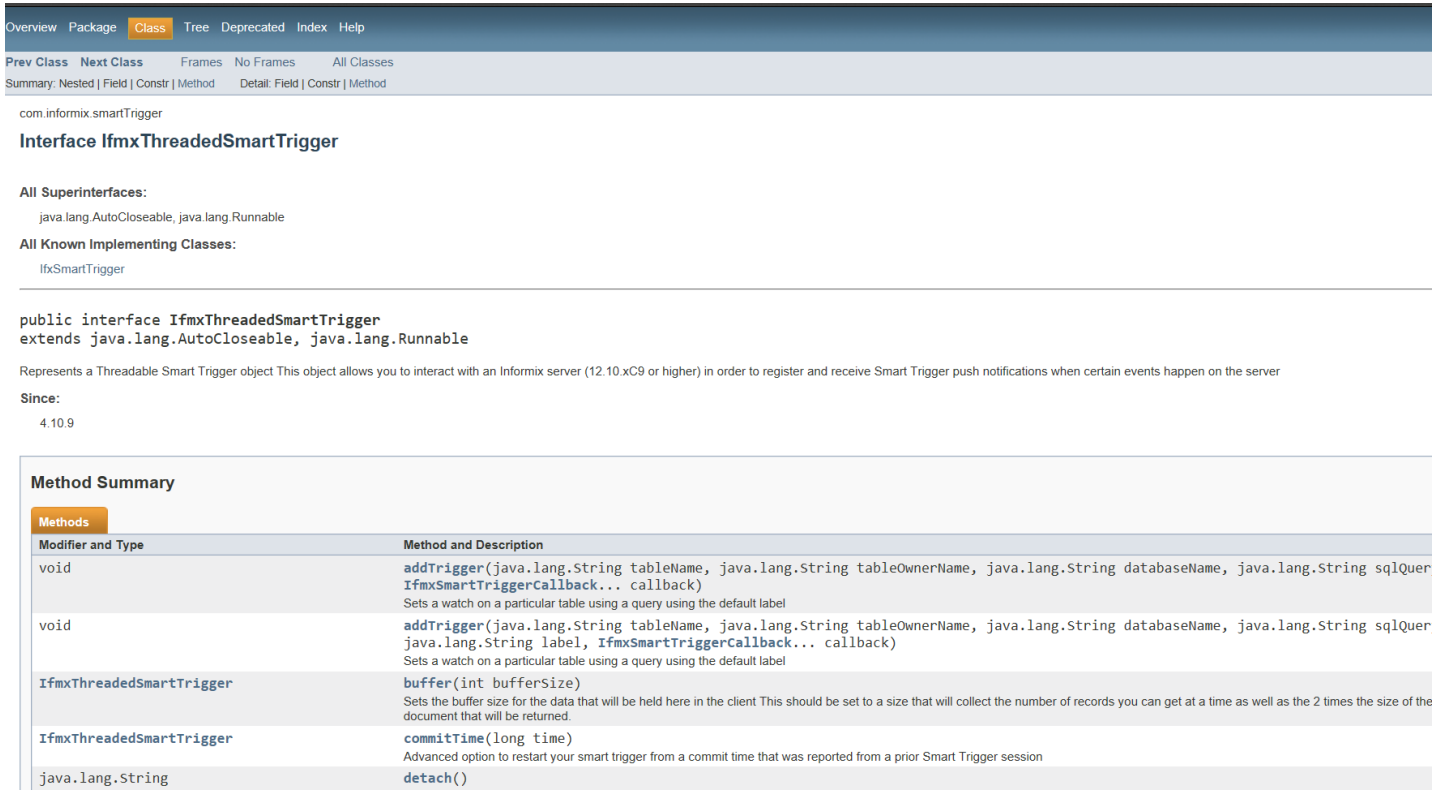
```
IfxSmartTrigger push = new IfxSmartTrigger(URL_HERE);
//Add the detachable flag to the properties before you call open()
push.label("test-label").timeout(-1).detachable(true);
push.open();
//After we have opened you can get the session id
String sessionID = push.getDetachableSessionID();
…
//Detach closes the client connection and returns the same session id
sessionID = push.detach();
```

You can reconnect to the session anytime, with a new process
even, by providing the session id

```
IfxSmartTrigger push = new IfxSmartTrigger(URL_HERE);
push.sessionID(sessionID);
BankMonitor callback1 = new BankMonitor();
push.registerCallback("test-label", callback1);
push.start();
```

Smart Trigger API documentation

- http://static.javadoc.io/com.ibm.informix/jdbc/4.10.10/com/informix/smartTrigger/package-summary.html



Overview Package Class Tree Deprecated Index Help

Prev Class Next Class Frames No Frames All Classes

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

com.informix.smartTrigger

Interface IfmxThreadedSmartTrigger

All Superinterfaces:

java.lang.AutoCloseable, java.lang.Runnable

All Known Implementing Classes:

IfxSmartTrigger

```
public interface IfmxThreadedSmartTrigger
extends java.lang.AutoCloseable, java.lang.Runnable
```

Represents a Threadable Smart Trigger object This object allows you to interact with an Informix server (12.10.xC9 or higher) in order to register and receive Smart Trigger push notifications when certain events happen on the server

Since:

4.10.9

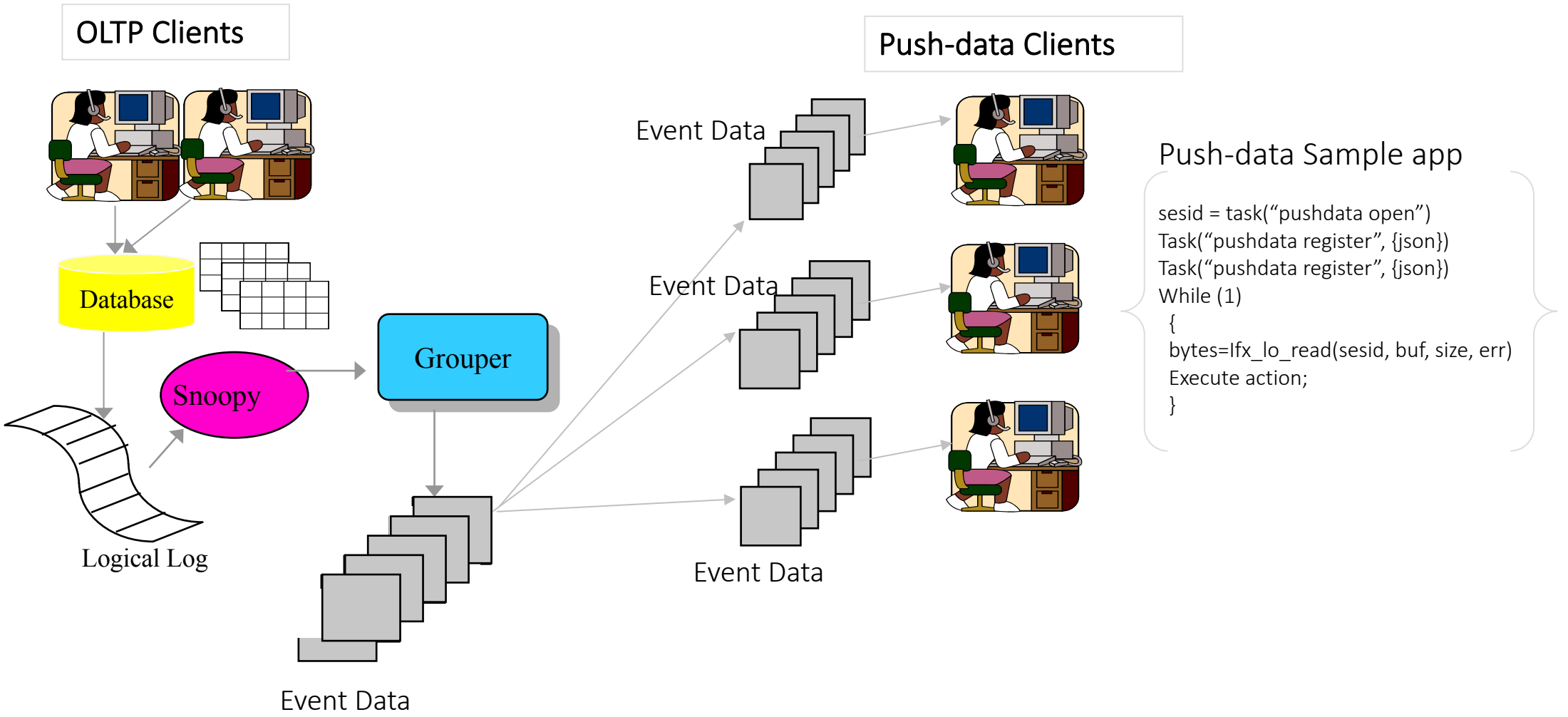
Method Summary

Methods

Modifier and Type	Method and Description
void	addTrigger(java.lang.String tableName, java.lang.String tableOwnerName, java.lang.String databaseName, java.lang.String sqlQuer IfmxSmartTriggerCallback... callback) Sets a watch on a particular table using a query using the default label
void	addTrigger(java.lang.String tableName, java.lang.String tableOwnerName, java.lang.String databaseName, java.lang.String sqlQuer java.lang.String label, IfmxSmartTriggerCallback... callback) Sets a watch on a particular table using a query using the default label
IfmxThreadedSmartTrigger	buffer(int bufferSize) Sets the buffer size for the data that will be held here in the client This should be set to a size that will collect the number of records you can get at a time as well as the 2 times the size of the document that will be returned.
IfmxThreadedSmartTrigger	commitTime(long time) Advanced option to restart your smart trigger from a commit time that was reported from a prior Smart Trigger session
java.lang.String	detach()

Push data functionality in server

Architecture Diagram



Push-data Sample app

```
sesid = task("pushdata open")
Task("pushdata register", {json})
Task("pushdata register", {json})
While (1)
 {
 bytes=Ifx_lo_read(sesid, buf, size, err)
 Execute action;
 }
```

API Calls

- TASK('pushdata open');
  - Register client session as a push data session
  - Returns session id, need this id to read event data.
- TASK('pushdata register', {event and session attributes});
  - Register event conditions, and session specific attributes
- Smart blob read API (ifx_lo_read() or equivalent call)to read event data
  - Pseudo smart blob interface to read event data.
  - Returns JSON document(s).
  - Can be configured as blocking or non-blocking call
- TASK('pushdata deregister', {event condition details});
  - De-register event conditions.

API Calls for detachable push-data sessions

- TASK(‘'pushdata setdetach'’);
  - Mark the push data session as a detachable session
  - Returns unique session id, need this to re-attach after reconnecting to server.
- TASK('pushdata join', "{session_id:”##"}”);
  - Reconnect to detached session.
  - Session id should match with unique id returned from ‘pushdata setdetach’ API.
- TASK('pushdata delete’);
  - Delete currently attached detachable session.
  - Push data session marked as detachable session must be deleted using ‘pushdata delete’ API.
- TASK('pushdata delete’, '{session_id:”##"}’);
  - Given unique session id, delete currently detached push data session.
- TASK('pushdata delete’, '{delete_all:"1"}’);
  - Delete all push-data sessions that aren't attached to client applications.

Comparing Smart Trigger and Regular I/U/D Trigger

Smart Trigger	Regular Trigger(I/U/D)
Post Commit	Pre Commit
Register Trigger on a specific Dataset/Event	Trigger gets fired for all changes
Asynchronous and Linear Scalability	Synchronous
Data is in JSON format	SQL format
Trigger logic gets executed in the client	Trigger logic gets executed in the server
Natural fit for event driven programming model	-
No schema changes required to define new smart trigger	Require schema changes and exclusive lock on the table to modify trigger definition

Comparing Push data and CDC

Push data	CDC
Designed for Smart Triggers	Designed for Data streaming/replication
Can register where clause	No where clause support
Data in JSON format	Byte stream
Push technology	Push technology
Only committed transactions are sent to Smart Trigger analysis	All records returned to the user including rollbacked operations
*Once the client disconnect from the server, events for the client aren't captured/staged	CDC can read old log files

Onstat commands

Print all event conditions:

```
> onstat -g pd event
      IBM Informix Dynamic Server Version 12.10.FC10 -- On-Line -- Up 00:20:13 -- 185676 Kbytes
      push-data subsystem structure at 0x4eebb028
      push-data session structure at 0x4eecc028
      push-data sql session id: 98 0x62
      Marked as detachable session, session unique id: 2
      Number of event conditions: 1
          Push-data event structure at 0x4ece0028
              Full Table Name: test:informix.t1
      User data:
      Replicate name: pushrepl_98_1497908205_1628814989
```

Print all sessions:

```
> onstat -g pd
      IBM Informix Dynamic Server Version 12.10.FC10 -- On-Line -- Up 00:21:31 -- 185676 Kbytes
      push-data subsystem structure at 0x4eebb028
      push-data session structure at 0x4eecc028
      push-data sql session id: 0 0x0
      Marked as detachable session, session unique id: 2
      Smartblob file descriptor: 39
      Number of event conditions: 0
      Number of pending event operations: 0
      Number of discarded event operations: 0
      Total event operations returned to client:
```

Onstat commands

Print information about specific session:

```
> onstat -g pd 98
      IBM Informix Dynamic Server Version 12.10.F -- On-Line -- Up 00:20:38 -- 185676 Kbytes
      push-data subsystem structure at 0x4eebb028
      push-data session structure at 0x4eecc028
      push-data sql session id: 98 0x62
      Marked as detachable session, session unique id: 2
      Smartblob file descriptor: 39
      Number of event conditions: 1
      Number of pending event operations: 0
      Number of discarded event operations: 0
      Total event operations returned to client: 0
```

Print event conditions for specific session:

```
> onstat –g pd 39 event
      push-data subsystem structure at 0x584cc028
      push-data session structure at 0x588f5028
      push-data session id: 39 (0x27)
      Marked as detachable session, session unique id: 2
      Number of event conditions: 1

      Push-data event structure at 0x461ed028
            Full Table Name: ycsb:informix.usertable
            User data: testing...
            Replicate name:         pushrepl_250_1487957951_1352060721
```

Questions ?