
How To ...

Table of Contents

Database	1
How to configure Shark to work with another database?	1
How to clear Shark database?	2
How to tune up database access?	2
Client interface	3
How to use Shark library	3
XPDL process definitions	7
How to write deadline expressions for activities?	7
How to write extended attributes to be able to update/view activity variables in shark's admin application	8
How to write XPDL to use custom Java classes as variables in Shark	9
How to define in XPDL that initial value of some variable should be 'null'	9
How to specify scripting language	10
How to use XPDL to directly map Application definition to particular ToolAgent (no need for Application mapping in runtime)	10

Abstract

You can find here answers to some frequently asked questions.

Database

How to configure Shark to work with another database?

After setup procedure finishes you'll have HipersonicSQL database created. While this is quite usable, Shark provides you an option to use other database vendor: *DB2, PostgreSQL, MySQL,....*

- first you'll need to stop any Shark instance that may be running(POJO swing admin/worklist handler, or CORBA server).
- Edit the `configure.properties` file and set values for:

`db_loader_job` name of the directory containing Octopus loader job, options are: *db2, hsql, informix, msql, mysql, oracle, postgresql, sybase*

`db_ext_dirs` directory containing jar file(s) with JDBC driver, if you need more then one directory specified here - use `${path.separator}` to concatenate them

`${db_loader_job}_JdbcDriver` class name of the *JDBC* driver you want to use

These entries are already filled with default values.

`${db_loader_job}_ConnectionFactoryURL` full database URL

These entries are already filled with default values, too.

`${db_loader_job}_user` username for database authentication

`${db_loader_job}_password` password for database authentication

- run the `configure.[bat|sh]`

Note

When loading newly created database, Octopus will complain about not being able to drop indices and tables, but these warnings should be ignored.

How to clear Shark database?

In the process of testing there will come the point you'll want to clear the database and start from the scratch. For clearing the database you may run the main `configure.[bat|sh]` file. If you don't want to wait for unnecessary filtering, archiving of war - you have `bin/recreatedB.[bat|sh]`.

The latter method runs only Octopus loader job to drop and create tables and indices.

How to tune up database access?

Shark engine is a pack of components, some of them are communicating with database using DODS [<http://dods.objectweb.org/>].

It may come in handy to know some parameters controlling DODS behaviour.

*DatabaseManager.DB.*The maximum number of connections that a connection pool will hold. If you know application won't need that much of concurrent connections, it's safe to bring this number down.*

*DatabaseManager.DB.*The number of Object Identifiers that are allocated as a group and held in memory. These identifiers are assigned to new data objects that are inserted into the database.*

*DatabaseManager.defaults.*The size of the cache is limited by the maximal number of objects that can be stored in it. When the cache is full, the objects in it are being replaced by new objects according to LRU (least recently used) algorithm.*

*DatabaseManager.defaults.cache.*Beside main object caches, there is possibility of using base query caches (simple and complex). These caches are also LRU caches.*

*DatabaseManager.defaults.*Every query that is executed longer than maxExecuteTime is printed (SQL statement, execution time and maxExecutionTime) in log file. This way you can pinpoint possible problems in your application or inside engine.*

```
DatabaseManager.DB.sharkdb.Connection.MaxPoolSize=300
DatabaseManager.DB.sharkdb.ObjectId.CacheSize=200
DatabaseManager.defaults.cache.maxCacheSize=100
DatabaseManager.defaults.cache.maxSimpleCacheSize=50
DatabaseManager.defaults.cache.maxComplexCacheSize=25
DatabaseManager.defaults.maxExecuteTime=200
```

As always bigger caches don't bring better performance :-), due to a memory consumption and other reasons.

Note

If you run multiple instances of engine using the same database (i.e. cluster), *DO NOT* use neither DODS nor Shark's caches.

Client interface

How to use Shark library

Client application uses Shark library through a set of interfaces in `org.enhydra.shark.api.client` package. The first thing the application should do is to configure library either by calling `configure()` method without parameters (then the configuration from `Shark.conf` file from jar is taken), or by specifying filename (as `String` or `File` object) or by preparing and calling the method with a `Properties` object. After configuration `org.enhydra.shark.Shark.getInstance()` returns an instance of `SharkInterface`. From this point on, it's application developer preference (or task) how it would use library, either getting a connection and executing assignments or getting an `AdminInterface` and managing users, groups, packages, ...

Example 1. Not very useful work-list handler

First line of this example configures engine using `conf/Shark.conf` file. Then after getting a connection and successfully connecting it asks `Resource` object how many assignments there are for this user (in line 4).

```
Shark.configure("conf/Shark.conf");
SharkConnection sConn = Shark.getInstance().getSharkConnection();
sConn.connect("qq", "lele", "shark", "");
if (0 < sConn.getResourceObject().how_many_work_item()) {
    System.err.println("Oh, let these tasks wait until tomorrow!");
}
System.out.println("Job done!");
```

Example 2. User group management

This example won't work if you set Shark to use LDAP user-group component, but database based component starts empty, and to do anything usefull you'll need to define at least one group and user.

```
Shark.configure();
UserGroupAdministration ugAdmin = Shark.getInstance()
    .getAdminInterface()
    .getUserGroupAdministration();
ugAdmin.createGroup("developers", "sweat-shop");
ugAdmin.createUser("developers",
    "user",
    "secret",
    "Jane",
    "Doe",
    "some@email.address");
System.out.println("Group and user created!");
```

Example 3. Loading package into Shark library

The location of the package's XPDL file is relative to the root of the package repository. Before you perform this operation, you might want to get all the package relative paths by calling method *getDefinedPackagesPath()* on your client object. First you need the location of the package's XPDL file relative to the root of the package repository, then you need a *PackageAdministration* instance.

```
String xpdlName = "test.xpdl";
Properties props = new Properties();
props.setProperty("enginename", "testSharkInstance");
props.setProperty("EXTERNAL_PACKAGES_REPOSITORY",
                  "c:/Shark/repository/xpdl");
Shark.configure(props);
String pkgId = Shark.getInstance()
    .getRepositoryManager()
    .getPackageId(xpdlName);
PackageAdministration pa = Shark.getInstance()
    .getAdminInterface()
    .getPackageAdministration();
if (!pa.isPackageOpened(pkgId)) {
    pa.openPackage(xpdlName);
}
System.out.println("Package " + xpdlName + " is loaded");
```

Example 4. Creating and starting process

After loading XPDL into shark, lets create, fill with initial variable values, and start some processes based on their XPDL definitions.

```
String pkgId = "test";
String pDefId1 = "basic";
String pDefId2 = "complex";

SharkConnection sConn = Shark.getInstance().getSharkConnection();

sConn.connect("user", "secret", "", "");

WfProcess proc1 = sConn.createProcess(pkgId, pDefId1);
WfProcess proc2 = sConn.createProcess(pkgId, pDefId2);

Map m1 = new HashMap();
m1.put("test_var",
      "This is String variable defined in XPDL for the process basic");
proc1.set_process_context(m1);
Map m2 = new HashMap();
m2.put("counter", new Long(55));
proc2.set_process_context(m2);

proc1.start();
proc2.start();
```

Example 5. Setting a variable

After successfully connecting to Shark, and acquiring list of assignments, lets do something useful, like setting a variable and completing the activity.

```
/*
SharkConnection sConn;
String activityId;
String vName;
String vValue;
*/
WfAssignment a = null;
WfAssignment[] ar = sConn.getResourceObject().get_sequence_work_item(0);
for (int i = 0; i < ar.length; ++i) {
    if (activityId.equals(ar[i].activity().key())) {
        a = ar[i];
        break;
    }
}
if (null == a)
    throw new BaseException("Activity:"
        + activityId
        + " not found in "
        + sConn.getResourceObject().resource_key()
        + "'s worklist");
if (!a.get_accepted_status())
    throw new BaseException("I don't own activity "+ activityId);
Map _m = new HashMap();
WfActivity activity = a.activity();
Object c = activity.process_context().get(vName);
if (c instanceof Long) {
    c = new Long(vValue);
} else {
    c = vValue;
}
_m.put(vName, c);
activity.set_result(_m);
activity.complete();
```

Example 6. Getting process managers based on some criteria

This example shows how to get process managers based on some criteria. It'll try to get all process managers which package Id is "test", and which state is enabled.

```
ExecutionAdministration eAdmin = Shark.getInstance()
    .getAdminInterface()
    .getExecutionAdministration();
eAdmin.connect("user", "secret", "", "");

WfProcessMgrIterator pmi = eAdmin.get_iterator_processmgr();
String query = "packageId.equals(\"test\") && enabled.booleanValue()";
pmi.set_query_expression(query);
WfProcessMgr[] procs = pmi.get_next_n_sequence(0);
```

Example 7. Getting processes based on some criteria

This example shows how to get processes created by some process manager based on some criteria. It'll try to get all processes which state is "open.running", which are started in last 10 minutes, which have more than 3 active activities, and which String variable called "myvariable" has value "test".

```
/*
WfProcessMgr mgr;
*/
WfProcessIterator wpi = mgr.get_iterator_process();
String query = "state.equals(\"open.running\") && "
              + "startTime.longValue()>(java.lang.System.currentTimeMillis()-10*60*
              + "activeActivitiesNo.longValue()>3 && "
              + "context_myvariable.equals(\"test\")";
wpi.set_query_expression(query);
WfProcess[] procs = wpi.get_next_n_sequence(0);
```

Example 8. Using external transactions

Each method of Shark API invocation presents separate transaction: engine internally creates, uses, optionally commits, and eventually releases transaction. This means that even quite simple code utilizing Shark may unknowingly use many transactions.

Sometimes, it is required to do things a differently, therefore SharkTransaction is introduced. An application's developer) may choose to use external transactions for number of reasons including usage of the same database to store application (non work-flow related) data, to avoid constantly creating/discarding transactions, ...

Of course, this approach comes with a price: you must obey the rules of usage. Transactions are created calling `Shark.getInstance().createTransaction()`; just before you release one, your application must call `Shark.getInstance().unlockProcesses(st)`; notifying Shark to do it's internal bookkeeping. If anything goes awry, you must catch a `Throwable` and call `Shark.getInstance().emptyCaches(st)`; Yes, you've read it right even `Error`'s must be caught, otherwise you'll leave the engine in undefined state.

Here is the variable setting example modified to use a single transaction.

```
/*
SharkConnection sConn;
String activityId;
String vName;
String vValue;
*/
SharkTransaction st = Shark.getInstance().createTransaction();
try {
    WfAssignment a = null;
    WfAssignment[] ar = sConn.getResourceObject(st)
        .get_sequence_work_item(st, 0);
    for (int i = 0; i < ar.length; ++i) {
```

```
        if (activityId.equals(ar[i].activity(st).key(st))) {
            a = ar[i];
            break;
        }
    }
    if (null == a) throw new BaseException("Activity:"
        + activityId
        + " not found in "
        + sConn.getResourceObject(st)
        + ".resource_key(st)
        + "'s worklist");
    if (!a.get_accepted_status(st)) throw new BaseException("I don't own activity "
        + activityId);

    Map _m = new HashMap();
    WfActivity activity = a.activity(st);
    Object c = activity.process_context(st).get(vName);
    if (c instanceof Long) {
        c = new Long(vValue);
    } else {
        c = vValue;
    }
    _m.put(vName, c);
    activity.set_result(st, _m);
    activity.complete(st);
    st.commit();
} catch (Throwable t) {
    Shark.getInstance().emptyCaches(st);
    st.rollback();
    if (t instanceof RootException)
        throw (RootException) t;
    else
        throw new RootException(t);
} finally {
    try {
        Shark.getInstance().unlockProcesses(st);
    } catch (Exception _) {}
    st.release();
}
```

XPDL process definitions

(You can easily create XPDLs by using our XPDL editor JaWE [<http://jawe.objectweb.org>].)

How to write deadline expressions for activities?

In shark deadline expressions along with all process variables, you can use special variables. The Java type of these variables is `java.util.Date`, and here is their description:

- `PROCESS_STARTED_TIME` - the time when the process is started
- `ACTIVITY_ACTIVATED_TIME` - the time when process flow comes to activity and assignments for the activity are created
- `ACTIVITY_ACCEPTED_TIME` - the time when the first assignment for the activity is accepted

Note

If activity is being rejected after its acceptance, or it is not accepted at all, the `ACTIVITY_ACCEPTED_TIME` is set to some maximum value in the future.

Here are some rules when creating deadline expressions:

- Deadline expressions has to result in `java.util.Date`
- If shark is setup not to re-evaluate deadlines, but to initially evaluate deadline limit times, `ACTIVITY_ACCEPTED_TIME` should not be used in expressions because it will contain some maximum time in the future
- There shouldn't be process variables (`DataField` or `FormalParameter` entities from XPDL) that have the same Id as the one of previously listed.

Here are few examples of deadline expressions:

```
// Deadline limit is set to 15 seconds after accepting activity
var d=new java.util.Date();
d.setTime(ACTIVITY_ACCEPTED_TIME.getTime()+15000);
d;
```

```
// Deadline limit is set to 5 minutes after activity is started (activated)
var d=new java.util.Date();
d.setTime(ACTIVITY_ACTIVATED_TIME.getTime()+300000);
d;
```

```
// Deadline limit is set to 1 hour after process is started
var d=new java.util.Date();
d.setTime(PROCESS_STARTED_TIME.getTime()+3600000);
d;
```

How to write extended attributes to be able to update/view activity variables in shark's admin application

In order to update activity variable (defined by XPDL) in shark admin application(s), XPDL activity definition must contain some predefined extended attributes.

Suppose that XPDL process definition contains variable (XPDL `DataField` tag) called "x", and variable (XPDL `FormalParameter` type) called "input_var".

If while executing activity you would like admin user only to be able to view those variables, you should define following Activity's extended attributes:

```
<ExtendedAttribute Name="VariableToProcess_VIEW" Value="x"/>
<ExtendedAttribute Name="VariableToProcess_VIEW" Value="input_var"/>
```

If you would like user to update the same variables, you should define following Activity's extended attributes:

```
<ExtendedAttribute Name="VariableToProcess_UPDATE" Value="x"/>
<ExtendedAttribute Name="VariableToProcess_UPDATE" Value="input_var"/>
```

How to write XPDL to use custom Java classes as variables in Shark

To be able to do that, you should define variable as XPDLs external reference, and set its location attribute to be the full name of the Java class you want to use. I.e., it should look like:

```
...
<DataField Id="participants" IsArray="FALSE">
  <DataType>
    <ExternalReference location="org.enhydra.shark.wrd.Participants"/>
  </DataType>
</DataField>
...
...
<FormalParameter Id="participantGroup" Mode="INOUT">
  <DataType>
    <ExternalReference location="org.enhydra.shark.wrd.Participants"/>
  </DataType>
</FormalParameter>
...
```

Maybe the better approach is to define TypeDeclaration element that would be of that type. In that case you can use it everywhere (you do not need time to define appropriate DataType when creating Application's/SubFlow's Formal-Parameters):

```
...
<TypeDeclaration Id="participants_type">
  <ExternalReference location="org.enhydra.shark.wrd.Participants"/>
</TypeDeclaration>
...
```

and then define DataField or FormalParameter as follows:

```
...
<DataField Id="participants" IsArray="FALSE">
  <DataType>
    <DeclaredType Id="participants_type"/>
  </DataType>
</DataField>
...
<FormalParameter Id="participantGroup" Mode="INOUT">
  <DataType>
    <DeclaredType Id="participants_type"/>
  </DataType>
</FormalParameter>
...
```

The classes specified by ExternalReference element must be in shark's classpath.

How to define in XPDL that initial value of some variable should be 'null'

You should simply write "null" for InitialValue element of DataField:

```
<DataField Id="participants" IsArray="FALSE">
  <DataType>
    <DeclaredType Id="participants_type"/>
    <InitialValue value="null"/>
  </DataType>
</DataField>
```

```
</DataType>
<InitialValue>null</InitialValue>
</DataField>
```

This enables you to use interfaces or abstract java classes as workflow variables. Concrete implementation of these variables can be created by some tool agent.

How to specify scripting language

Shark currently supports three script interpreters: JavaScript, BeanShell and Python (the last one is not fully tested). To tell shark which scripting language is used for writing conditional expressions (i.e. in Transition conditions), you should specify Package's script element:

```
# if you want to use java-like syntax (interpreted by BeanShell), specify:
<Script Type="text/java"/>

# if you want to use java script syntax, specify:
<Script Type="text/javascript"/>

# if you want to use python syntax, specify:
<Script Type="text/pythonscript"/>
```

Shark will complain if you do not specify Script, or if you specify value not supported by shark.

How to use XPDL to directly map Application definition to particular ToolAgent (no need for Application mapping in runtime)

If you would like to specify directly in XPDL what particular ToolAgent will be executed by Tool activity, you should define some extended attributes for XPDL Application definition.

The main extended attribute that should be defined by each Application definition that tends to be mapped to ToolAgent has name "ToolAgentClass", and its value should be full name of the class representing tool agent to be executed, i.e.:

```
<ExtendedAttribute Name="ToolAgentClass" Value="org.enhydra.shark.toolagent.JavaScriptTool
```

This attribute is read by Default tool shark's tool agent, and he executes specified ToolAgent based on the value of this attribute.

Other extended attributes are specific to implementation of the tool agent, and are read by them. I.e., JavaScript and BeanShell tool agent specify extended attribute named "Script", and its content is the script to be executed by this tool agent at the runtime. In this case, you are actually using XPDL for programming, i.e.:

```
<ExtendedAttribute Name="Script" Value="java.lang.System.out.println(&quot;I'm going to pe
```

This script performs multiplication of variable "a" and "b", and stores it in "c" (those variables are formal parameters of XPDL Application definition).