Enhydra Shark Admin

Table of Contents

What is an Enhydra Shark Admin Application?

Shark Admin application is Java swing application meant to be used by administrator to manage Shark engine. It can be configured to use shark directly as POJO library, shark deployed as EJB service or shark exposed as WEB Services through EJBs (default configuration is to use shark embedded through POJO interface). It can be used to load XPDL files into shark, unload it, update it, to instantiate and monitor shark's processes, to perform mappings among participant definitions and real users, and among application definitions and Tool agents, ... It contains a built-in worklist handler application that can be used to execute workitems, or to reassign workitems from one user to another.

Next sections describe the possibilities of admin application.

Starting Shark Admin Application

To start the shark admin application, you simply have to execute proper runSA script (depending on OS).

When application is started, the login screen is shown. To actually connect to the shark, first you have to enter your username. If you are using default configuration (provided in SharkClient.conf), you have to enter "admin" for username, and "enhydra" for password. After entering credentials, press OK button and you're going to connect to shark.

By default, admin application uses shark through POJO which means that shark runs in the same VM as the admin application. By editing configuration parameters in SharkClient.conf file you can make admin application use shark through EJB or WebService wrappers in which case you first have to deploy shark in EJB container using EAR file that can be generated for several containers (JBoss, JOnAS and Geronimo).

Using Shark Administrator application

The shark administrator application is divided into several logical parts. Each part will be described in following sections.

Repository Management

The repository management displays all available files in choosen XPDL repository. This is the place where you can manage XPDL repository. You can upload a new XPDL file to this repository, or delete one from the repository. By pressing a 'location' button you can change the location of your XPDL repository.

To upload new package, press 'upload' button. The dialog for choosing XPDL files from your local file system is shown. When you choose the package file you want to upload, the dialog for entering the file path relative to the repository is shown. Here you can enter the directory structure and the file name that your XPDL will have in the repository. You can enter something like: test/conformance/test1.xpdl.

After the file is uploaded into XPDL repository, it can be loaded into engine, and its processes can be started, which will be described in following sections.

You can also delete the files from repository by selecting the file you want to delete, and pressing 'delete' button.

Package Management

The package management displays all packages (XPDLs) that are loaded into engine. It enables you to load and unload packages to/from engine, as well as to update already loaded packages and to synchronize engine's package cache.

- Loading packages: To load package(s) into engine, you have to press the 'load' button, and select the one of the offered packages. The packages you can load are all packages from XPDL repository, except the ones you've already loaded, and the ones that have the same Id as already loaded packages. When you select a package from the list, its file name and Id are displayed in the text box. Then, if you press 'Load' button, package will be loaded into engine (if it is valid and if there are no problems while package loading), and you will be able to start processes instances based on the process definitions within that package.

  NOTE: If the package references some external packages, and you are using shark through POJO,

they will also be loaded into engine (of course, they also have to be valid). Otherwise (if using shark through EJBs), you have to upload referenced packages first.

NOTE: if the file you want to load into engine is not 'shark' valid, the error messages describing problems will be shown, and the package want be uploaded

- Unloading packages: To unload package from engine, you have to select the wanted package, and press the 'unload' button. If there are no instantiated processes from that package's process definitions that are still held into DB, and this package is not referenced by any other package, it will be unloaded from the engine. After that, you will not be able to instantiate the processes from its process definitions.

  In demo/professional version, you also have possibility to unload all versions of selected package, but than the above requirements must be fulfilled for each package version.

- Updating packages (not available in community version): If you want to update package, you can do it by selecting it, and pressing 'update' button. The list of the packages from repository, with the same Ids as the one you want to update, is shown. You can select a package from the list, and press 'update' button. The processes that were running based on old package's process definitions remain to run based on them, but you'll be able to create processes based on definitions contained in new package version.

  If the file you want to update the package file with, is not 'shark' valid, the error messages describing problems will be shown, and the package want be updated

Process Instantiation Management

Here from, you can view the package-processdefinition tree of the loaded packages. In demo/professional version, if you select a package from the tree, and press left-mouse button, you can get the property dialog of the package. If you select some of the package's process definition, you can also get the property dialog by right-mouse button, but you can also do some other things:

- In the right pane, you can see some general process properties, along with the number of currently running processes based on this process definition.
- You can create a new running instance of the process and start it by pressing button 'Instantiate'
- In demo/professional version, you can see the graphical presentation of the process by clicking the 'View' button
- You can see the description taken from process definition by pressing 'Description' button
- In demo/professional version, you can enable or disable specific process definition or whole process definitions for package(s).

Process Monitor

Process monitor is divided into four major parts (in community version, only three). The "package-processdefinition-processinstances" tree enables you to select a running instance of a package's process definition. When you select the process instance, other section's data correspond to this process instance. You can see the main properties of the instance (the name, and the current state), in professional/demo version you can see the graphical diagram of the process instance with activities that are currently running being marked, and you can perform different operations on that process instance using the buttons at the bottom.

The operations you can perform are:

- start the process - this can be done in the case that process is in open.not_running.not_started state
- suspend the process - all its active activities and synchronous sub-processes instantiated by some active subflow activities will be suspended
- resume the process - all its activities and synchronous sub-processes instantiated by an active subflow activities will be resumed NOTE: if you try to resume a synchronous process started by some subflow activity of the suspended process - you will not be able to do it, it will be automatically resumed when the 'parent' activity is resumed
- terminate the process - all activities and synchronous sub-processes instantiated by an active subflow activities will be terminated (if using professional/demo version you can perform group action - based on selected process definition, package definition or all package definitions)
- abort the process - all activities and synchronous sub-processes instantiated by an active subflow activities will be aborted (if using professional/demo version you can perform group action - based on selected process definition, package definition or all package definitions)
- (in professional/demo version only) view the process history - the chronological view of what have happened since the process started (when the process started, when process changed its state, when some process variables are changed, when some process activity changed state, when a process activity variables changed, when are activity assigned to resource, ...)
- see the description of the process
- see and edit the process variables, and that way you can manage the process flow if needed (if a transition condition depends on the variable value)
- enter activity management dialog. The dialog displays the list of process activities, and when you select one of them, its current state is displayed in the text box. In professional/demo version, from this dialog, you can perform the similar operations on single activities:
  - start activity (accept it)
  - suspend activity
  - resume activity
  - complete activity
  - terminate activity (when you terminate an activity, process proceeds to the next activities if transition conditions are satisfied)
  - abort activity - process becomes 'stucked'
  - manually start an activity
- delete selected finished process (if using professional/demo version you can perform group action -

based on selected process definition, package definition or all package definitions)

- re-evaluate assignments for selected process (if using professional/demo version you can perform group action - based on selected process definition, package definition or all package definitions)
- perform a check for activity deadlines for selected process (if using professional/demo version you can perform group action - based on selected process definition, package definition or all package definitions)
- perform a check for limits for selected process and its activities (if using professional/demo version you can perform group action - based on selected process definition, package definition or all package definitions)

User Management

It is divided into three parts:

- Groups - You can manage the groups by defining the new ones, deleting the existing ones or changing their properties.

  NOTE: If shark is configured to use LDAP implementation of UserGroup manager, you will not be able to create, modify or delete group, but just to see existing ones.

- Users - You can manage the users by defining the new ones, deleting the existing ones or changing their properties.

  NOTE: If shark is configured to use LDAP implementation of UserGroup manager, you will not be able to create, modify or delete user, but just to see existing ones.

- Mapping - enables you to map the package and package's processes participants to the real shark users. When you define some mappings, and the process comes to the point when an activity needs to be performed by participant that is mapped to one or more real users, the workitem will be placed on the worklist of every mapped user.

Application Mapping

In professional/demo version, you can map a package and package's processes applications to the real applications handled by a tool agent. Currently, six tool agents come with the Shark distribution. To map application definition to tool agent application, you have to go to the application mapping section of admin application, and press the "add" button. The dialog will arise, and you have to select the application definition at the left side of dialog, and the tool agent on the right side of the dialog. Then you should enter some mapping parameters for tool agent. When you map the application definition to the tool agent, shark will try to connect to the proper tool agent and ask him to execute its application, and will retrieve the results of execution. Here is the brief description of parameters you can enter when mapping application definition to tool agent application:

- username and password - not required for tool agents distributed with Shark. Some other tool agents can use it when calling applications that require login procedure
- Application name - the name of application that should be started by tool agent (i.e. for JavaClassToolAgent that would be the full name of the class, for RuntimeApplicationToolAgent it would be the name of executable file that should be in the path of the machine where tool agent resides, for JavaScriptToolAgent this can be either the name of the java script file, or the java script itself, depending on Application mode attribute), for SOAPToolAgent it is the location of WEB service and for MailToolAgent it is a class of MailMessageHandler called to actually send/receive mails.
- Application mode - various tool agents use this attribute for different purposes. I.e., RuntimeApplicationToolAgent uses mode 0 to indicate that it should not finish execution until the system application is finished (otherwise it will start system application and return finished status -> activity does not wait for system application to finish, but process proceeds to the next activity), and JavaScriptToolAgent uses mode 0 to indicate that it should search for java script file (otherwise, the application name will be considered the java script)

You can find more about tool agent mappings in Tool Agent documentation.

Cache Management

In professonal/demo version, you can handle the size of the shark caches using this section. You can change the size of Process and Resource caches, as well as to clear the caches.

Work List

Here from, you can execute the workitems of the instantiated processes. You can perform your workitems, and you can see other's workitems. In professional/demo version, you have a possibility to reassign the workitem from one user to another.

The workitem is executed by pressing 'complete' button, or by double-clicking it in the table. If the workitem has variables that are meant to be updated by the user, you will be asked to do so when 'completing' it, or you can enter the update dialog before you complete the workitem (but only if you have accepted it).

When a workitem is put into the list of two or more different users, it will stay there till any of them accept it. When someone accepts the workitem, it is removed from other user's worklists, and if he reject it afterwards, the workitem will be put back to the proper users worklist.

NOTE: TO BE ABLE TO UPDATE OR VIEW VARIABLES WHEN EXECUTING A WORKITEM, THE ACTIVITY DEFINITION HAS TO HAVE SOME SPECIAL EXTENDED ATTRIBUTEs DEFINED, AND HERE ARE THE EXAMPLES:

- if you want to allow performer to update the 'x' process variable when executing activity, when creating the process definition, you should define the extended attribute for that activity as follows:

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

- if you want to allow performer only to see the 'y' and 'z' process variables when executing activity, when creating the process definition, you should define two extended attributes for that activity as follows:

```
<ExtendedAttribute Name="VariableToProcess_VIEW",Value="y"/>
<ExtendedAttribute Name="VariableToProcess_VIEW",Value="z"/>
```

- if you want to allow performer to update 'x', 'y' and 'z' process variables, and to see 'a', 'b' and 'c' variables, you should define the following extended attributes for given activity:

```
<ExtendedAttribute Name="VariableToProcess_UPDATE",Value="x"/>
<ExtendedAttribute Name="VariableToProcess_UPDATE",Value="y"/>
<ExtendedAttribute Name="VariableToProcess_UPDATE",Value="z"/>
<ExtendedAttribute Name="VariableToProcess_VIEW",Value="a"/>
<ExtendedAttribute Name="VariableToProcess_VIEW",Value="b"/>
<ExtendedAttribute Name="VariableToProcess_VIEW",Value="c"/>
```

YOU CAN SIMPLY DO ALL OF THAT BY USING Enhydra JaWE [http://jawe.objectweb.org] WORKFLOW EDITOR

Process List

Here from, you can see the process instances you've created, and if you are allowed, you can also see the process instances created by other users.