## Getting Started with the SAS® Life Science Analytics Framework Java API

Since you are reading this document, you've already expanded the client distribution file (lsaf-java-api-client-<release>.zip) into a directory. In that directory is now an "lsaf-java-api-client-<release>" subdirectory which will be referred to as *HOME*.

In your HOME directory you should see:

- README
- docs [a directory containing javadoc]
- lib [a directory of jars the API needs to run]
- log4j2.properties [a file to configure client logging]

This document assumes that the server-side portion of the API has already been installed on your application server. Speak to your system administrator if this has not been done.

The API requires that you have a Java Runtime Environment (JRE) version 8 installed. At your command prompt you can type "java –version" to confirm this.

## Verifying Your Client Installation

To run one of the sample programs or verify your client installation:

- 1. From a command prompt, change directory to HOME/lib.
- 2. Run the following command:

java -jar sas.lsaf.api.client.jar -h

You should see a list of commands that this sample program provides. This indicates the client install has been done correctly.

To verify your deployment and your connection to the SAS LSAF server, run the command:

java -jar sas.lsaf.api.client.jar –u userid –p password –s <u>https://<LSAF\_HOST\_NAME></u> –test

You should see a success message indicating that the connection was successful.

## Setting up a Development Environment

The SAS Life Science Analytics API requires the use of Java Runtime Environment (JRE) version 8. For this document we'll assume you are using Eclipse.

1) In Eclipse, create a new java project, specifying a Java 8 execution environment:

| 💓 New Java Project  |                      | — 🗆 X            |  |  |
|---|----------------------|------------------|--|--|
| Create a Java Project<br>Create a Java project in the workspace or in an  | external location.   |                  |  |  |
| Project name: myLsafApiProject  |                      |                  |  |  |
| Use <u>default location</u>   | afApiProject         | B <u>r</u> owse  |  |  |
| JRE   |                      |                  |  |  |
| Use an execution environment JRE:   | JavaSE-1.8           | ~                |  |  |
| <ul> <li>Use a project specific JRE:</li> <li>Use def<u>ault JRE (currently 'jre1.8.0_102')</u></li> </ul>  | jre1.8.0_102         | Configure JREs   |  |  |
| Project layout<br>O <u>U</u> se project folder as root for sources and  | d class files        |                  |  |  |
| • <u>Create separate folders for sources and c</u>  | lass files <u>C</u>  | onfigure default |  |  |
| Working sets  |                      | Ne <u>w</u>      |  |  |
| W <u>o</u> rking sets:  | ~                    | S <u>e</u> lect  |  |  |
| (i) The default compiler compliance level for the current workspace is 1.7. The new project will use a project specific compiler compliance level of 1.8. |                      |                  |  |  |
| (?) < <u>B</u> ack <u>N</u>   | ext > <u>F</u> inish | Cancel           |  |  |

2) Once the project is made, open the project properties and make sure the JRE 8 library is in the java build path setting:

| Properties for myLsafApiProj  | ect  | — 🗆 X  |
|---|--|--|
| type filter text  | Java Build Path  | ↓ ↓ ↓ ↓  |
| <ul> <li>&gt; Resource</li> <li>BPMN2</li> <li>Builders</li> <li>Coverage</li> <li>FindBugs</li> <li>Java Build Path</li> <li>&gt; Java Code Style</li> <li>&gt; Java Compiler</li> <li>&gt; Java Editor</li> <li>Javadoc Location</li> <li>Project Facets</li> <li>Project References</li> <li>Run/Debug Settings</li> <li>Server</li> <li>&gt; Task Repository</li> <li>Task Tags</li> <li>&gt; Validation</li> <li>WikiText</li> </ul> | Source Projects Libraries Order and Export JARs and class folders on the build path: | Add <u>J</u> ARs<br>Add External JARs<br>Add <u>V</u> ariable<br>Add Library<br>Add <u>C</u> lass Folder<br>Add External Class Fol <u>d</u> er<br><u>E</u> dit<br><u>R</u> emove<br><u>M</u> igrate JAR File |
|   |  | Apply  |
| ?   | E  | OK Cancel  |

3) Click the "Add External JARs…" button. Navigate to the lib directory of *HOME*, select all the jars there, and add them to your project (selection of jars may differ slightly depending on the release of the API in use).

| Properties for myLsafApiProje   | ect  | – 🗆 X   |
|---|--|---|
| type filter text  | Java Build Path  | <> ▼ <> ▼ ▼   |
| <ul> <li>Resource<br/>BPMN2<br/>Builders<br/>Coverage<br/>FindBugs<br/>Java Build Path</li> <li>Java Code Style</li> <li>Java Compiler</li> <li>Java Compiler</li> <li>Java Editor<br/>Javadoc Location<br/>Project Facets<br/>Project References<br/>Run/Debug Settings</li> <li>Task Repository<br/>Task Tags</li> <li>Validation<br/>WikiText</li> </ul> | <ul> <li>Source Projects Libraries Order and Export</li> <li>JARs and class folders on the build path:         <ul> <li>aopalliance-1.0.0.0_SAS_20100917120439.jar - C:\temp\lsaf-java-api-client-2.4\lib</li> <li>commons-codec-1.10.jar - C:\temp\lsaf-java-api-client-2.4\lib</li> <li>commons-io-2.6.jar - C:\temp\lsaf-java-api-client-2.4\lib</li> <li>commons-lang3-3.4.jar - C:\temp\lsaf-java-api-client-2.4\lib</li> <li>commons-logging-1.2.jar - C:\temp\lsaf-java-api-client-2.4\lib</li> <li>momons-logging-1.2.jar - C:\temp\lsaf-java-api-client-2.4\lib</li> <li>momons-2.3.jar - C:\temp\lsaf-java-api-client-2.4\lib</li> <li>mogdj-japi-2.13.2.jar - C:\temp\lsaf-java-api-client-2.4\lib</li> <li>mogdj-japi-2.13.2.jar - C:\temp\lsaf-java-api-client-2.4\lib</li> <li>moddj-java-api-client-2.4\lib</li> <li>moddj-java-api-client-2.4\lib</li> <li>moddj-java-api-client-2.4\lib</li> <li>moddj-java-api-3.20.RELEASE.jar - C:\temp\lsaf-java-api-client-2.4\lib</li> <li>moddj-java-api-3.20.RELEASE.jar - C:\temp\lsaf-java-api-client-2.4\lib</li> <li>moddj-java-api-client-2.4\lib</li> <li>moddj-java-api-client-2.4\lib</li> <li>moddj-java-api-client-2.4\lib</li> <li>moddj-java-api-client-2.4\lib</li> <li>moddj-java-api-client-2.4\lib</li> <li>moddj-java-api-client-2.4\lib</li> <li>moddj-java-a</li></ul></li></ul> | Add JARs<br>Add External JARs<br>Add Variable<br>Add Library<br>Add Class Folder<br>Add External Class Folder<br>Edit<br>Remove<br>Migrate JAR File |
| (?)   |  | OK Cancel   |

4) Finally, you can create your main program, starting with logging on to your instance of the application:

To simplest way to start is to logon via the <u>LsafClient</u>. LsafClient allows a static logon and static access to services like the <u>RepositoryService</u>. For example:

```
import com.sas.lsaf.LsafClient;
import com.sas.lsaf.content.repository.RepositoryService;
...
LsafClient.logon("https://yourLSAFMachine.domain.com", "myLSAFuserId",
"myLSAFpwd".getBytes());
RepositoryService repositoryService = LsafClient.getRepositoryService();
repositoryService.checkout("/YOURORG/Files/Folder/someProgram.sas");
...
```

Once logged in, a single user session is established and used when accessing the services. Note subsequent calls to logon will logoff the current session and establish a new session.

Alternatively, you can log on using SessionFactory.logon(URL,String, byte[]) which will create and return a Session. From the Session, you can access services and manage the session.

Each SessionFactory logon creates a new Session.