Use the Indivo Java API in a PHA

1. Required jars

The **indivo-api.jar** and all the .jar files in the **lib/** directory must be added to the classpath of the PHA.

2. Sources

The **src/** directory contains the source code of the API and a sample PHA that showcase the use of the API.

3. Javadoc

The **doc/** directory contains the API documentation in javadoc format.

4. Create the Indivo object

The base of the API is the Indivo object. It is created thanks to the server coordinates and the PHA consumer information:

```
Indivo indivo=new Indivo(consumerKey, consumerSecret, appId
                         indivoServerUrl, uiServerUrl);
```

Here is an example of such configuration:

```
consumerKey="citron2";
consumerSecret="d0595ddbed79";
appId="citron@orange.org";
indivoServerUrl="http://10.20.3.51:8080/";
uiServerUrl="http://10.20.3.51:8002/";
```

Consumer key and secret are related to the PHA while the server URL are related to the Indivo server.

5. OAuth: handle the Start URL

Indivo uses OAuth authentication process. Therefore any PHA should handle the Start URL (given to Indivo at PHA declaration). The Indivo API provides an abstract servlet to handle it: com.orange.indivo.template.StartUrlServlet. This servlet asks a request token to Indivo, stores it into the HTTP session and redirect the browser to the Indivo authorize URL. To handle the OAuth Start URL, simply create a servlet which extends the StartUrlServlet:

```
public class IndivoStartServlet extends StartUrlServlet {

    public IndivoStartServlet(){
            // create the Indivo object thanks to a dedicated factory
            super(IndivoFactory.getIndivo());
    }
}
```

6. OAuth: handle the Callback URL

Like the Start URL, any PHA sould handle the Callback URL (also given to Indivo at PHA declaration). The Indivo API provides an abstract servlet to handle it: com.orange.indivo.template.CallbackUrlServlet. This servlet asks an access token to Indivo. To handle the OAuth Callback URL, simply create a servlet which extends the CallbackUrlServlet:

```
public class IndivoCallbackServlet extends CallbackUrlServlet {

    public IndivoCallbackServlet(){
            // create the Indivo object thanks to a dedicated factory
            super(IndivoFactory.getIndivo());
    }
      protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            super.doGet(request, response);
            // access token is available for storing
            response.getOutputStream().println(this.accessToken);
    }
}
```

7. Create a document manager

Thanks to an Indivo object, you may get instances of document managers. For example if you wish to use the VitalSign Indivo documents:

```
VitalSignManager vitalManager=indivo.getVitalSignManager();
```

8. List documents of a particular type

Here is the best method to use to get the list of VitalSign documents within an Indivo record:

```
Reports<VitalSign>
reports=vitalManager.getVitalSignReportsForRecord(accessToken);
For(Report<VitalSign> report: reports.getReportList()){
      // get meta data on the document (dates, size, author, etc.)
      DocumentMeta meta=report.getMeta();
      // get a VitalSign object, representing the document
      VitalSign vital=report.getMedical();
}
```

9. Add a document

Here is the simplest way to create aVitalSign document and post it to Indivo:

```
Date dateMeasured=new Date();
String name="Blood Pressure Systolic";
BigDecimal value=new BigDecimal(127);
String unit=new "millimeters of mercury";
VitalSign vital=new VitalSign(dateMeasured, name, value, unit);
vitalManager.postVitalSignForRecord(accessToken,vital);
```

VitalSign objects can include more detailed information but we wanted to keep it simple here.