```java
/**
 *
 */
package uk.leeds.csp.client;

import org.junit.Assert;

import org.apache.log4j.Logger;
import org.apache.xmlbeans.XmlDateTime;
import org.apache.xmlbeans.XmlObject;

import java.util.Calendar;
import java.util.GregorianCalendar;

import javax.security.auth.login.LoginContext;
import javax.security.auth.login.LoginException;
import org.ogf.graap.wsag.security.core.KeystoreProperties;
import org.ogf.graap.wsag.security.core.keystore.KeystoreLoginContext;
//import org.ogf.graap.wsag.security.core.*;
import org.w3.x2005.x08.addressing.EndpointReferenceType;
import org.ogf.graap.wsag.client.AgreementFactoryRegistryLocator;
import org.ogf.graap.wsag.api.client.AgreementFactoryRegistryClient;
import org.ogf.graap.wsag.api.client.AgreementFactoryClient;
import org.ogf.graap.wsag.api.client.NegotiationClient;
import org.ogf.graap.wsag.api.client.AgreementClient;
import org.ogf.graap.wsag.api.exceptions.ResourceUnknownException;
import org.ogf.graap.wsag.api.exceptions.NegotiationException;
import org.ogf.graap.wsag.api.exceptions.ResourceUnavailableException;
import org.ogf.graap.wsag.api.exceptions.ResourceUnknownException;
import org.ogf.graap.wsag.api.exceptions.ResourceUnavailableException;
import org.ogf.graap.wsag.api.AgreementOffer;
import org.ogf.graap.wsag.api.types.AgreementOfferType;

import org.ogf.schemas.graap.wsAgreement.AgreementTemplateType;
import org.ogf.schemas.graap.wsAgreement.negotiation.NegotiationContextDocument;
import org.ogf.schemas.graap.wsAgreement.negotiation.NegotiationRoleType;
import org.ogf.schemas.graap.wsAgreement.negotiation.NegotiationType;
import org.ogf.schemas.graap.wsAgreement.negotiation.NegotiationOfferContextType;
import org.ogf.schemas.graap.wsAgreement.negotiation.NegotiationOfferStateType;
import org.ogf.schemas.graap.wsAgreement.negotiation.NegotiationConstraintSectionType;
import org.ogf.schemas.graap.wsAgreement.negotiation.NegotiationOfferItemType;
import org.ogf.schemas.graap.wsAgreement.OfferItemType.ItemConstraint;
import org.ogf.schemas.graap.wsAgreement.ServiceDescriptionTermType;
import org.ogf.schemas.graap.wsAgreement.ServiceTermStateType;
import org.ogf.schemas.graap.wsAgreement.GuaranteeTermStateType;
import org.ogf.schemas.graap.wsAgreement.AgreementStateType;
import org.ogf.schemas.graap.wsAgreement.TermTreeType;

import org.ogf.graap.wsag.samples.actions.SampleAgreementTemplate;
import org.ogf.graap.wsag.samples.actions.SampleNegotiationOffer;

import org.ggf.schemas.jsdl.x2005.x11.jsdl.ResourcesType;
import org.ggf.schemas.jsdl.x2005.x11.jsdl.RangeValueType;
```

```
54  import org.ggf.schemas.jsdl.x2005.x11.jsdl.JobDefinitionDocument;
55  import org.ggf.schemas.jsdl.x2005.x11.jsdl.JobDefinitionType;
56
57  import org.ogf.schemas.graap.wsAgreement.negotiation.NegotiationOfferType;
58
59  import org.wsag4j.types.scheduling.TimeConstraintType;
60  import org.wsag4j.types.scheduling.TimeConstraintDocument;
61
62
63  import org.ogf.schemas.graap.wsAgreement.negotiation.NegotiationContextType;
64
65
66  /**
67   * @author adriano
68   *
69   */
70  public class Main {
71
72      private static final Logger LOG = Logger.getLogger( Main.class);
73      /**
74       * @param args
75       */
76      public static void main(String[] args) {
77          try {
78
79              /*** LOADING AGREEMENT FACTORY CLIENTS & TEMPLATES ***/
80
81              /**
82               * Since WSAG4J uses WS-Security to digitally sign messages exchanged
                 between client and server,
83               * a client must provide a set of security credentials.
84               */
85              System.out.println(" 1) Before getLoginContext");
86              LoginContext loginContext = getLoginContext();
87
88              /**
89               * This LoginContext is passed to the AgreementFactoryLocator that allows
                 retrieval of available agreement factory clients
90               * from the deployed wsag4j server identified by its url as shown below.
91               * The client factory array (factories) stores objects of
                 AgreementFactoryClient's type.
92               */
93              System.out.println(" 2) ");
94              EndpointReferenceType wsag4j_server_epr = EndpointReferenceType.Factory.
                newInstance();
95              System.out.println(wsag4j_server_epr);
96              wsag4j_server_epr.addNewAddress().setStringValue("
                http://127.0.0.1:8080/wsag4j-agreement-factory-1.0.2");
97              System.out.println(wsag4j_server_epr);
98              System.out.println(" 3) Before retreiving the registry");
99              AgreementFactoryRegistryClient registry = AgreementFactoryRegistryLocator.
                getFactoryRegistry(wsag4j_server_epr, loginContext);
100             //System.out.println(" 4) " + registry);
```

```
101             AgreementFactoryClient[] factories = registry.listAgreementFactories();
102
103             System.out.println(" factories length: "+factories.length);
104             /**
105              * It is possible to get a particular agreement factory specified by an ID.
106              * The agreement factory IDs can be specified in the wasg4j-engine.config
                 file.
107              * In current snippet an agreement factory with ID "SAMPLE-INSTANCE-1" is
                 retrieved:
108              */
109             System.out.println(" 5)");
110             AgreementFactoryClient factory = null;
111
112             for (int i = 0; i < factories.length; i++) {
113                 System.out.println(factories[i].getResourceId());
114                 if ("CARMEN-INSTANCE-1".equals(factories[i].getResourceId())) {
                    //"SAMPLE-INSTANCE-1"  The same in the line 360
115                     factory = factories[i];
116                     //break;
117                 }
118             }
119             System.out.println(" 5a) SAMPLE-INSTANCE-1 - SAMPLE-INSTANCE-2 and
                ISQoS_TEST are in " +
120                     "../WEB-INF/classes/wsag4j-engine/instance1 and
                        ../WEB-INF/classes/wsag4j-engine/instance1");
121             System.out.println();
122             System.out.println(" WSAG4J supports multiple agreement factories within
                one web application.\n" +
123                     " This can be useful e.g. when you have a set of systems that
                        should be accesses via the WS-Agreement protocol.\n" +
124                     " Instead of having a separate web application for each system,
                        wsag4j allows you to have one agreement factory\n" +
125                     " per system, each factory represented by a separate engine.\n" +
126                     " The WSAG4JEngineInstances configuration section allows you to
                        specify a set of configuration files,\n" +
127                     " where each configuration file configures a separate
                        AgreementFactory.");
128             System.out.println();
129             System.out.println(" You have to add the path where the instance is in
                ../WEB-INF/classes/wsrf-engine.config");
130
131             /**
132              * After the agreement factory client is created,
133              * this client can be used to retrieve valid agreement templates from the
                 factory.
134              * The next example shows how an agreement factory client retrieves all
                 templates from the agreement factory:
135              */
136             /**
137              * AgreementTemplateType[] templates = factory.getTemplates();
138              */
139
140             /**
```

```
141                 * Retrieving templates by ID is also possible,for example,
142                 * the sample template with template name "SAMPLE-TEMPLATE" and id "1" from
                    the "SAMPLE-INSTANCE-1" agreement factory:
143                 */
144
145                 //AgreementTemplateType template2 = factory.getTemplate("SAMPLE-TEMPLATE",
                    "1");
146
147                 //System.out.println(" ++++ " + template2.xmlText());
148
149                /**
150                 * After an agreement template was retrieved from the factory,
151                 * a client can use this template to create a valid agreement offer.
152                 * The agreement offer is a request to create a new service level agreement
                    under the conditions stated in the offer.
153                 * An agreement offer can be created by the following code:
154                 */
155                //AgreementOffer offer = new AgreementOfferType(template); <--- This seems
                   to be placed later in the code
156
157                //System.out.println(offer.getXMLObject());
158
159                /*** NEGOTIATION AGREEMENT ***/
160                //NegotiationClient negotiation = factory.initiateNegotiation(arg0);
                   //org.ogf.schemas.graap.wsAgreement.negotiation.NegotiationContextType
161                NegotiationClient negotiation = null;
162                //
163                // initiate the negotiation instance
164                //
165                negotiation = initiateNegotiation(negotiation, factory);
166
167                //
168                // retrieve the agreement templates for which negotiation is supported,
169                // and select the one with template name "SAMPLE-TEMPLATE".
170                //
171                System.out.println(" 12) Getting negotiation template ");
172
173                AgreementTemplateType[] negotiableTemplates = negotiation.
                   getNegotiableTemplates();
174                System.out.println(" 12a) Numbers of negotiable templates : " +
                   negotiableTemplates.length);
175                System.out.println("      The List of the negotiable templates \n" +
176                                   "      (for one of the instances in
                                   ../classes/wsag4j-engine/(instance1, instance2,
                                   ISQoSInstance) ) \n" +
177                                   "      are in ../classes/samples/ ");
178
179                AgreementTemplateType template = null;
180                for ( int i = 0; i < negotiableTemplates.length; i++ )
181                {
182                    AgreementTemplateType agreementTemplate = negotiableTemplates[i];
183                    System.out.println();
184                    System.out.println( "retrieved template: " + agreementTemplate.getName()
```

```
                    + ":" + agreementTemplate.getTemplateId() );
185                 System.out.println( agreementTemplate.toString() );
186
187                 if ( agreementTemplate.getName().equals( "CARMEN-TEMPLATE-1" ) )   //
                    "SAMPLE-TEMPLATE"
188                 {
189                     template = agreementTemplate;
190                 }
191             }
192             /*
193             System.out.println( "Overview of the available factories and their
                templates." );
194             for ( int i = 0; i < 2; i++ )
195             {
196                 System.out.println( "+ " + factories[i].getResourceId() );
197
198                 AgreementTemplateType[] templates = factories[i].getTemplates();
199                 System.out.println( "   + Num of templates: " + templates.length );
200
201                 for ( int k = 0; k < templates.length; k++ )
202                 {
203                     System.out.println( "      + " + templates[k].getName() );
204
205                     if(templates[k].getName().equals("SAMPLE3"))
206                     {
207                         System.out.println(" 13) " + templates[k].xmlText());
208                     }
209                 }
210             }
211             */
212             //
213             // The "SAMPLE-TEMPLATE" template exposes the current availability of
                computing resources by a
214             // resource provider.
215             //
216             System.out.println( );
217             //System.out.println(" 13) " + template.xmlText() );   It's equal to
                "negotiationTemplate.getXMLObject().xmlText()"
218             SampleAgreementTemplate negotiationTemplate = new SampleAgreementTemplate(
                template );
219             if ( LOG.isTraceEnabled() )
220             {
221                  System.out.println(" 12b) negotiation-template: " + negotiationTemplate
                    .getXMLObject().getName() );
222                  LOG.trace( "negotiation-template: " + negotiationTemplate.getXMLObject
                    ().xmlText() );
223                  System.out.println("\n negotiation-template: " + negotiationTemplate.
                    getXMLObject().xmlText() );
224                  LOG.trace(" *** ");
225                  System.out.println(" *** ");
226                  LOG.trace( "negotiation-template: " + negotiationTemplate.getXMLObject
                    ().getName() );
227             }else{
```

```
228                 System.out.println(" *** 13)a NO LOG *** ");
229             }
230 
231             /*****************************************************************
232             * First round of the Negotiation
233             *****************************************************************/
234             //
235             // This template will include the service description term 'RESOURCE_SDT'
                with a JSDL document
236             // describing the available compute resources.
237             //
238             // NegotiationOffer_1: 05 resources for 15 minutes duration with a time
                frame as
239             // startTime = current, endTime = startTime + 60
240             //
241             NegotiationOfferType counterOffer1 = negotiateRound1( negotiation,
                negotiationTemplate );
242 
243             /*****************************************************************
244             * Second round of the Negotiation
245             *****************************************************************/
246             //
247             // we receive 2 counter offers with different resource availability as
248             //
249             // CounterOffer_1: 05 resources for 20 minutes duration with a time fame as
250             // startTime = current + 5, endTime = startTime + 20
251             //
252             // CounterOffer_2: 05 resources for 15 minutes duration with a time fame as
253             // startTime = current + 10, endTime = startTime + 30
254             //
255             // Say that both counter offers are not sufficient with respect to time,
256             // we create another negotiation offer with
257             //
258             // NegotiationOffer_2: 05 resources for 15 minutes duration with a time
                frame as
259             // startTime = current + 10, endTime = startTime + 20
260             //
261             NegotiationOfferType selectedCounterOffer = negotiateRound2( negotiation,
                counterOffer1 );
262 
263             //
264             // create agreement if negotiated offer satisfy the requirements
265             //
266             AgreementOffer offer = new AgreementOfferType( selectedCounterOffer );
267             System.out.println(" EPR in client \n" + offer.getInitiatorEPR());
268 
269             AgreementClient agreement = getFactoryClient().createAgreement( offer );
270             Assert.assertNotNull( agreement );
271             LOG.info( "negotiated agreement successfully created" );
272 
273             //
274             // After agreement creation, current states of service description term(s)
                can be queried as shown below:
```

```
275             //
276
277             ServiceTermStateType[] states = agreement.getServiceTermStates();
278             System.out.println(" - STATES LENGTH ------------------------- ");
279             System.out.println( states.length );
280
281             ServiceTermStateType resourceServiceTerm  = agreement.getServiceTermState(
                "RESOURCE_STD");
282             System.out.println(" - RESOURCE SERVICE TERM Description -----------------
                RESOURCE_STD");
283             System.out.println( resourceServiceTerm.xmlText() );
284
285             XmlObject[] jobDefinition = resourceServiceTerm.selectChildren(
                JobDefinitionDocument.type.getDocumentElementName());
286
287             System.out.println(" - JOB DEFINITION LENGTH ------------------------- ");
288             System.out.println( jobDefinition.length );
289             while(jobDefinition.length == 0){
290                 System.out.print(".");
291                 states = agreement.getServiceTermStates();
292                 resourceServiceTerm  = agreement.getServiceTermState("RESOURCE_STD");
293                 jobDefinition = resourceServiceTerm.selectChildren(JobDefinitionDocument
                    .type.getDocumentElementName());
294             }
295             System.out.println(" - JOB DEFINITION LENGTH ------------------------- ");
296             System.out.println( jobDefinition.length );
297             System.out.println(" - RESOURCE SERVICE TERM Description -----------------
                RESOURCE_STD");
298             System.out.println( resourceServiceTerm.xmlText() );
299
300             JobDefinitionType jobDef = (JobDefinitionType) jobDefinition[0];
301             ResourcesType resources = jobDef.getJobDescription().getResources();
302             System.out.println( " - RESOURCES ---------------------- " );
303             System.out.println( resources.xmlText() );
304
305             //TermTreeType TermsAgreementInstance  = agreement.getTerms();
306             //System.out.println( " - TERMS Agreement Instance ---------------------- "
                );
307             //System.out.println( TermsAgreementInstance.toString() );
308
309             System.out.println( " - HOST NAME ---------------------- " );
310             String reservedHostName = resources.getCandidateHosts().getHostNameArray(0);
311             System.out.println( reservedHostName );
312             //
313             //To access the Guarantee Term States, the getGuaranteeTermStates() method
                can be used.
314             //
315             GuaranteeTermStateType[] guaranteeTermStates = agreement.
                getGuaranteeTermStates();
316             System.out.println( " - GUARANTEE TERMS STATES ---------------------- " );
317             System.out.println( guaranteeTermStates.length);
318             //
319             // To get the status of an agreement:
```

```
320             //
321             AgreementStateType state = agreement.getState();
322 
323             System.out.println( " ---------------------- " );
324             System.out.println( state.toString() );
325 
326             //
327             // finally terminate the negotiation process
328             //
329             LOG.info( "terminating negotiated agreement" );
330             negotiation.terminate();
331 
332         }
333         catch ( NegotiationException e )
334         {
335             Assert.fail( "NegotiationException: " + e.getMessage() );
336         }
337         catch ( ResourceUnavailableException e )
338         {
339             Assert.fail( "ResourceUnavailableException: " + e.getMessage() );
340         }
341         catch ( ResourceUnknownException e )
342         {
343             Assert.fail( "ResourceUnknownException: " + e.getMessage() );
344         }
345         catch (Exception ex) {
346             // catching error
347             ex.printStackTrace();
348         }
349 
350     }
351 
352     private static AgreementFactoryClient getFactoryClient() throws
        ResourceUnknownException, ResourceUnavailableException
353     {
354         AgreementFactoryClient factory=null;
355         //
356         // First all agreement factories are loaded and then an agreement factory having
357         // "SAMPLE-INSTANCE-1" resource id is selected.
358         //
359 
360         AgreementFactoryClient[] factories = getAgreementFactoryClients();
361         System.out.println(" factory.length: " + factories.length);
362         //Assert.assertEquals( 3, factories.length ); /* It is 3 because we have
            SAMPLE-INSTANCE-1, SAMPLE-INSTANCE-1, ISQoS_TEST
363         LOG.info( "factories: " + factories.length );
364 
365         /*
366         if ( factories[0].getResourceId().equals( "CARMEN-INSTANCE-1" ) )   //
            "SAMPLE-INSTANCE-1"  The same in line 114
367         {
368             System.out.println(" *** This is   CARMEN-INSTANCE-1 ");
369             factory = getAgreementFactoryClients()[0];
```

```
370             }
371             else
372             {
373                 System.out.println(" *** This is NOT  CARMEN-INSTANCE-1 ");
374                 factory = getAgreementFactoryClients()[1];
375             }
376             */
377             for( int i=0; i < factories.length; i++)
378             {
379                 if( factories[i].getResourceId().equals( "CARMEN-INSTANCE-1" ) )
380                 {
381                     factory = getAgreementFactoryClients()[i];
382                 }
383                 System.out.println(" *** factories[i].getResourceId() " + factories[i].
                    getResourceId());
384             }
385             return factory;
386         }
387 
388         /**
389          *
390         * @return the agreement factory clients
391         */
392         public static AgreementFactoryClient[] getAgreementFactoryClients()
393         {
394             AgreementFactoryClient[] factories = null;
395             try
396             {
397                 LoginContext loginContext = getLoginContext();
398 
399                 // START SNIPPET: ListAgreementFactories
400                 EndpointReferenceType epr = EndpointReferenceType.Factory.newInstance();
401                 epr.addNewAddress().setStringValue( "
                    http://127.0.0.1:8080/wsag4j-agreement-factory-1.0.2" );
402 
403                 AgreementFactoryRegistryClient registry = AgreementFactoryRegistryLocator.
                    getFactoryRegistry( epr, loginContext );
404                 factories = registry.listAgreementFactories();
405                 // END SNIPPET: ListAgreementFactories
406             }
407             catch ( Exception ex )
408             {
409                 LOG.error( ex );
410             }
411         return factories;
412         }
413 
414 
415         private static NegotiationOfferType negotiateRound2( NegotiationClient negotiation,
        NegotiationOfferType counterOffer1 ) throws Exception
416         {
417         /*****************************************************************
418         * Second round of the Negotiation
```

```
419         ************************************************************/
420         //
421         // we receive 2 counter offers with different resource availability as
422         //
423         // CounterOffer_1: 05 resources for 20 minutes duration with a time fame as
424         // startTime = current + 5, endTime = startTime + 20
425         //
426         // CounterOffer_2: 05 resources for 15 minutes duration with a time fame as
427         // startTime = current + 10, endTime = startTime + 30
428         //
429         // Say that both counter offers are not sufficient with respect to time,
430         // we create another negotiation offer with
431         //
432         // NegotiationOffer_2: 05 resources for 15 minutes duration with a time frame as
433         // startTime = current + 10, endTime = startTime + 20
434         //
435             SampleNegotiationOffer negotiationOffer2 = new SampleNegotiationOffer(
                counterOffer1 );
436
437             ResourcesType jobResources2 = negotiationOffer2.getResourceDefinition();
438
439             RangeValueType totalCountRange2 = RangeValueType.Factory.newInstance();
440             totalCountRange2.addNewExact().setDoubleValue( 5 );
441             jobResources2.setTotalResourceCount( totalCountRange2 );
442
443             TimeConstraintType timeConstraint2 = negotiationOffer2.getTimeConstraint();
444
445             Calendar startTime2 = (Calendar) timeConstraint2.getStartTime().clone();
446             startTime2.add( Calendar.MINUTE, 10 );
447             Calendar endTime2 = (Calendar) startTime2.clone();
448             endTime2.add( Calendar.MINUTE, 20 );
449             timeConstraint2.setStartTime( startTime2 );
450             timeConstraint2.setEndTime( endTime2 );
451             timeConstraint2.setDuration( 15 );
452
453             setResourcesSDT( negotiationOffer2, jobResources2 );
454             setTimeConstraintSDT( negotiationOffer2, timeConstraint2 );
455
456             NegotiationOfferType[] negotiationOfferTypes2 = { negotiationOffer2.getXMLObject
                () };
457             if ( LOG.isTraceEnabled() )
458             {
459                 for ( int i = 0; i < negotiationOfferTypes2.length; i++ )
460                 {
461                     LOG.trace( "Iteration-2: negotiation offers: " + negotiationOfferTypes2[
                        i].toString() );
462                 }
463             }
464
465             NegotiationOfferType[] counterOffers2 = negotiation.negotiate(
                negotiationOfferTypes2 );
466             Assert.assertNotNull( counterOffers2 );
467             Assert.assertEquals( 1, counterOffers2.length );
```

```
468
469             System.out.println();
470             LOG.info( "Iteration-2: Number of counter offers received: " + counterOffers2.
                length );
471
472             if ( LOG.isTraceEnabled() )
473             {
474                 for ( int i = 0; i < counterOffers2.length; i++ )
475                 {
476                     System.out.println();
477                     LOG.info( "Iteration: Number of counter offer received " + i );
478                     System.out.println();
479                     LOG.trace( "Iteration-2: counter_offer: " + counterOffers2[i].xmlText()
                        );
480                 }
481             }
482
483             LOG.info( "second iteration of negotiation is successful" );
484
485             NegotiationOfferType selectedCounterOffer = counterOffers2[0];
486
487             //
488             // check whether the negotiation (counter) offer is rejected
489             //
490             if ( selectedCounterOffer.getNegotiationOfferContext().getState().isSetRejected
                () )
491             {
492                 String message = "Iteration-2: counter offer [" + selectedCounterOffer.
                    getOfferId()
493                                 + "] is rejected. Reason: "
494                                 + selectedCounterOffer.getNegotiationOfferContext().getState
                                    ().xmlText();
495                 LOG.error( message );
496                 Assert.fail( "Iteration-2: NegotiationException: " + message );
497             }
498
499
500             return selectedCounterOffer;
501         }
502
503         private static NegotiationOfferType negotiateRound1( NegotiationClient negotiation,
            SampleAgreementTemplate negotiationTemplate ) throws Exception
504         {
505             /*****************************************************************
506             * First round of the Negotiation
507             *****************************************************************/
508             //
509             // This template includes the service description term 'RESOURCE_SDT' with a
                JSDL document
510             // describing the available compute resources.
511             //
512             // NegotiationOffer_1: 05 resources for 15 minutes duration with a time frame as
513             // startTime = current, endTime = startTime + 60
```

```
514         //
515         //
516         String offerID = negotiationTemplate.getContext().getTemplateId() + "-" +
            negotiationTemplate.getName();
517         System.out.println();
518         System.out.println(" 14) " + offerID);
519         //String offerID2 = negotiationTemplate.getTemplateId() + "-" +
            negotiationTemplate.getName();
520         //System.out.println(" 14a) " + offerID2); // This seems to be equal to 14)
521         ResourcesType jobResources1 = negotiationTemplate.getResourceDefinition();
522         System.out.println();
523         System.out.println(" 14a) \n" + jobResources1);
524
525         RangeValueType totalCountRange1 = RangeValueType.Factory.newInstance();
526         totalCountRange1.addNewExact().setDoubleValue( 5 );
527         jobResources1.setTotalResourceCount( totalCountRange1 );
528         System.out.println();
529         System.out.println(" 14b) \n" + jobResources1);
530
531         //
532         // The service description term 'TIME_CONSTRAINT_SDT' defines the time frame
533         // during which the resources can be available.
534         // The start and end time specified by a user is considered to be the earliest
535         // possible start time and the deadline.
536         // We need an advance reservation of the resources for 15 minutes effective
            duration,
537         // however, within a same time frame as received from a resource provider
538         //
539         TimeConstraintType timeConstraint1 = negotiationTemplate.getTimeConstraint();
540         System.out.println();
541         System.out.println(" 14c) \n" + timeConstraint1);
542
543         Calendar startTime1 = (Calendar) timeConstraint1.getStartTime().clone();
544         Calendar endTime1 = (Calendar) timeConstraint1.getEndTime().clone();
545
546         timeConstraint1.setStartTime( startTime1 );
547         timeConstraint1.setEndTime( endTime1 );
548         timeConstraint1.setDuration( 15 );
549
550         System.out.println();
551         System.out.println(" 14d) \n" + timeConstraint1);
552         //
553         // now we create negotiation offer from a negotiable template for the
            reservation of
554         // required resources to fulfil the application execution requirements.
555         //
556         SampleNegotiationOffer negotiationOffer1 = negotiationTemplate.
            getNegotiationOffer(); // Taking Context and Terms from "negotiationTemplate"
557         negotiationOffer1.setOfferId( offerID );
558         System.out.println();
559         System.out.println(" 14f) \n" + negotiationOffer1.getXMLObject().xmlText());
560         System.out.println(" 14f) \n" + negotiationOffer1.getXMLObject().toString());
561         //
```

```
562         // creating negotiation offer context
563         //
564         NegotiationOfferContextType negOfferContext = NegotiationOfferContextType.
            Factory.newInstance();
565         negOfferContext.setCreator( NegotiationRoleType.NEGOTIATION_INITIATOR );
566         GregorianCalendar expireDate = new GregorianCalendar();
567         expireDate.add( Calendar.MINUTE, 5 );
568         negOfferContext.setExpirationTime( expireDate );
569         System.out.println();
570         System.out.println(" 14g) \n" + negOfferContext);
571
572         NegotiationOfferStateType negOfferState = NegotiationOfferStateType.Factory.
            newInstance();
573         negOfferState.addNewAdvisory();
574         negOfferContext.setState( negOfferState );
575         System.out.println();
576         System.out.println(" 14h) \n" + negOfferContext);
577
578         negOfferContext.setCounterOfferTo( offerID ); // a fully qualified name of the
            template
579                                                       // (templateID-TemplateName)
580         System.out.println();
581         System.out.println(" 14i) \n" + negOfferContext);
582
583         negotiationOffer1.setNegotiationOfferContext( negOfferContext );
584         System.out.println();
585         //System.out.println(" 14j) \n" + negotiationOffer1.getXMLObject().xmlText());
586         System.out.println(" 14j) \n" + negotiationOffer1.getXMLObject().toString());
587
588         //
589         // one negotiation constraint is added that basically define the preferred time
            frame window within
590         // which the user would like to have the reservation of resources (say within
            first 45 minutes)
591         //
592         NegotiationConstraintSectionType constraints1 = addNeogtiationOfferConstraints(
            timeConstraint1.getStartTime() );
593         negotiationOffer1.setNegotiationConstraints( constraints1 );
594         System.out.println();
595         System.out.println(" 14m) \n" + negotiationOffer1.getXMLObject().toString());
596         //
597         // SDT Service Description Terms 'RESOURCE_SDT' and 'TIME_CONSTRAINT_SDT' are
            updated in a negotiation offer.
598         //
599         setResourcesSDT( negotiationOffer1, jobResources1 );
600         setTimeConstraintSDT( negotiationOffer1, timeConstraint1 );
601
602         NegotiationOfferType[] negotiationOfferTypes1 = { negotiationOffer1.getXMLObject
            () };
603
604         System.out.println();
605         if ( LOG.isTraceEnabled() )
606         {
```

```
607             for ( int i = 0; i < negotiationOfferTypes1.length; i++ )
608             {
609                 LOG.trace( "Iteration-1: negotiation offers: " + negotiationOfferTypes1[
                    i].toString() );
610             }
611         }
612
613         //
614         // invoking negotiate method from Negotiation instance and
615         // in return counter offers are received
616         //
617         NegotiationOfferType[] counterOffers1 = negotiation.negotiate(
            negotiationOfferTypes1 );
618         //assertNotNull( counterOffers1 );
619         //assertEquals( 2, counterOffers1.length );
620
621         System.out.println();
622         LOG.info( "Iteration-1: Number of counter offers received: " + counterOffers1.
            length );
623
624         if ( LOG.isTraceEnabled() )
625         {
626             for ( int i = 0; i < counterOffers1.length; i++ )
627             {
628                 System.out.println();
629                 System.out.println(" >>> Counter Offer Number: " + i);
630                 System.out.println();
631                 LOG.trace( "Iteration-1: counter_offer: " + counterOffers1[i].xmlText()
                    );
632             }
633         }
634
635         //
636         // first check whether the negotiation (counter) offer is rejected
637         //
638         NegotiationOfferType counterOffer1 = counterOffers1[0];
639         if ( counterOffer1.getNegotiationOfferContext().getState().isSetRejected() )
640         {
641             String message = "Iteration-1: counter offer [" + counterOffer1.getOfferId()
                 + "] is rejected. Reason: "
642                             + counterOffer1.getNegotiationOfferContext().getState().
                            xmlText();
643             LOG.error( message );
644             System.out.println();
645             System.out.println(" Error: " + message);
646             //fail( "Iteration-1: NegotiationException: " + message );
647         }
648
649         System.out.println();
650         System.out.println(" counterOffer1: " + counterOffer1.xmlText());
651         System.out.println();
652         LOG.info( "first iteration of negotiation is successful" );
653
```

```
654             return counterOffer1;
655         }
656 
657         private static void setTimeConstraintSDT( SampleNegotiationOffer negotiationOffer,
            TimeConstraintType timeConstraint )
658         {
659 
660             ServiceDescriptionTermType timeConstraintSDT = null;
661 
662             ServiceDescriptionTermType[] sdts =
663             negotiationOffer.getTerms().getAll().getServiceDescriptionTermArray();
664 
665             if ( sdts != null )
666             {
667                 for ( int i = 0; i < sdts.length; i++ )
668                 {
669                     if ( sdts[i].getName().equals( "TIME_CONSTRAINT_SDT" ) )
670                     {
671                         timeConstraintSDT = sdts[i];
672                         break;
673                     }
674                 }
675             }
676 
677             String name = timeConstraintSDT.getName();
678             String serviceName = timeConstraintSDT.getServiceName();
679 
680             TimeConstraintDocument timeConstraintDoc = TimeConstraintDocument.Factory.
            newInstance();
681             timeConstraintDoc.addNewTimeConstraint();
682             timeConstraintDoc.getTimeConstraint().set( timeConstraint );
683 
684             timeConstraintSDT.set( timeConstraintDoc );
685             timeConstraintSDT.setName( name );
686             timeConstraintSDT.setServiceName( serviceName );
687 
688             System.out.println();
689             System.out.println(" 14r) timeConstraintSDT: \n" + timeConstraintSDT.xmlText());
690         }
691 
692 
693         private static void setResourcesSDT( SampleNegotiationOffer negotiationOffer,
            ResourcesType jobResources ) throws Exception
694         {
695             ServiceDescriptionTermType resourcesSDT = null;
696             ServiceDescriptionTermType[] sdts = negotiationOffer.getTerms().getAll().
            getServiceDescriptionTermArray();
697             if ( sdts != null )
698             {
699                 for ( int i = 0; i < sdts.length; i++ )
700                 {
701                     System.out.println();
702                     System.out.println(" 14n) \n" + sdts[i].getName());
```

```
703                 if ( sdts[i].getName().equals( "RESOURCE_STD" ) )
704                 {
705                     resourcesSDT = sdts[i];
706                     break;
707                 }
708             }
709         }
710 
711         System.out.println();
712         System.out.println(" 14n)a) resourcesSDT: \n" + resourcesSDT.xmlText());
713 
714         String name = resourcesSDT.getName();
715         String serviceName = resourcesSDT.getServiceName();
716         System.out.println();
717         System.out.println(" 14o) name: " + name + " - serviceName: " + serviceName);
718 
719         JobDefinitionDocument resourcesDoc = JobDefinitionDocument.Factory.newInstance();
720         resourcesDoc.addNewJobDefinition().addNewJobDescription().addNewResources();
721         resourcesDoc.getJobDefinition().getJobDescription().getResources().set(
          jobResources );
722 
723         System.out.println();
724         System.out.println(" 14p) \n" + resourcesDoc.toString());
725 
726         resourcesSDT.set( resourcesDoc );
727         resourcesSDT.setName( name );
728         resourcesSDT.setServiceName( serviceName );
729 
730         System.out.println();
731         System.out.println(" 14q) resourcesSDT: \n" + resourcesSDT.xmlText());
732 
733     }
734 
735     private static NegotiationConstraintSectionType addNeogtiationOfferConstraints(
      Calendar startTime )
736     {
737 
738         final String constraintItemName = "TimeConstraintSDT_TimeConstraint_START_TIME";
739         final String constraintXpath =
740                 "declare namespace wsag-tc='
                  http://schemas.wsag4j.org/2009/07/wsag4j-scheduling-extensions';"
741               + "declare namespace wsag='
                  http://schemas.ggf.org/graap/2007/03/ws-agreement';"
742               + "$this/wsag:Terms/wsag:All/wsag:ServiceDescriptionTerm[@wsag:Name =
                  'TIME_CONSTRAINT_SDT']"
743               + "/wsag4jt:TimeConstraint";
744 
745         NegotiationConstraintSectionType constraints = NegotiationConstraintSectionType.
          Factory.newInstance();
746 
747         // add one item constraint
748         NegotiationOfferItemType offerItem = constraints.addNewItem();
749         offerItem.setName( constraintItemName );
```

```
750             offerItem.setLocation( constraintXpath );
751             System.out.println();
752             System.out.println(" 14k) \n" + offerItem.toString());
753 
754             ItemConstraint constraint = offerItem.addNewItemConstraint();
755             constraint.addNewMinInclusive().setValue( XmlDateTime.Factory.newValue(
                startTime ) );
756             Calendar preferredEndTime = (Calendar) startTime.clone();
757             preferredEndTime.add( Calendar.MINUTE, 20 );
758             constraint.addNewMaxInclusive().setValue( XmlDateTime.Factory.newValue(
                preferredEndTime ) );
759             System.out.println();
760             System.out.println(" 14l) \n" + constraint.xmlText());
761 
762             return constraints;
763         }
764 
765 
766         private static NegotiationClient initiateNegotiation(NegotiationClient negotiation,
                AgreementFactoryClient factory)
767         {
768             try
769             {
770                 // Now creates a negotiation context that defines the roles and obligations
771                 // of the negotiating parties and specifies the type of the negotiation
                    process.
772                 //
773                 NegotiationContextDocument negContextDoc = NegotiationContextDocument.
                    Factory.newInstance();
774                 System.out.println(" 6) " + negContextDoc.xmlText());
775                 NegotiationContextType negContext = negContextDoc.addNewNegotiationContext();
776                 negContext.setAgreementFactoryEPR( factory.getEndpoint() );
777                 System.out.println(" 7) " + negContext.xmlText() );
778                 negContext.setAgreementResponder( NegotiationRoleType.NEGOTIATION_RESPONDER);
779                 System.out.println(" 8) " + negContext.xmlText() );
780 
781                 GregorianCalendar expireDate = new GregorianCalendar();
782                 expireDate.add( Calendar.HOUR, 1 );
783                 negContext.setExpirationTime( expireDate );
784                 System.out.println( );
785                 System.out.println(" 9) " + negContext.xmlText() );
786 
787                 //
788                 // set the nature of the negotiation process (e.g. negotiation or
                    re-negotiation).
789                 //
790                 NegotiationType negotiationType = negContext.addNewNegotiationType();
791                 negotiationType.addNewNegotiation();  // <--- It seems I'm not using
                    negotiationType
792                 // negotiationType.addNewRenegotiation();   <---- RE-negotiation
793                 System.out.println( );
794                 System.out.println(" 10) " + negContext.xmlText() );
795 
```

```
796                 //
797                 // creating negotiation instance based on a negotiation context from a
                    selected agreement factory
798                 //
799                 negotiation = factory.initiateNegotiation( negContext );
800                 System.out.println( );
801                 System.out.println(" 11) " + negotiation.getNegotiationContext().xmlText() );
802                 System.out.println( );
803                 System.out.println(" 11a) *** Negotiation instance is created successfully
                    ***" );
804     /*      }
805             catch ( NegotiationFactoryException e )
806             {
807                 System.out.println( "NegotiationFactoryException: " + e.getMessage() );
808             }
809             catch ( ResourceUnavailableException e )
810             {
811                 System.out.println( "ResourceUnavailableException: " + e.getMessage() );
812             }
813             catch ( ResourceUnknownException e )
814             {
815                 System.out.println( "ResourceUnknownException: " + e.getMessage() );
816     */      }
817             catch ( Exception e )
818             {
819                 System.out.println( "Could not create negotiation client instance. Error: "
                    + e.getMessage() );
820             }
821
822             return negotiation;
823         }
824
825
826
827         /**
828          * A client must provide a set of security credentials.
829          * The easiest way to provide these credentials is to provide a valid LoginContext
           as shown below:
830          *
831          * @return LoginContext
832          * @throws LoginException
833          */
834         private static LoginContext getLoginContext() throws LoginException {
835             /*
836             * create a keystore login context
837             */
838           System.out.println("Beginning");
839           KeystoreProperties properties = new KeystoreProperties();
840           properties.setKeyStoreAlias("wsag4j-user");
841           properties.setPrivateKeyPassword("user@wsag4j");
842
843           System.out.println(" 1.a)");
844           properties.setKeyStoreType("JKS");
```

```
845         properties.setKeystoreFilename("/wsag4j-client-keystore.jks");
846         properties.setKeystorePassword("user@wsag4j");
847 
848         System.out.println(" 1.b)");
849         properties.setTruststoreType("JKS");
850         properties.setTruststoreFilename("/wsag4j-client-keystore.jks");
851         properties.setTruststorePassword("user@wsag4j");
852 
853         System.out.println(" 1.c)");
854         LoginContext loginContext = new KeystoreLoginContext(properties);
855         System.out.println(" 1.d)");
856         loginContext.login();
857         System.out.println(" 1.e)");
858 
859         return loginContext;
860     }
861 
862 }
863 
```