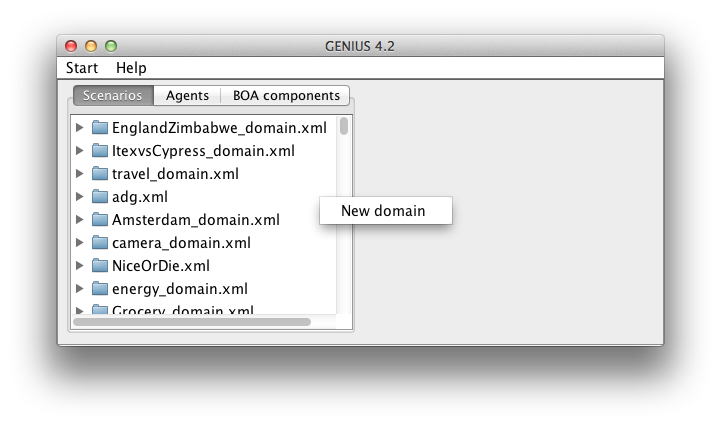
**Quick Start Tutorial: How to create a Domain**

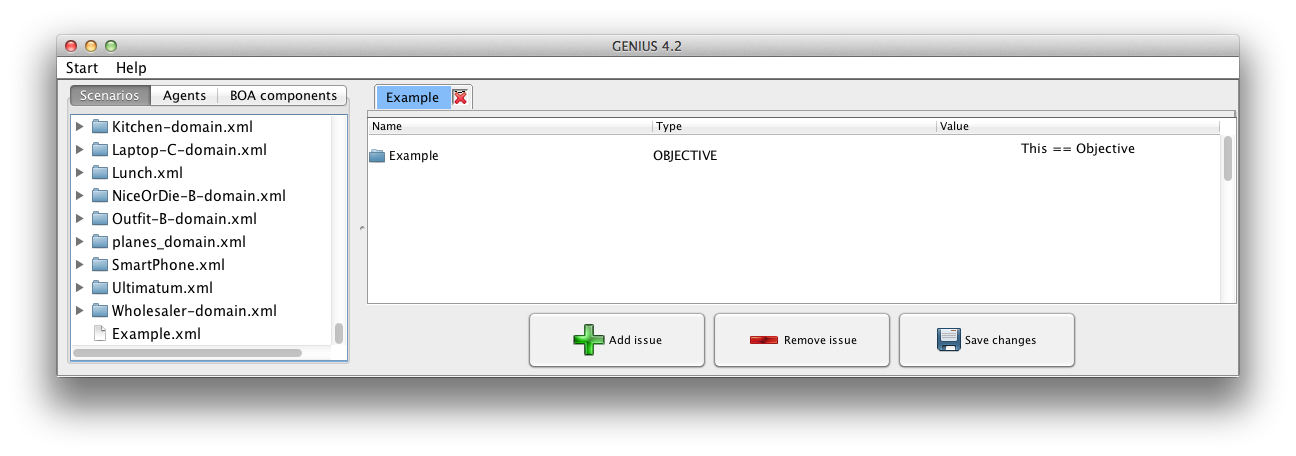
In this tutorial you will learn how to create a domain as part of creating a negotiation scenario.

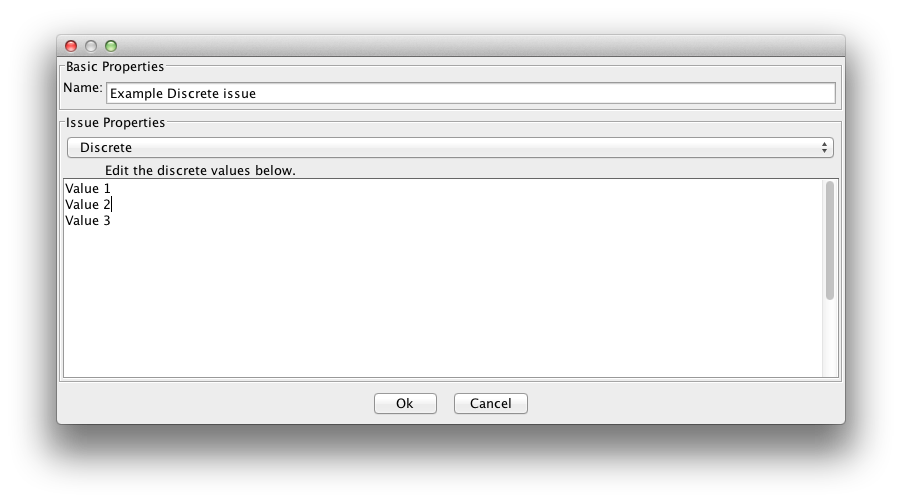
Before running a negotiation in GENIUS, a scenario must be created. A scenario consists of a domain and a set of preference profiles (follow the tutorial “How to create a Preference Profile”). A domain consists of a set of issues.

To create a domain, **right click** on the list of available scenarios in the **Components Window**. A popup menu with the option to create a new domain will be displayed as shown in the figure below:



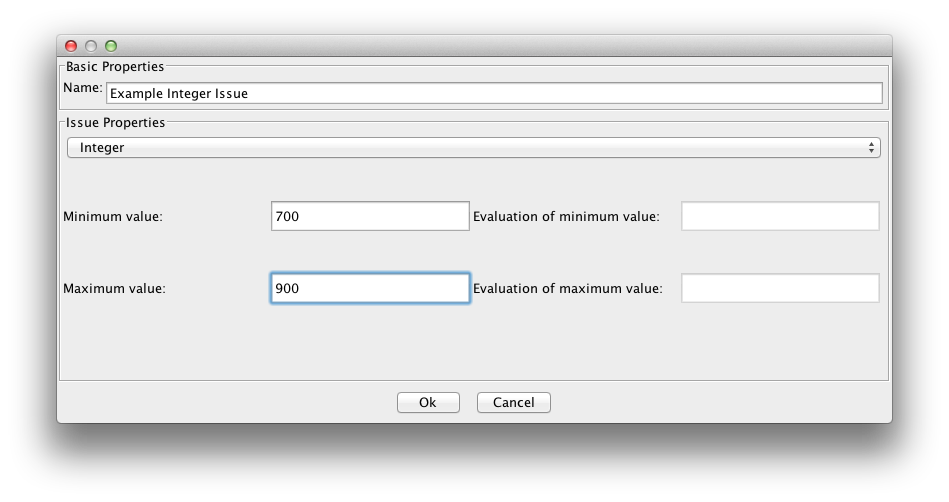
After selecting “**New domain**,” you will be prompted to enter a domain name. Enter a name and select “**Ok**.” A domain will be created and added at the bottom of the “Scenarios” tab in the Components Window. The domain will be displayed in a window similar to the figure below.



Next, issues must be added to the domain. Select the “**Add issue**” button at the bottom of the window (shown circled in the figure above). A dialog box similar to the figures below below will appear. Enter the issue name. Under “Issue Properties” you can choose to create a **Discrete** or an **Integer** issue.

For a **Discrete issue**, enter each

issue value on a separate line.



For an **Integer issue**, enter the minimum

and maximum values for the issue. During

a negotiation, a bid can offer any value

within the specified range for the issue.

The evaluation values will be entered at

a later time.

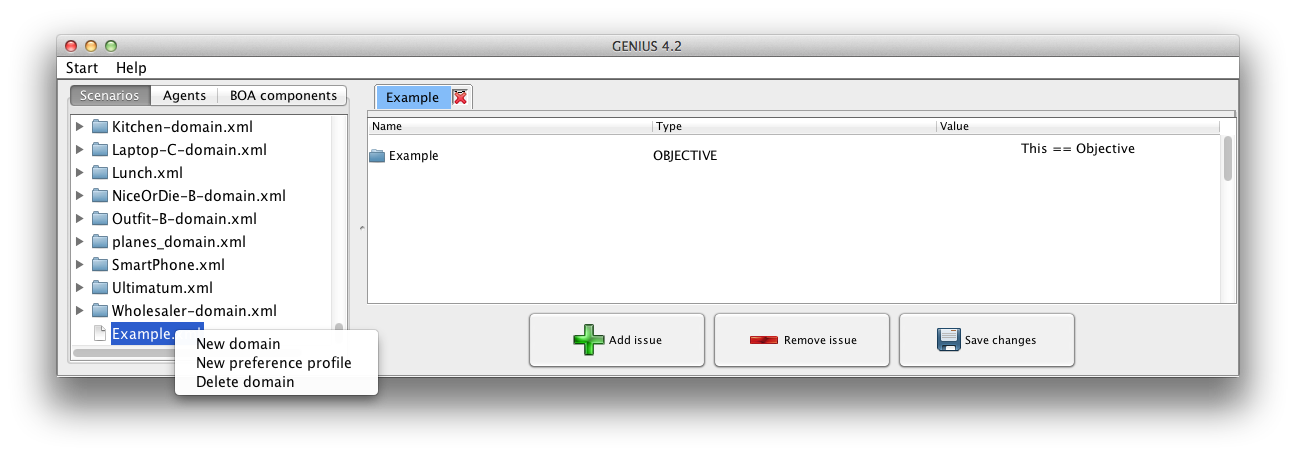
Make sure to save the issues in the domain by selecting “Save changes” before creating preference profiles.

**2.2.2. Creating a Preference Profile:**

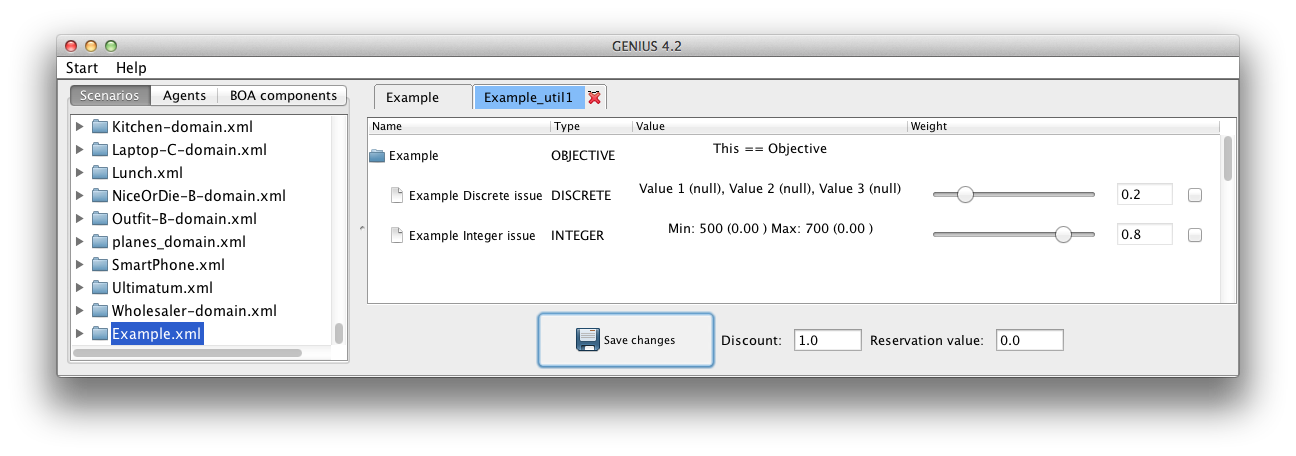
A preference profile specifies the following:

* **Weight** (importance) of the issues in the domain
* **Reservation value** and **discount**
* **Evaluation** (preference) of the values of the issues

To create a preference profile, **right click** on the domain in the “Scenarios” tab of the Components Window. Select “**New preference profile**” from the popup menu as shown in the figure \_\_\_.



A utility space window will open similar to figure \_\_\_ . Set the **weights of the issues** by moving the sliders next to each issue. Note, when you move a slider, the weights of the other issues are automatically updated such that the sum of the weights is equal to one. If you do not want that the weight of an issue to automatically change select the checkbox next to the slider.

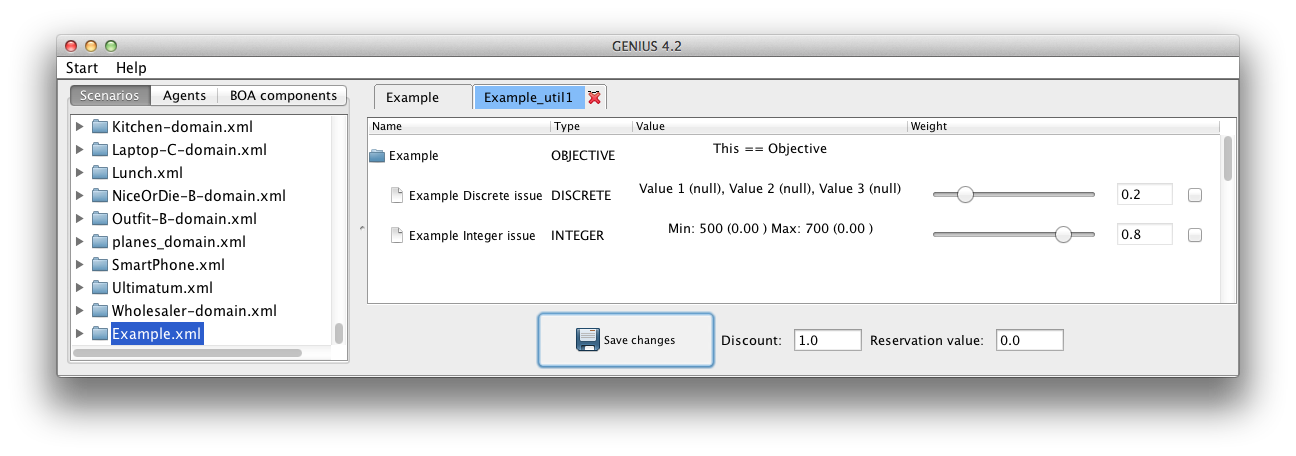
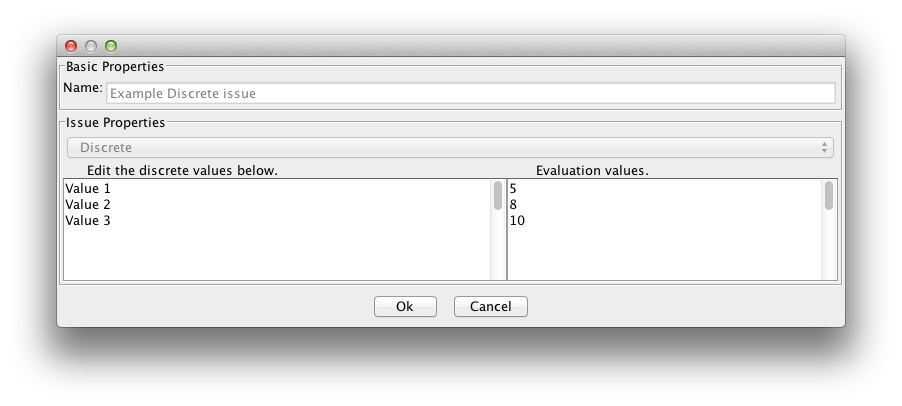
A **discount rate** and **reservation value** can be entered at this window as well. 

Set Discount and Reservation value

Set the weights of the issues

Select “**Save changes**.” The weights of the issues must be saved before the evaluation values of the issues can be set.

To set the **evaluation of the issues**, double click on the issue name in the utility space. See figure \_\_\_ below.



For a **discrete issue**, specify the evaluation

value of each discrete value. Evaluation values

must be non-zero integers. During the negotiation

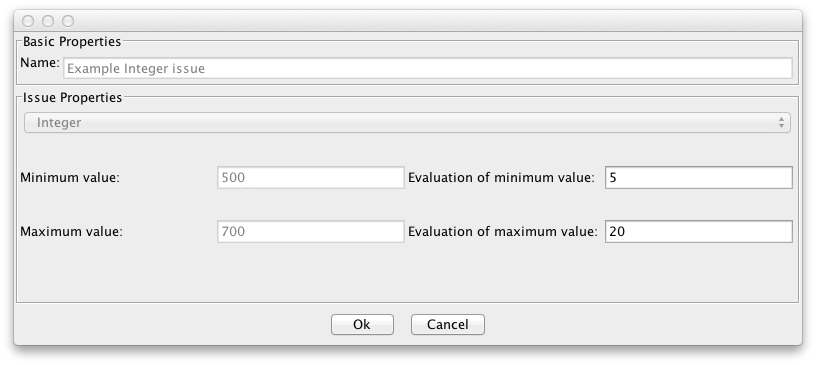
the **utility** of an issue value is determined by dividing

the evaluation value by the highest evaluation

value for that particular issue.

(For example, in figure \_\_\_ the utilities of

Values 1, 2 and 3 are respectively 0.5, 0.8, and 1.0)



For an **integer issue**, specify the utility of the

lowest possible value and the highest possible value.

The utility of a value in this range is calculated

during the negotiation using linear interpolation.

Make sure to select “**Ok**” to save the evaluation values.

Finally, select “Save changes” in the “Utility space window.”

You can create additional preference profiles for the domain in order to run a negotiation.

**3. How to create a Negotiation Agent:**

There are a number of example negotiation agents provided in the zipfile. If you would like to run a negotiation using one of the example agents provided, skip to section **How to run a negotiation** (link provided). If you would like to create your own agent, follow this tutorial.

**3.1 How to create a basic Negotiation Agent**

A standard negotiation agent implements an agent as a single block of logic: a mix of a bidding strategy, acceptance strategy, and possibly an opponent model.

In contrast, we recommend to separately implement these components to create a BOA agent as discussed in the section **How to create a BOA Negotiation Agent** (link provided). The main advantage of a BOA agent is that existing components can be reused, allowing for easier agent development.

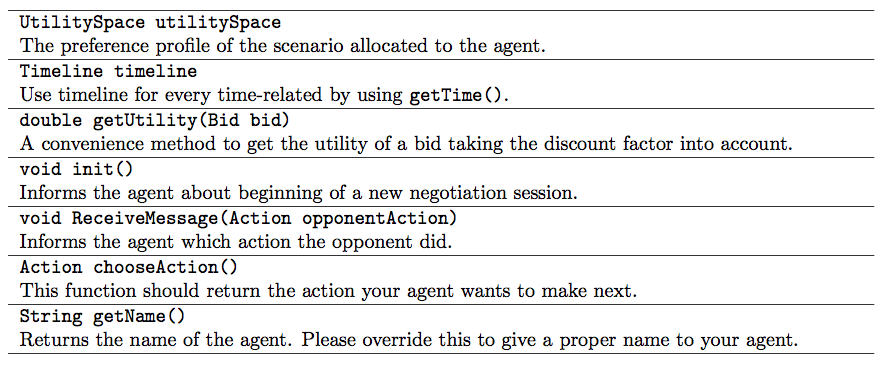
In this section we assume that you are familiar with programming in Java. In case you are not familiar with Java, please consult the following tutorial: http://www.oracle.com/technetwork/java/javase/documentation/index.html (updated link). The Java API definitions can be found here as well.

In a Java editor, such as Eclipse or Netbeans, create a new Java project.

Add **negotiationsimulator.jar** as an external library.

Create a new class with a distinct name (ex. YourAgent) as a subclass to extend the superclass, **negotiator.Agent**.

Table \_\_\_ shows the most important fields and methods of this class. For more information, please refer to the javadoc of Genius.



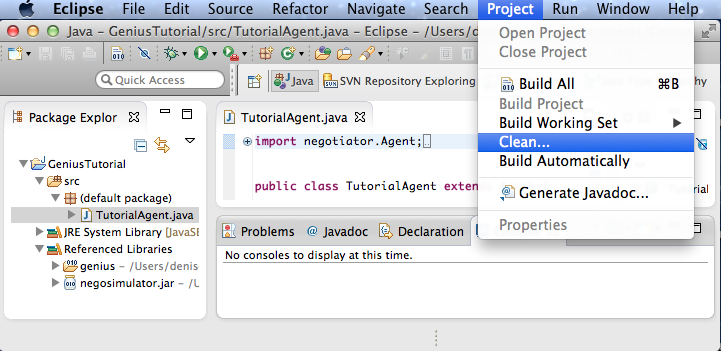
To implement your agent, you need to override the three methods:

* **ReceiveMessage**
* **init**
* **chooseAction**

An agent may consist of multiple classes as long as one class extends the negotiator.Agent class. For documentation of the data structures, refer to the Javadoc found in the download of GENIUS.

**3.1.1. How to compile an Agent**

If you are using **Eclipse**, the agent file is often compiled automatically and can be found in the “bin” folder in your workspace. If the file has not been compiled automatically, you can compile it by selecting **project🡪 Clean…** (see figure\_\_).



To manually compile the agent, first, place the agent java file (ex. TutorialAgent.java**)** in the directory containing the **negotiator.jar** file**.** Then, in the computer’s Terminal enter the command line: **javac -cp negosimulator.jar YourAgent.java**

The next step is to load the agent in Genius. Towards this end, first we should move the agent to the directory of Genius. Note that the directory in which the agent should be put depends on your project. If for example you have a project with a package “agents” in which the agent is located, then the agent should be moved in a folder “agents” in the root of your Genius installation. An agent may consist of multiple class files. Now we can add the agent in one of the following two ways:

**4. How to run a Negotiation**

There are three modes for running a negotiation:

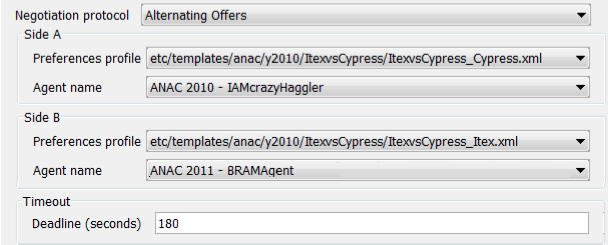
* **Negotiation session** – a single negotiation in which two agents compete. This mode is mainly intended for new users.
* **Tournament** – a collection of negotiation sessions that take place between two sets of agents competing against each other on a set of domains. The results of the sessions are stored in the “log” directory. These results can be more easily viewed by importing them into Excel and using pivot tables (link to how to view logs)
* **Distributed tournament** - a tournament which is stored in a database and can therefore be divided among multiple computers to speed up calculation.

There are two types of agents that can be used:

* **Automated agents** – agents that can compete against other agents in a negotiation without relying on input from a user. In general, these agents are able to make a large amount of bids in a limited amount of time.
* **Non-automated agents** – agents that are fully controlled by the user. These types of agents ask the user each round which action they should make. Genius by default includes the UIAgent – which has a simple user interface – and the more extensive Extended UIAgent.

**4.1. How to run a Negotiation Session**

Select “**Start**” and then “**Negotiation Session**”. This opens a window similar to Figure \_\_\_.



The following parameters need to be specified to run a negotiation:

• **Negotiation protocol** – the set of available protocols (normally “Alternating Offers” is used)

• **Side A/Side B** – the configuration of the agents on both sides

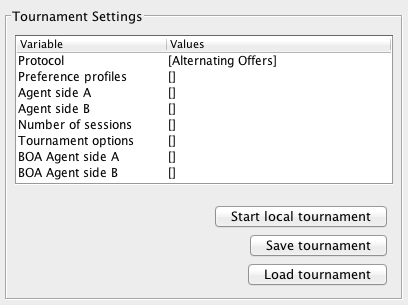
• **Preference profile** -the preference profile to be used by the agent of that side.

• **Agent name** – the agent participating in the negotiation

• **Deadline (seconds)** - the length of the negotiation in seconds

**4.2. How to run a Tournament**

Select “**Start**” and then “**Tournament**”. The Tournament tab will appear similar to Figure \_\_\_.

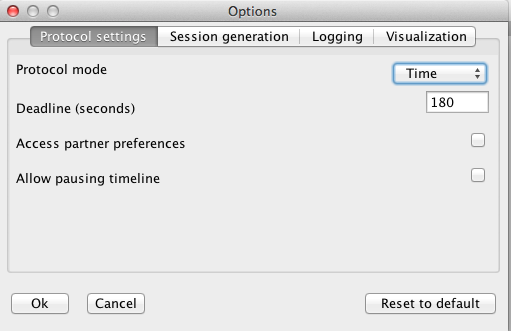


The following options need to be specified to run a tournament.

* **Protocol** – the set of available protocols
* **Preference profiles** – the set of scenarios on which the agents should compete. Each selected  scenario should feature at least two preference profiles
* **Agent side A/B** – the set of agents in set A competes against all agents in set B
* **Number of sessions** – the number of times each session should be repeated
* **Tournament options**  - specify how to run the tournament (see detailed descriptions below)
* **BOA Agent side A/B** – type of agents that consist of multiple components (link to section)

The value of an option can be specified by double clicking the option in the “**Values**” column.

**Tournament options** specify the composition of the tournament. Double clicking on “Tournament options” will open a window similar to figure \_\_\_.



There are four categories of options: protocol settings, Session generation, Logging, and Visualization.

**Protocol settings:**

* 1. **Protocol mode** – specifies if the negotiation features **rounds** or **time**.

In a **rounds**-based negotiation the deadline is specified in rounds. Time does not pass within a round so an agent can take more time to compute an action.

In a **time**-based negotiation there is an amount of time to reach an agreement. Time passes while an agent deliberates an action.

* + - 1. **Deadline** – depending on the protocol mode, rounds or time, this is the maximum amount of rounds or amount of time in seconds respectively. Note, for rounds based negotiation, one single round corresponds to one turn of a single agent.
  1. **Access partner preferences** – when enabled, allows agents to access the preference profile of the negotiation session, which contains the opponent’s preference profile.
  2. **Allow pausing timeline** – when enabled, allows agents to pause the negotiation by using the timeline.pause()  and time.resume() methods.
  3. **Session generation:**
  4. **Play both sides** – when enabled, each pair of agents should play both sides on a scenario.
  5. **Play against self** – when enabled, if an agent is included in both Agent side A and side B sets, it is allowed to play against itself. If disabled, the sessions in which agents negotiate against themselves are removed.

**Logging**:

**Log detailed analysis** – when enabled, activates a set of quality measures to capture the quality of the negotiation process. The quality measures are added to the default log. In addition, for the whole tournament an overview log is created. This log is prefixed with “TM-”.

**Visualization:**

**Show all bids** - when enabled, all bids in a scenario are visualized as red points in the negotiation status window. This option has some impact on performance.

**Show last bid** - when enabled the last bid is marked with a special symbol to make it clear which move an agent performed.

**Disable GUI** - when enabled most GUI elements are disabled. This speeds-up the negotiation up to a factor of 200 times. The progress of the tournament is printed to the console.

**4.3. How to run a Distributed Tournament**