Part 1: Introduction

Thank you for your interest in IAGO: Interactive Arbitration Guide Online.

IAGO is a platform and programming API designed to assist you in creating virtual agents that can negotiate with humans over the web.

IAGO is part of an ongoing effort to design agents that use human-like tactics in negotiation, and can eventually be used to teach negotiation skills in a simple and interactive way.

IAGO is a web-based program, and runs in a browser window on most modern browsers, including Google Chrome, Opera, Firefox, Safari, and Internet Explorer and Edge.

No installation is necessary to play against an IAGO agent, and negotiations are intended to be intuitive and fun.

For researchers, IAGO represents a tool for running real-world studies where participants, recruited through systems like Qualtrics or Amazon's Mechanical Turk, can be matched against agents designed by you.

Agent development is easy, and requires only basic understanding of the Java programming language to get started.

This tutorial is intended to familiarize you with the first steps in downloading the IAGO framework and building your first agent.

When you are ready, please continue to Part 2: Download

Part 2: Download

Downloading the IAGO zip file for developers can be done on the IAGO website by filling out the web form. Alternatively, you may access it here. Once you have downloaded and saved the zip file, please extract its contents to a new folder on your hard drive. This will be your development folder.

Inside the IAGO zip file, you will find two folders. The first folder, labeled "src", contains the source files that will help you get started using IAGO. These include two sample agents that can be used out of the box with IAGO, as well as some view files that allow you to customize the game features that define the interaction between your players.

The second folder is the WebContent folder, and contains the resources that will be deployed onto your web server when starting IAGO. Most of this folder will not need to be modified,
but you may choose to do so in order to create custom art for IAGO, or to update the IAGO core files should they become out of date.

Finally, you will find two files in the IAGO zip file. The first, "build.xml", contains an Ant build script for creating your IAGO deployment using the command line. If you do not have Ant installed, please install it by referring to the appropriate tutorial.

The second file, and the final item in the IAGO zip file is "IAGO.war". This is the fully compiled and deployable file that you will create when you are ready to deploy IAGO on a web server.
We have included this file for you, so that you can immediately test IAGO once you configure your server.

We will now discuss the requirements for third party software for programming and deploying IAGO. Continue on to Part 3: Installation Prerequisates.

Part 3: Installation Prerequisites

Java

IAGO is a Java API, so you will need to download and install a Java Software Development Kit, or JDK. Oracle provides a detailed guide for installing Java on your computer, which can be found at the link provided in this tutorial description.

https://docs.oracle.com/javase/8/docs/technotes/guides/install/install_overview.html

Please note that you may already have Java installed. The easiest way to determine this is to open up a command prompt or terminal and type in the following command:

```
java -version
```

If successful, the command should print out the version number of Java. If the command is not recognized, eithe rthe PATH variable for JAVA_HOME is not set, or Java is not installed at all. IAGO supports both Java 7 or Java 8. Please note that the JDK is required, not just the JRE. If you are confused about how to do this, refer to the tutorial on Java.

Ant

If you intend to build IAGO by hand, you will also require Apache Ant. Ant is a program that reads instructions on how to compile projects and automates this process for you. If you do not intend to manually build IAGO, you do not need Ant.

Ant comes by default on some versions of Mac OS X. To test this, you can simply type the command:

```
ant -version
```

An Ant installation tutorial is provided by Apache. The link is available here.

http://ant.apache.org/manual/install.html

However, Ant can be easily be installed on OS X using the Homebrew command:

```
brew install ant
```

Please note that you should install Java prior to installing Ant.

Eclipse

We recommend using the Eclipse Interactive Development Environment to develop your IAGO agent. This simplifies the build and deployment process to a web server considerably. However, all of these steps are possible manually if you do prefer a different IDE.

As of this writing, the current version of Eclipse is Eclipse Neon. Eclipse can be downloaded from the link below.

https://www.eclipse.org/downloads/eclipse-packages/

We recommend you download the Java EE package, although any installation will work if all additional packages are downloaded.

Once you have installed Eclipse, you will need to select a workspace. The workspace is the location on the file system where files will be stored during development. Choose an empty folder, as you will be importing IAGO into this folder shortly.

When opening Eclipse for the first time, it will ask you where you want to place the workspace (this can be changed later).

Tomcat

In order to run IAGO, the best way is to host it on a web server, such as Apache Tomcat. We recommend Tomcat 7 or 8, which can be found here:

https://tomcat.apache.org/download-70.cgi

Tomcat offers a simple, downloadable installer for Windows systems. Once Tomcat is installed, it will register as a Windows service which can be enabled or disabled from the Services menu (Start --> Search --> "Services"). Additionally, Eclipse provides support for automated launching of Tomcat servers, once configured properly.

Part 4: Testing

First, you should attempt to test your Tomcat installation, as well as get a feel for what IAGO is like when played by the end-user. To do this, you will need to locate your Tomcat directory. Simply copy the IAGO.war file included in the IAGO zip directory into the "webapps" folder of your Tomcat installation. Then, enable Tomcat from the Services menu (see part 3).

If successful, a folder called "IAGO" should appear in the Tomcat webapps directory.

You should be able to pull up IAGO in a web browser using the following URL:

http://localhost:8080/IAGO/start

If you see a page with content and not a 404: Not Found page, you have successfully installed IAGO.

Once you can access IAGO using your web browser, try playing it. The splash page for IAGO provides a sample tutorial for the game. You may also open up developer tools in most modern browsers to view JS console messages and to assist with debugging.

Note that Tomcat does by default use port 8080 as opposed to the standard HTTP port of 80. Therefore, you must include the port information in the URL unless you change Tomcat's configuriation in conf/server.xml.

You may further edit the configuration of Tomcat to your liking. For example, you may change the path at which IAGO is accesssed by adding a Context element. The recommended way to do this is to add a ROOT.xml file to the localhost directory in Tomcat, although you may instead edit server.xml directly.

Note: If you make changes to IAGO later, you will need to rebuild the .war file. To do this, simply type:

```
ant
```

from a console while in the directory containing build.xml. IAGO will build and generate a new IAGO.war file, which may be used exactly like the old one. Note that building and testing from within Eclipse will NOT create this war file.

If you do not want to bother configuring Tomcat, you may skip to Part 5.

IMPORTANT: The server.xml file in your Tomcat directory is NOT the same server.xml file found within the Server folder in Eclipse. Eclipse by default maintains a private configuration of Tomcat. You may notice that Eclipse's server.xml already contains a Context element for your new project once configured (see Part 5).

Add a file called ROOT.xml in <catalina_home>/conf/Catalina/localhost/

This ROOT.xml will override the default settings for the root context of the tomcat installation for that engine and host (Catalina and localhost).

Enter the following to the ROOT.xml file:

```
<Context
docBase="<yourApp>"
path=""
reloadable="true"
/>
```

Your application is now the default application and will show up on http://localhost:8080

Part 5: Setting up IAGO for Eclipse

If you wish to develop for IAGO within Eclipse, which will allow for real-time debugging, visible console messages, and automatic deployment, follow the instructions in this section.

After launching Eclipse, you will need to import the IAGO project. To do so, click File --> Import... --> General --> Projects from Folder or Archive. Then, click Directory, and navigate to the unzipped directory contained within the IAGO zip folder.

If successful, you should see the project appear in the window. Click finish.

The project, when imported, will have errors! This is because the Tomcat libraries have not yet been associated with the project.

Let's add Tomcat as a server now.

Click Window --> Show View --> Servers

Within the newly opened Servers view, right-click --> New --> Server.

Select Apache --> Tomcat 7.0 --> Next

We need to locate the Tomcat installation directory. Since you have already installed Tomcat above, do NOT click Download and Install. Simply click Browse... then find the Tomcat installation directory. Click Ok --> Next.

Now, select your IAGO project, and Click Add. It should move to the "Configured" column. This automatically configures it using server.xml. Click finish.

Finally, you may additionally need to associate the libraries. Right-click your project in the Package Explorer window --> Build Path --> Configure Build Path --> Add Library... --> Server Runtime --> Apache Tomcat 7.0.

That's it! Now, there should be no errors. To run the project, select Run as... --> Run on Server. This should now synchronize your console and debugger with the running instance. We do recommend that you view IAGO through a full web browser (Eclipse may try to use it's "built-in" browser).

WARNING: You may find that the server fails to start because "another instance is running". This may be due to the fact that the Windows service for Tomcat is still running on port 8080! To solve this problem, either shut down the Windows service for Tomcat before launching in Eclipse, OR edit either Eclipse's server.xml or Tomcat's main server.xml to run on a different port (we use 8085). Then, you may have BOTH the Windows Tomcat and the Eclipse Tomcat instance running simultaneously.

Part 6: A Brief Tour of IAGO's Code

You are now ready to design your own agent. If you are interested in configuring the game itself, including the number of issues, the points awarded, and other structural items, please refer to the documentation for the GameSpec interface. A sample class implementing this interface, ResourceGameSpec, has been provided for you.

GameBridge contains the mechanics for hosting IAGO as a web app, and will not require modification except to configure IAGO's mail data. To receive results of a game for analysis, please make sure both ServletUtils.setCredentials() AND GameSpec.getTargetEmail() are both correct. Failing to configure ServletUtils.setCredentials() will lead to a game exception.

IAGO comes with two sample agents for testing. The DefaultAgent is a skeleton for your code, but the PinocchioAgent has more advanced behaviors, including preference estimation and fair exchange. For a detailed description of that agent, please refer to the upcoming paper reference at AAMAS 2017, found at:

http://people.ict.usc.edu/~mell/publications/iago_4agent.pdf

FAQs

I'm getting a message that the "opponent has left" or the "connection was terminated early" when I try to play IAGO! What should I do?

Your code is throwing an exception. Refer to either Tomcat's logs or Eclipse's debugger.

I think I've found a bug. Is that true, and how can I get it fixed?

While we've tried to make a stable release of IAGO for your use, IAGO is still in Beta Release. Please send an email to iago@ict.usc.edu with your name, a description of the problem, and ideally a screenshot and we will try to address your concerns as soon as possible.

I don't like the idea of MessagePolicies and BehaviorPolicies etc. being separate. Can't I just put it all in one class?

Sure, so long as your code isn't cluttered! Your agents just need to extend GeneralVH and implement all the override methods.

It looks differerent in Internet Explorer 6.0 than Chrome! What gives?

We try to support as many modern browsers as we can. However, some older browsers may lead to slight visual distortion. IE 11 is the only currently supported IE browser by Microsoft.

I'm competing in ANAC! How will I submit my code?

Additional instructions for submitting and compiling your code will be made available closer to the deadline. Generally, you will submit your agent .java file (which extends GeneralVH.java) and any supporting files. You will NOT submit any WebContent or .js files, nor will you submit any implementing classes of GameSpec.java. For now, feel free to test your code locally. The ANAC deadline will be in June 2017, subject to change.

I made some changes to items in edu.usc.ict.iago.views for testing. Will those changes be preserved for ANAC?

No. While it's fine to test differnt types of games by altering ResourceGameSpec, the organizers will provide their own GameSpec implementation, with point values and other settings prepared by us. Your agent and any associated classes should simply extend GeneralVH and work with various types of points and issues.

I see a lot of data being outputted to the Eclipse console, but it's hard to read there. Can I just write it to a file?

Sure. See this tutorial for a quick way to do this.

I've changed the agent name in several places, but it's still showing up as "Brad"!

There are a couple of places to check. First of all, have you changed the "Welcome Message" on line 612 of iago.js? Secondly, has your class that implements "GameSpec.java", which is by default "ResourceGameSpec.java" set the return data for `getEndgameMessage()`? Please note that these changes **will not be kept** for the competition. Only changes made within your agent class, such as `getArtName()` (which should always be either "Brad" or "Ellie") and `agentDescription()` can be made permanently.

I'm trying to configure my Gmail to be a server for sending out game summaries for me to look at to help with debugging and it wont work.

Everyone's email server is different, but the following settings worked for my gmail. This code was inserted into the `GameBridge.java` constructor. Don't forget all this data is available in the console output as well, so it's not strictly necessary to set this up.

```
System.setProperty("mail.smtp.ssl.enable", "true");
ServletUtils.setCredentials("yourAccount@gmail.com", "password", "Your Human Name",
"true", "smtp.gmail.com", "465");
```

Do note that you should enable SSL as per above! Gmail requires this, as it is more strict that other servers.

What are message codes, as alluded to by getMessageCode in the Event class?

Message codes are a way of quickly determining what message a user has sent to the agent without doing a lot of natural language processing or String comparisons. Message codes 0 - 12 are reserved for the natural language utterances specified in the menu. These are listed in the example code in ResourceGameSpec.java. So, for example, message code 4 corresponds to "Accept this or there will be consequences". Message code 100 is reserved for an offer rejection, and message code 101 is reserved for an offer acceptance.