JATOS Documentation

Last generated: December 30, 2016

Table of Contents

Reference

What's JATOS

JATOS (Just Another Tool for Online Studies) helps you setup and run your online studies on your own server.

JATOS 2 allows you to run group studies (page 0), where multiple workers can interact with each other.

You can read our open access paper (http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0130834) for the basics, but this wiki contains up-to-date information about the new group feature.

We also started a blog about JATOS and online experiments in general (http://blog.jatos.org). There we'll post about topics that do not necessarily fit into this wiki.
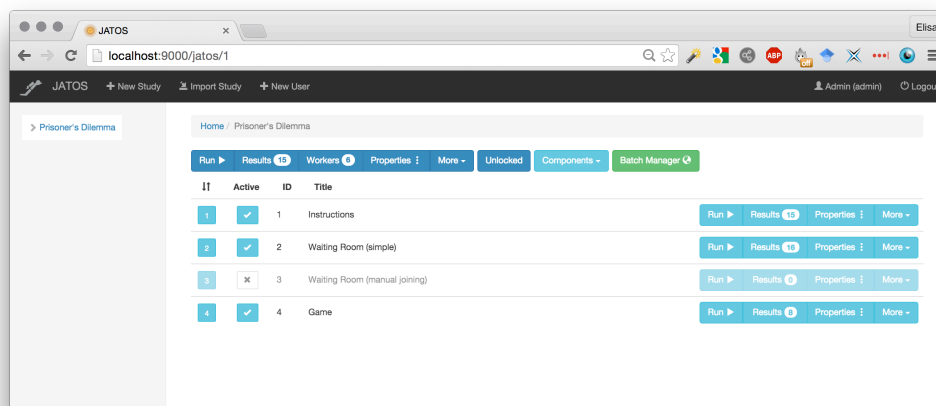
JATOS at a glance:

- Run studies on your **own server**. This means that you keep complete control over who can access your result data.

- **Run group studies.**

- It's GUI-based, so there's no need to use the terminal to talk to your server.

- **Mobile phone ready**: Run your own, tailor-made studies, on any device with a browser (mobile phone, tablet, desktop, lab computer, thin client).

- Studies are easily written by yourself in **HTML / JavaScript / CSS** (you will need some basic scripting skills).

- Connect to the Amazon Mechanical Turk (https://www.mturk.com) (MTurk) marketplace to recruit participants.

- Manage workers, to e.g. make sure that each participant does your study only once.

- **Export/Import** studies to facilitate exchange with other researchers.

- It's open-source and free to use.

- It's very easy to Get started (page 0)!

Please cite us (http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0130834) if you use JATOS for your research.

This is the wiki for JATOS 2 - Are you looking for the Wiki of JATOS 1.x (https://github.com/JATOS/JATOS_docs_v1/wiki/Home)?

Download the latest release (https://github.com/JATOS/JATOS/releases/)

Installation

Easy installation on your local computer

JATOS runs on MacOS X, MS Windows and Linux

A local installation is straightforward.

Usually you first develop your study with JATOS on a local computer. Then in a second step you bring it to a server installation of JATOS.

With a local installation only you have access to JATOS - with a server installation (page 45) others can run your study via the internet too. This is especially true if you want to publish your study on Mechanical Turk.

For convenience JATOS is available in a variant bundled with Java.

To run JATOS, you need Java installed on your computer (to be precise, you need a Java Runtime Environment, aka JRE). Chances are, you already have Java installed. To check whether Java is installed on your system, type `java -version` in your terminal (MacOS X / Linux) or command window (MS Windows). If you don't have Java installed, you can either download and install it directly (http://www.oracle.com/technetwork/java/javase/downloads/index.html) or download and install JATOS bundled with Java, according to your operating system.

Installation MS Windows

1. Download the latest JATOS release (https://github.com/JATOS/JATOS/releases/latest) (exchange 'xxx' with the current version)

   - Without Java: *jatos-xxx.zip*

   - Bundled with Java: *jatos-xxx_win_java.zip*

2. Unzip the downloaded file

3. In the File Explorer move to the unzipped JATOS folder and double-click on `loader.bat` . (Or `loader` alone, if your filename extensions are hidden). A command window will open and run your local JATOS installation. Simply close this window if you want to stop JATOS.

4. All set! Now go to the browser of your choice and open http://localhost:9000 (http://localhost:9000). You should see the login screen (wait a moment and reload the page if you don't). Login with username 'admin' and password 'admin'.

Installation MacOS X and Linux

1. Download the latest JATOS release (https://github.com/JATOS/JATOS/ releases/latest) (exchange 'xxx' with the current version)

    • Without Java: *jatos-xxx.zip*

    • For MacOS bundled with Java: *jatos-xxx_mac_java.zip*

    • For Linux bundled with Java: *jatos-xxx_linux_java.zip*

2. Unzip the downloaded file

3. In your terminal window, cd into the unzipped JATOS folder

4. Run the loader shell script with the command `./loader.sh start` (You might have to change the file's permissions with the command `chmod u+x loader.sh` to make it executable)

5. All set! Now go to the browser of your choice and open http://localhost:9000 (http://localhost:9000). You should see the login screen (wait a moment and reload the page if you don't). Login with username 'admin' and password 'admin'.

Your local JATOS installation will run in the background. If you want to stop it, just type `./loader.sh stop` in your terminal window.

How to go on from here

The easiest way to start with JATOS is to download and import one of the example studies (https://github.com/JATOS/JATOS_examples/wiki/Example-Studies) and play around with it (page 7).

Get started

1. **Download JATOS and install a local instance (page 5)**

2. **Open JATOS' GUI by going to http://localhost:9000/ (http://localhost:9000/) in your browser window**

3. **Download and import an example study**

4. Download one of the Example Studies (https://github.com/JATOS/ JATOS_examples/wiki/Example-Studies), e.g. the 'Go- / No-Go Task' with jsPsych. Do not unzip the downloaded file.

5. Import the study into JATOS: Go to JATOS' GUI in your browser and click on **Import Study** in the header. Choose the .zip file you just downloaded. The imported study should appear in the sidebar on the left.

6. **Explore the GUI**

In the sidebar click the study to get into the study's page.

To do a test run of the entire study, click on **Run** in the toolbar on top of the page.

If you finished running through the study, you can check the results. * To see whole-study results, click on the **Results** button on the top of the page. * To see results from individual components, click on the **Results** buttons on each component's row.

You can see each result's details be clicking on the little arrow to the left of its row.

Here's a screenshot of a study's results view:
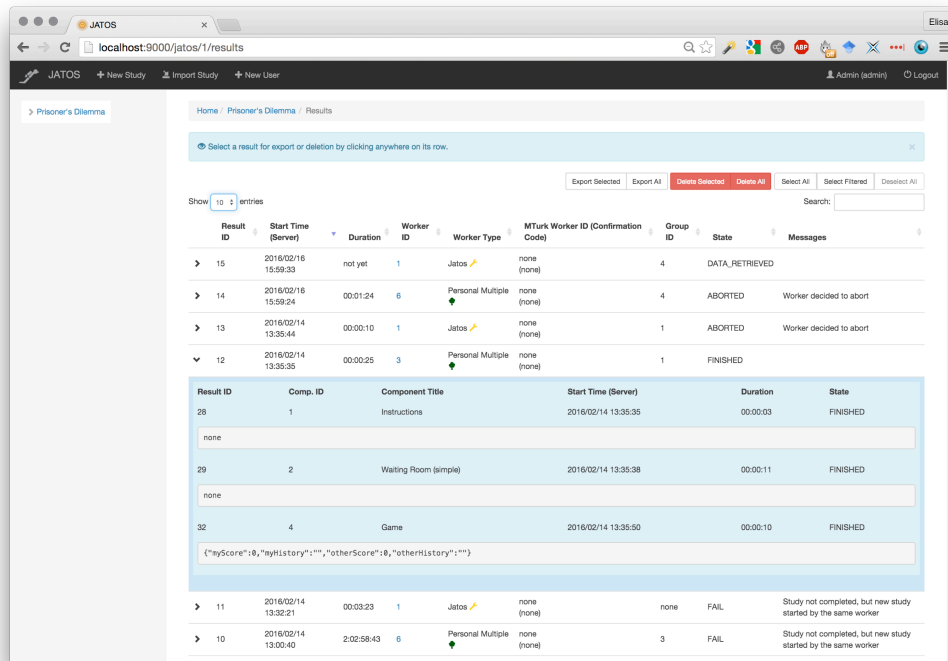
Explore

Now it's time to explore a little bit more.

- You can click on any component's position button and drag it to a new position within the study.

- Each component has a **Properties** button. The component's HTML file may read the data in the field 'JSON data'. This is a way to make changes in the details of the code (wording of instructions, stimuli, timing, number of trials, etc) without having to hard-code them into JavaScript.

- Where are the actual HTML, JavaScript, and CSS files? They are the files that actually run your study, so make sure you can locate them. They will be in `/path_to_my_JATOS/study_assets_root/name_of_my_study/`.

Here's a screenshot of a component's properties view:
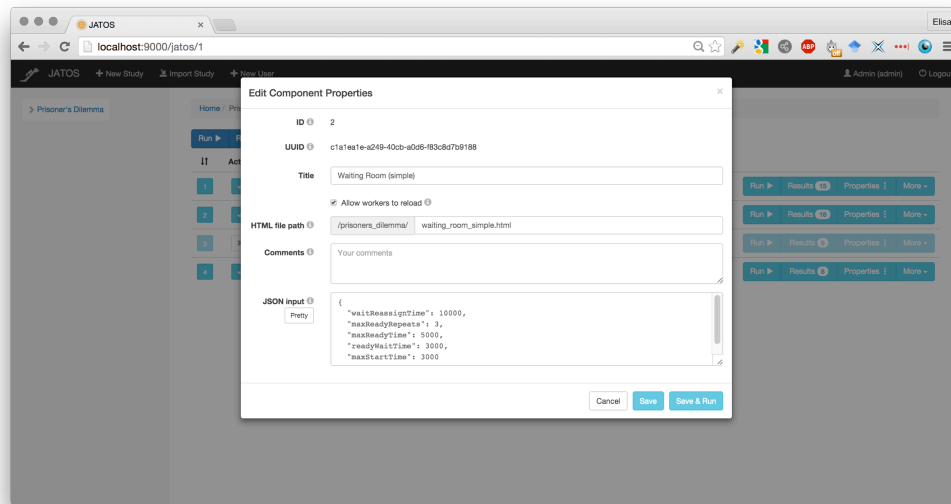
Set up your own studies

So, now you've installed JATOS and tried out all the examples studies. What now?

Create a new study

There are two ways to create a new study:

1. from scratch by pressing the **New Study** button in the header of each JATOS page. Then edit the study properties and add new components manually.

2. take an existing study (e.g. from Example Studies (https://github.com/ JATOS/JATOS_examples/wiki/Example-Studies)) as a prototype and modify it bit by bit. We recommend you start with this to get a quick idea of how JATOS works. Just import an existing study (e.g. one from the study examples) and clone it by clicking on More –> Clone in the study bar.

Write your own studies in JavaScript

The most difficult part - though it's still easy! - is to learn to write your own study component scripts, using HTML, CSS and JavaScripts. Or, instead of reinventing the wheel, you could use a framework like jsPsych that helps you with this (see jsPsych and JATOS (page 25)).

Check out the Mandatory lines in your components' HTML (page 0) page to know what you absolutely must include in your scripts in order to let JATOS see them. Also check out the jatos.js Reference (page 62), that contains a set of very useful functions that you have to use to communicate with jatos (to e.g. submit and receive data).

If you are a newbie to HTML/JavaScript programming, there are LOADS of free and excellent tutorials online. Like this one from the Kahn Academy (https://www.khanacademy.org/computing/computer-programming) or more searchable tutorials like the simple ones from the w3 schools (http://www.w3schools.com/). In addition, StackOverflow (http://stackoverflow.com/ questions/tagged/html) is the best place to solve the problems that hundreds of others have encountered before.

Import / Export of studies

Usually you conveniently develop your study on your local computer where you have a local installation of JATOS (page 5). Then just use the export and import buttons in your installations to transfer the study to your JATOS server (page 45).

1. In the GUI of your local installation press **Export** in the Study Toolbar. JATOS saves your study asset folder and some data about your study and it's components (mostly the properties) into a ZIP file and lets you download it. Leave the ZIP file as it is.

2. In the GUI of your server installations press **Import** in the header. Select the ZIP you saved in step 1. JATOS will upload and unpack your study.

If you have trouble with the export and you are using a Safari browser have a look into this issue in our Troubleshooting section (page 0).

Decide how you're going to recruit your workers and generate the links

Once you have your study running and have it on a server instance, you'll need to get your workers to access your study. Different types of workers (page 16) might be allowed to run your study. You could, but you don't have to, recruit your workers using MTurk (page 29). You could also generate direct links in the Worker Setup (page 0) to send to different workers.

Export your result data

After you let workers run your study and you gathered result data you probably want to export them to your local computer for further analysis. Go to one of the **Results** views and select the results you want to export (select them by just clicking somewhere in the row). Then click 'Export Selected'. This will download all your results in one text file.

Analyse your data

Once you have collected all your data, export it to text files. If you used the JSON format (which is handy - but any other text is fine too) you can analyse your data using this nice JSON parser for Matlab and Octave (http://iso2mesh.sourceforge.net/cgi-bin/index.cgi?jsonlab) or the JSON parser for R (http://cran.r-project.org/web/packages/jsonlite/index.html).

Run your Study with Batch Manager & Worker Setup

If you run a study during development from within the JATOS GUI you will do so as a Jatos worker. There are other worker types (page 16) to distribute to real participants.

A worker in JATOS is a person who runs a study. A batch organizes workers and their study runs and results. Each study has a Batch Manager.

Generate links to send to your workers

For all worker types except Jatos and MTurk you'll need to generate links to send to your potential workers, which they can follow to access and run your study. Those links are generated in the **Worker Setup** (Batch Manager –> Worker Setup).

Restrictions

1. Common to all worker types is that **the same worker can do only one study at the same time**. If a worker is running a study and starts another study run, the first study run will automatically be finished with an error. This is independent of whether the study runs are in the same or different browsers.

2. **[Valid only for versions 2.1.12 and older] The same browser can only run one study at the same time**. If you start a second study in a browser where you have started a first study, the first study will automatically be finished with an error.

3. **[Valid only for versions 2.2.1 and newer] The same browser can run one study in up to 10 different tabs**. This is useful if you want to test a group study. You can open up to 10 different tabs in the same browser (with a personal multiple link or as the jatos worker) and all these 10 "different" workers will interact with each other. Once you open the 11th tab, the first worker will be thrown out of the group.

Batch Manager

Since JATOS 2 you run studies in batches. A batch consists of its properties and a collection of workers that do the study. Using different batches is useful to organize your study runs, separate their results and vary their setup. E.g. you could separate a pilot run from the 'proper' experiment, or you could use different batches for different worker types.

Batches are organized in the Batch Manager. Here you can create and delete batches and access each batch's properties, worker setup and study results.

Each study comes with a 'Default' batch.
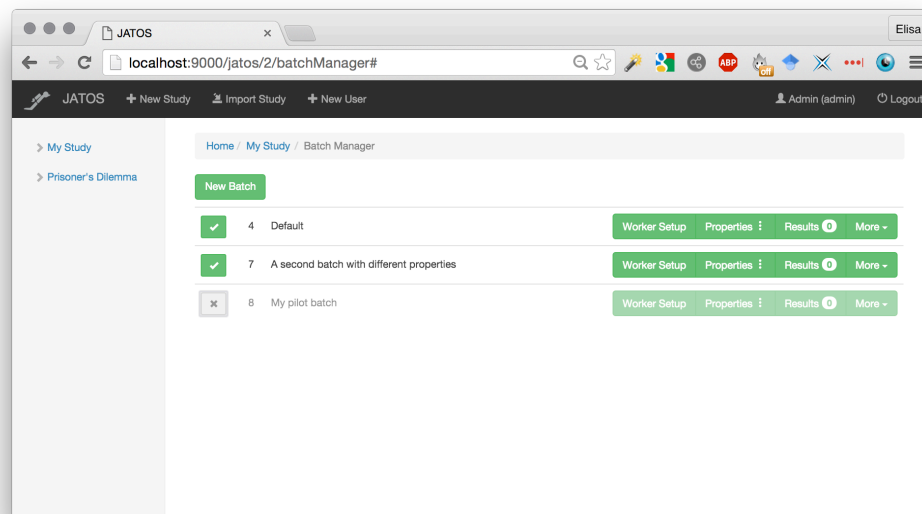
Batch Properties

For each batch, you can limit the maximum number of workers that will ever be able to run a study in this batch by setting the **Maximum Total Workers**.
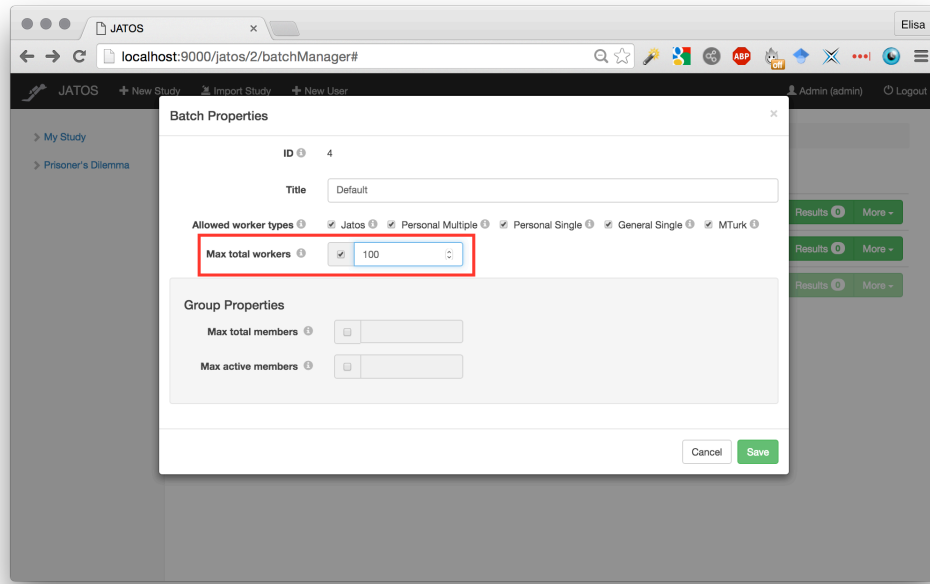
Additionally you can switch on or off worker types in the **Allowed Worker Types**. Forbidden worker types are not able to run a study. Always check before you send out links to study runs that the corresponding worker types are switched on. You can do the same in the Worker Setup.
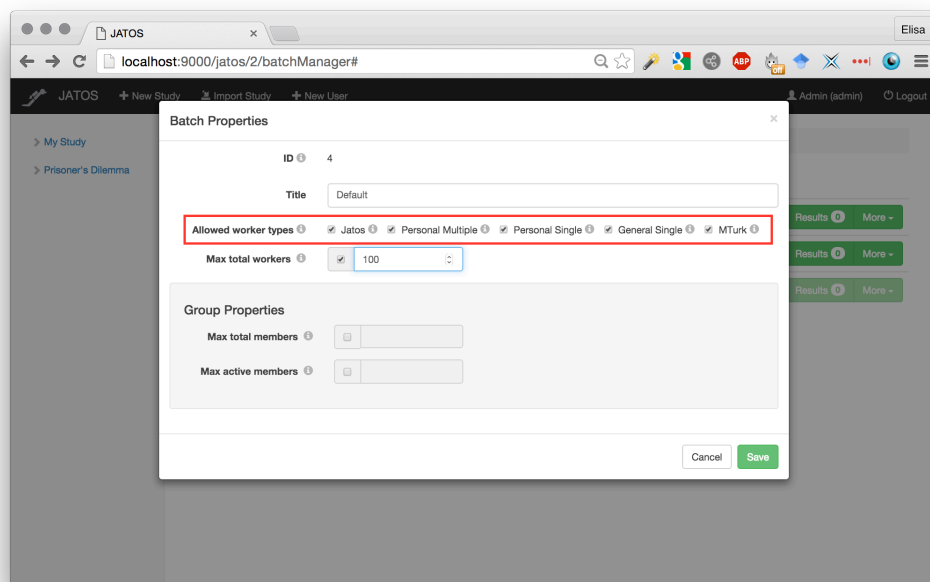
The **Group Properties** relate to group studies (page 0).

Worker Setup
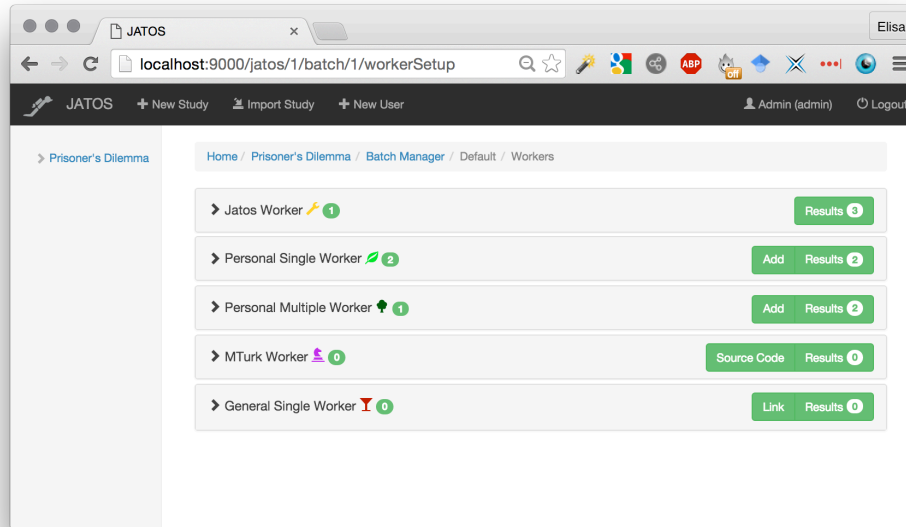
This is the place where you generate the links to study runs for most of your workers types (except for Jatos and MTurk).

For **Personal Single Workers** and **Personal Multiple Workers** click **Add**. You can enter a worker description or identification in the 'Comments' box.

General Single Workers only have one link and each link will create a new separate worker. Get this link by clicking on **Link** in its row.

How to connect to MTurk and create links to run for **MTurk Workers** is described in the page Connect to Mechanical Turk (page 29).

The differences between the worker types is explained in the Worker Types wiki page (page 16).

Worker Types

Following Amazon Mechanical Turk's terminology, a worker in JATOS is a person who runs a study. Different worker types access a study in different ways. For example, some workers can run the same study multiple times, whereas others can do it only once. Below we describe the different workers:

Jatos Worker

Jatos workers run a study (or any of its components individually) by clicking on the *Run* buttons from the GUI. Jatos workers are usually the researchers trying out their own studies. Each JATOS user (i.e., anybody with a JATOS login) has their own Jatos worker.
Jatos workers can run the same study as many times as they want.

Personal Single Worker

These are external workers that can **only run the study once** (*But see Preview Links (page 18)). You send a *Personal Single* link to workers that you contact individually. Each link can be personalized with a **Comment** you provide while creating it (e.g. by providing a pseudonym). Personalized single links with single access are useful in small studies, where it's feasible to contact each worker individually, or (e.g.) you want to be able to pair up several results (either from the same or different studies) in a longitudinal design. You can add in bulk up to 100 Personal Single Workers at once by changing the **Amount** value.

Personal Multiple Worker (personal with multiple access)

These are external workers that **can run a study as many times as they want**. You could send *Personal Multiple* links to your pilot workers. Each link can be personalized with a **Comment** you provide while creating it (e.g. by providing a pseudonym). You can add in bulk up to 100 Personal Multiple Workers at once by changing the **Amount** value.

MTurk Worker

MTurk workers access a study **only once**, through a link in Amazon's Mechanical Turk (AMT). Each MTurk worker in JATOS corresponds to a single worker in AMT.

DATA PRIVACY NOTE: If the same worker from AMT does two of your studies, the two results will be paired with the same MTurk Worker in JATOS. This means that you could gather data from different studies, without your workers ever consenting to it. For this reason, we recommend that you delete your data from JATOS as soon as you finish a study. This way, if the same worker from AMT

takes part in a different study, they will get a new MTurk Worker, and you will not be able to automatically link their data between different studies. See our [[Data Privacy and Ethics]] page for more details on this.

General Single Worker

These are external workers that can **run a study only once** (*But see Preview Links (page 18)). You could distribute a *General Single* link through a mailing list or posting it on a public website. It is essentially useful for cases where you want to collect data from a large numbers of workers. Each new worker accessing the study will be assigned a new General Single Worker.

Keep in mind, however, that JATOS uses the browser's cookies to decide whether a worker has already accessed a study. If a worker uses a different computer, a new browser, or simply deletes their browser's cookies, then JATOS will assume that it's a new worker. So the same person can easily use a General Single link several times.

Preview Links

A normal **General Single** or **Personal Single** is restrictive: once a worker clicked on the link - that's it. JATOS will not let them run the study twice. But in some cases you may want to distribute a link and let your workers preview the first component of your study (where you could e.g. describe what they will have to do, how long it will take, ask for consent, etc). Often, workers see this description and decide that they want to do the study later. To allow them to do this, activate the checkbox **Allow preview** (this will add a `&pre` to the end of the URL). Now your workers can use the link as often as they want - as long as they don't go further than the first component. But once the second component is started, JATOS will restrict access to the study in the usual way as it is for General Single and Personal Single workers. This means another usage of the link will lead to an error.

Two Types of Session Data

Often you want to store information during a study run and share it with other components of the same study, or - if it is a group study - within the group. For this JATOS provides the Session Data. There are two different Session Data types, both of which are stored **temporarily** in your database, and will be deleted once they are no longer in use.

The difference between session data and the result data is that the results are stored **permanently** in the database, and will stay there after the study is finished. All results should therefore go into the result data.

Study Session Data

The Study Session Data is useful to share information between components of a given study. Some examples include:

- Sending information about percentage of correct responses in component 1 to component 2.

- Having a study-wide progress bar, showing how much of the entire study has been completed.

- Keeping track of the number of iterations of a given component that is repeated.

Group Session Data

The Group Session Data is useful to share information between members of a given group. Some examples include:

- Responses to each trial in a group study like the Prisoner's Dilemma

- gender of group members

Mandatory lines in your components' HTML

The best way to write an HTML/JavaScript file that runs with JATOS is to use one of the HTML files from the Example Studies (https://github.com/JATOS/ JATOS_examples/wiki/Example-Studies) as a starting point.

Here are the absolute basics that any component HTML file MUST have in order to correctly run with JATOS

1. A link to the jatos.js library in the head section

```
<html>
  <head>
    <script src="/assets/javascripts/jatos.js"></script>
  </head>
</html>
```

Remember (page 32): Any URL or file path in a HTML file should only use '/' as a file path separator - even on Windows systems.

2. A callback function that will execute after a component has finished loading

```
<script>
  jatos.onLoad();
</script>
```

The code above is valid, but your component will do nothing after being loaded, which is probably not what you want. Typically, you would define the function that would get your JavaScript going after component load. Something like this, for example, will show an alert box:

```
<script>
  jatos.onLoad(function() {
    alert("I am an alert box!");
  });
</script>
```

Adapt Pre written Code to run it in JATOS (Jatosify)

Make Your Existing Code Run in JATOS - or How To Jatosify a Study

You might have a task, experiment, survey, or study running in a browser. You might have all its files like HTML, JavaScripts, images, etc. Maybe you wrote it with jsPsych (http://www.jspsych.org) or got it from The Experiment Factory (http://expfactory.github.io). Do you want to run it with JATOS? That's easy!

Create the study in JATOS

1. Create a new study with the '**New Study**' button in JATOS' header. Choose a study title and a folder name. Leave the other fields empty for now and click 'Create'. JATOS will have created a new folder within your assets root folder (default is

   `/path_to_your_JATOS/study_assets_root/`).

2. Copy all your files (HTML, JavaScripts, images, audio, …) into your new study folder.

3. Back in the JATOS GUI, and within the newly created study, create a **new component** by clicking 'Components' and then 'New'. Choose a component title and set the HTML file name, to the name of the HTML file you just copied into the study folder.

4. In your HTML and JavaScripts, change all paths to your **study assets** folder: Always use the prefix `/study_assets/` and then the study assets name you specified in your study's properties when you created it.

   - E.g. if you load the CSS file `snake.css` and the study's assets name is `group_snake` use

     `<link rel="stylesheet" type="text/css" href="/study_assets/group_snake/sna`

   - Or if you want to load some JavaScript from your local study assets with the name `prisoner_dilemma`, e.g. the jQuery library, use

     `<script src="/study_assets/prisoner_dilemma/jquery-1.11.1.min.js"></script`

5. Now it's time for a first glimpse: Click the '**Run**' button in either the study's or the component's toolbar. Your experiment should run like it did before without JATOS. Use the browser's developer tools to check for eventually missing files and other occurring errors.

Edit your HTML and JavaScript

Up to this point JATOS served as a mere provider of your files. Now we want to use a feature of JATOS: We want to store your result data in JATOS' safe database.

1. Include the **jatos.js** library in your HTML `<head>`

   Add the line

   ```
   <script src="/assets/javascripts/jatos.js"></script>
   ```

2. Add `jatos.onload`

   Every study in JATOS starts with this call. So whatever you want to do in your study it should start there.

   ```
   jatos.onLoad(function() {
     // initialize and start your JavaScript here
   });
   ```

   E.g. if you want to initialize a jsPsych experiment:

   ```
   jatos.onLoad(function() {
     jsPsych.init( {
       ...
     });
   });
   ```

3. Now to actually send our result data to JATOS we use jatos.js' function `jatos.submitResultData`. We can pass this function any data in text format including JSON, CSV or XML.

   E.g. if we want to send a JavaScript object as JSON

   ```
   var resultJson = JSON.stringify(myObject);
   jatos.submitResultData(resultJson, jatos.startNextCompon
   ent);
   ```

Conveniently but optionally `jatos.submitResultData` takes a second parameter which specifies what should be done after the result data got sent. Usually one want to jump to the next component (`jatos.startNextComponent`) or finish the study (`jatos.endStudy`).

Another example where we use jsPsych: We have to put `jatos.submitResultData` into jsPsych's `on_finish` :

```
jsPsych.init( {
  ...
  on_finish: function(data) {
    // Submit results to JATOS
    var resultJson = JSON.stringify(jsPsych.data.getDat
a());
    jatos.submitResultData(resultJson, jatos.startNextCo
mponent);
  }
});
```

That's about it. Infos about other jatos.js functions and variables you can find in the reference (page 62).

Beyond the basics

- Think about dividing your study into **several components**. You could have separate components e.g. for introduction, training, experiment and feedback. You could even consider splitting the experiment into several parts. One advantage is that if your participant stops in the middle of your study you still have the result data of the first components. Also, you can re-use components in different studies.

- Use the study's and component's '**JSON input data**'. With them you can change variables of your code directly through JATOS' GUI, which might come handy if someone isn't good in JavaScript.

- You can add a **quit button** to your study to allow the participant to abort at any time (page 36).

jsPsych and JATOS

JATOS basically cares for the server side: it stores result data, does worker management etc. JATOS doesn't care so much for what happens in the browser itself - your HTML, JavaScript and CSS. Of course you can write this all yourself, but you could also use a framework for this. A very good one is jsPsych (http://www.jspsych.org/).

In our example studies (https://github.com/JATOS/JATOS_examples/wiki/Example-Studies) are a couple of jsPsych ones.

Here are the necessary changes if you want to adapt your jsPsych experiment so that it runs within (and send the result data to) JATOS. Additionally you can have a look at Adapt Pre written Code to run it in JATOS (Jatosify) (page 22).

How to turn your jsPsych experiment into a JATOS study

1. Include the `jatos.js` library in the `<head>`

```
<script src="/assets/javascripts/jatos.js"></script>
```

Remember (page 32): Any URL or file path in a HTML file should only use '/' as a file path separator - even on Windows systems.

2. Wrap jsPsych's init call `jsPsych.init` in a `jatos.onload` call

```
jatos.onload(function() {
  jsPsych.init( {
    // ...
  });
});
```

That's all. If you additionally want to send your result data to JATOS read on.

Send jsPsych's result data back to JATOS

Here we use jsPsych's function `jsPsych.data.getData()` to collect the data into a variable and then 'stringify' the JSON format into a simple string. Then we use JATOS' function `jatos.submitResultData` to send your result to JATOS and asks JATOS to move to the next component, if there is one.

```
jatos.onload(function() {
  jsPsych.init( {
    // ...
    on_finish: function() {
      var resultJson = JSON.stringify(jsPsych.data.getData());
      jatos.submitResultData(resultJson, jatos.startNextCompone
nt);
    }
  });
});
```

Tips & Tricks

Imitate a run from Mechanical Turk

You should always test your study before posting it anywhere. Testing that your study runs via a simple link is easy: just generate the link, start the study and run once through it. Testing studies posted in MTurk is especially cumbersome, because you should make sure that the confirmation codes are correctly displayed when the study is over. The standard way to test this is to create a study in MTurk's Sandbox (https://requester.mturk.com/developer/sandbox). JATOS offers a way to emulate MTurk, without having to set up anything in the sandbox. Here's how.

If you think about it, MTurk simply calls a JATOS URL. The URL to start a study is normally `http://your-jatos-server/publix/study-id/start` (where `study-id` is a placeholder for the ID of the study you want to run). Two additional variables in the URL's query string tell JATOS that this request comes from MTurk (and that it should display the confirmation code when the study is over): `workerId` and `assignmentId`. Both pieces of information are normally generated by MTurk; but they can be any arbitrary string. The only constraint is that the `workerId` does not already exist within JATOS. (Think of it this way: Because a MTurk worker can run a study only once, the same `workerId` can be used only once in JATOS.)

Here are some concrete examples:

To run the study with the ID 4 on a local JATOS use
`http://localhost:9000/publix/4/start?workerId=123456&assignmentId=abcdef`.

To imitate a run from **MTurk's Sandbox**, use any arbitrary values in the query strings `workerId` and `assignmentId` (In this example, `workerId = 12345` and `assignmentId = abcdef`). Also, set `turkSubmitTo` to the value 'sandbox'.

`http://localhost:9000/publix/4/start?workerId=123456&assignmentId=abcdef&turkSubmitTo=sa`

Lock your studies before running them

Each Study bar has a button that toggles between the 'Unlocked' and 'Locked' states. Locking a study prevents changes to its (or any of its components') properties, change the order of components, etc.
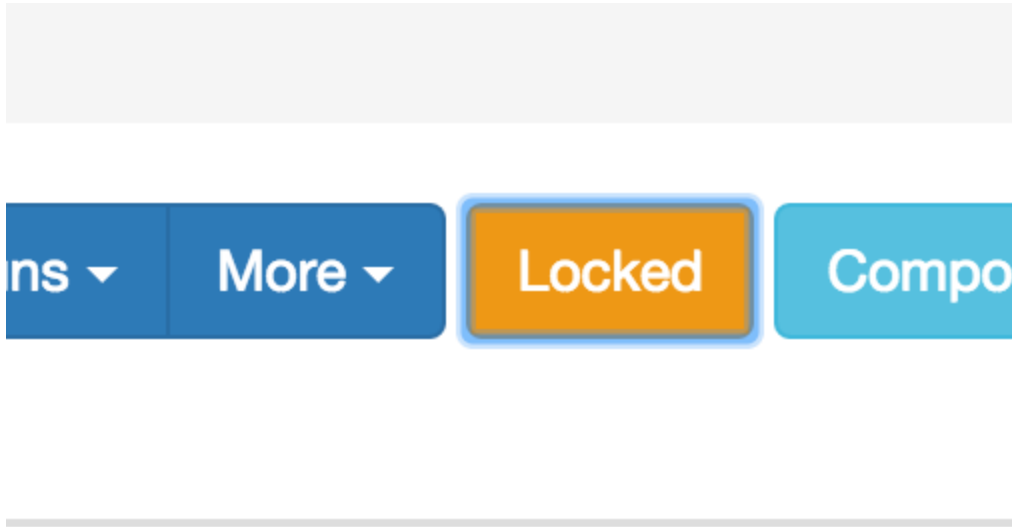
Do a General Single Run more than once in the same browser

The problem here is that a General Single Run is intended to work only once in the same browser. Although this is a feature to limit participants doing the same study twice, it can be a hassle for you as a study developer who just want to try out the General Single Run a second time. Luckily there is an easy way around: Since for a General Single Run all studies that the worker already participated in are stored in a browser cookie, it can be easily removed. Just **remove the cookie with the name JATOS_GENERALSINGLE_UUIDS** in your browser. You can find this cookie in every webpage hosted by a JATOS server. If it doesn't exist you probably never did a General Single run yet.

Connect to Mechanical Turk

Connecting your JATOS study to the Mturk is easy, although a fair amount of clicking is required.

You will need: * A requester Mturk account * Your study running on JATOS * A description of the study (this can be the same as the one you included in the study description within JATOS)

The steps to create a project are part of the MTurk interface.

1. Create –> New Project –> Other –> Create Project

2. Complete the 'Enter Properties' tab

3. Click on the 'Design layout' button in the bottom of the page.

4. Click on the 'Source' button. You'll see some text in an editable window, corresponding to an html file. Delete the entire text in this field.
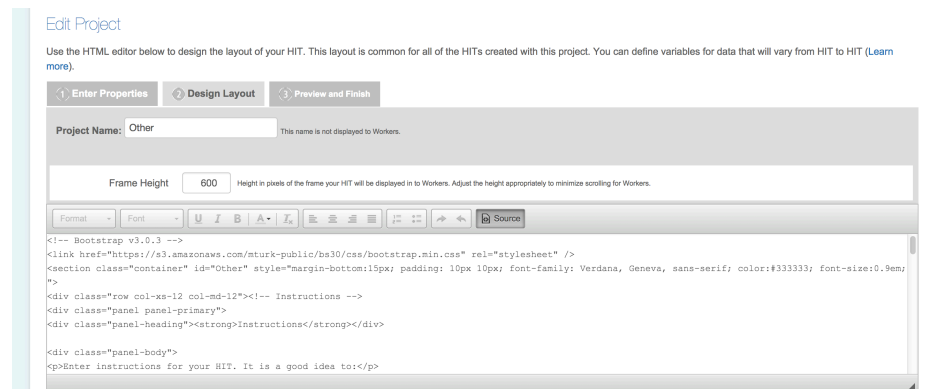
5. In JATOS, go to the Study Toolbar —> Batch Manager —> Worker Setup —> MTurk Worker row —> Source Code.
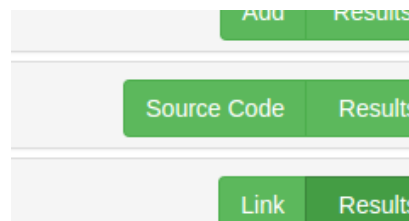
6. You'll see a box with HTML code, similar to the one shown here. Copy paste the code from JATOS to the MTurk source section.
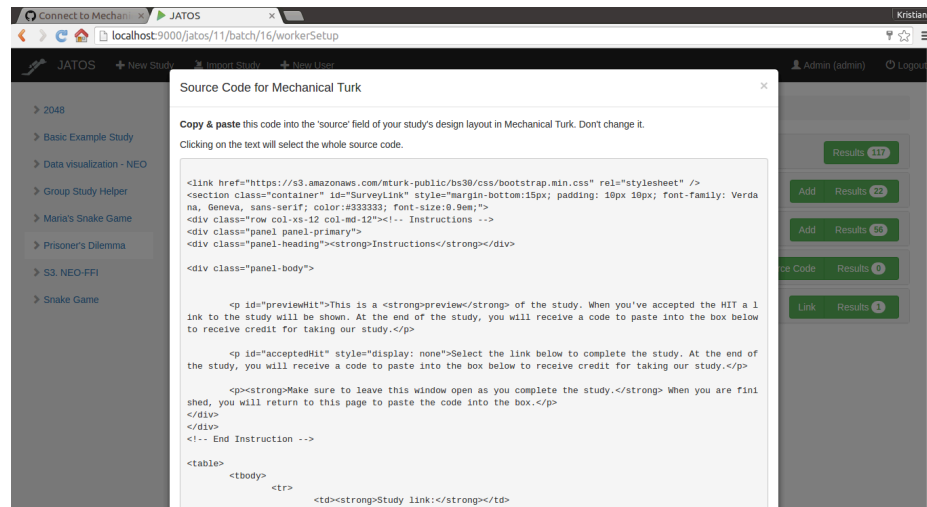
7. Back in MTurk click on the 'Source' button again, and continue setting up your study in MTurk.
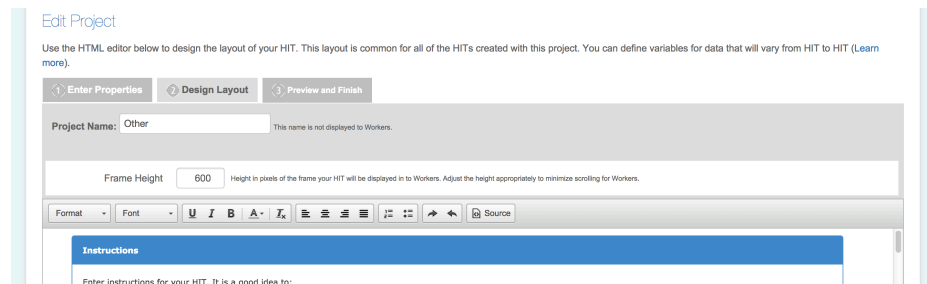
8. When an MTurk worker finishes a study they'll see a confirmation code. To assign payment to individual workers, just compare the confirmation codes stored in JATOS' results view to those stored in MTurk.

Confirmation code:
dedb7ee1-5c94-4e91-a725-06cac3889a76
(Copy and paste the confirmation code to Amazon Mechanical Turk.)

See the Tips & Tricks (page 27) page for some useful information on how to test your study before officially posting it on MTurk.

Troubleshooting

JATOS failed to update the GUI automatically?

Try reloading the browser.

Downloading a study / exporting a study fails (e.g. in Safari browsers)

As a default, Safari (and some other browsers) automatically unzips every archive file after downloading it. When you export a study, JATOS zips your study together (study properties, all components, and all files like HTML, JavaScripts, images) and delivers it to your browser, who should save it in your local computer. Safari's default unzipping interferes with this. Follow these instructions (https://discussions.apple.com/thread/1958374?start=0&tstart=0) to prevent Safari's automatic unzip.

JATOS fails to start?

(Or, if you are running Windows, do you get the message 'JATOS is already running. Press any key to continue'…)

This will happen if your computer crashed before you had the chance to close JATOS.
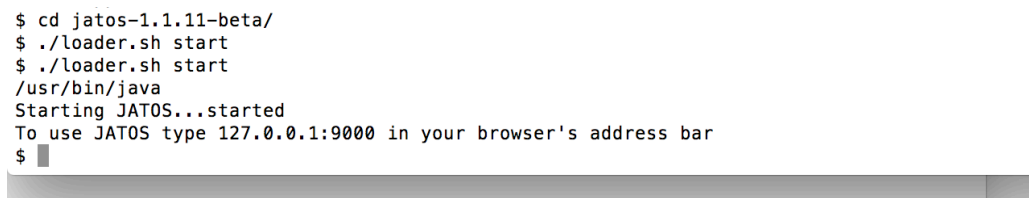
This is what you might see on a Mac Terminal if JATOS doesn't start:

```
$ cd /Applications/
$ cd jatos-1.1.11-beta/
$ ./loader.sh start
$
```

Close any open command prompt windows. Then look into your JATOS folder, and check if there's a file called 'RUNNING_PID'. Delete this file and try to start JATOS again.

Here is how it should look if JATOS started successfully:

```
$ cd jatos-1.1.11-beta/
$ ./loader.sh start
$ ./loader.sh start
/usr/bin/java
Starting JATOS...started
To use JATOS type 127.0.0.1:9000 in your browser's address bar
$
```

Read log file in the browser

In a perfect world, JATOS always works smoothly and, when it doesn't, it describes the problem in an error message. Unfortunately we aren't in a perfect world: every now and then something will go wrong and you might not get any clear error messages, or no message at all. In these (rare) cases, you can look into JATOS' log file to try to find what the problem might be.

The standard way to read the log file is directly on the server. You'll find your complete log file in `jatos_directory/logs/application.log` . Because JATOS is designed to avoid the command line interface, we offer a way to view your log file directly in your browser.

Just open the URL `http://your-jatos-server/jatos/admin/log` . For privacy and security reasons, you must be logged in as **Admin**. For example, if you're running JATOS locally with the standard settings, you'd have to go to http://localhost:9000/jatos/admin/log (http://localhost:9000/jatos/admin/log) to view your log file.

By default, JATOS will display the last 1000 lines of the application.log file. If you want to see more than the last 1000 lines, add the query parameter `limit` . E.g. to display the last 10000 lines on a local JATOS instance, you'd have to go to http://localhost:9000/jatos/admin/log?limit=100000 (http://localhost:9000/jatos/admin/log?limit=10000).

A file (library, image, …) included in the HTML fails to load?

There is a common mistake Windows users make that might prevent files in the HTML from loading: Any URL or file path in a HTML file should only use '/' as a file path separator - even on Windows systems. So it should always be e.g.
```
<script src="/study_assets/mystudy/jsPsych-5.0.3/jspsych.js"></script>
```
and not
```
<script src="\study_assets\mystudy\jsPsych-5.0.3\jspsych.js"></script>.
```

Update JATOS

We'll periodically update JATOS with new features and bug fixes. We recommend you stay up to date with the latest release (https://github.com/JATOS/JATOS/releases). However if you are currently running a study it's always safest to keep the same JATOS version throughout the whole experiment.

Updating a local installation of JATOS

(The procedure is different if you want to update JATOS on a server installation (page 60))

To be absolutely safe you can install the new JATOS version and keep the old one untouched. This way you can switch back if something fails. Just remember that only one JATOS can run at the same time. Always end JATOS before starting another one.

You can update your local JATOS instance in two main ways:

First, easy way: discarding your result data

If you don't care about result data stored in JATOS:

1. Simply download and install the new version as if it were a new fresh download. Don't start it yet.

2. Export any studies you wish to keep from the old JATOS installation.

3. Stop the old JATOS and start the new JATOS.

4. Import all the studies your previously exported. This will transfer the files and subfolders in your study's asset folder (HTML, JavaScript, CSS files).

What will be transferred:

1. Files and subfolders in study's assets folder

2. All your studies' and components' properties

3. The **properties** of the first (Default) batch

What will be lost:

1. **All result data will be lost**

2. All workers in all batches (including Default batch)

3. All batches other than the Default batch

Second way: keeping everything (including your result data)

If you do want to keep your studies, batches, and your result data you'll have to move them to the new JATOS.

However: Occasionally (on major version updates such as 1.X.X to 2.1.1) we will make big changes in JATOS, which will require some restructuring of the database. In these cases, the second way described here will not be possible. We will do our best to prevent these big changes and inform you explicitly in the release description.

1. Stop JATOS (on Unix systems, type `$ ./loader.sh stop` on the terminal. On Windows MS, close your command window)

2. Go to the folder of your old JATOS installation. From there copy your assets root folder to the new JATOS installation (Note: By default your assets root folder is called `study_assets_root` and lays in the JATOS folder but you might have changed this. You can find the location and name in `conf/production.conf` . It is specified in the line beginning with `jatos.studyAssetsRootPath=` .)

3. From your the folder of your old JATOS installation copy the folder `database` to the new JATOS installation.

4. If you had changed the `conf/production.conf` file in your old JATOS instance (for example to set a custom location for your `study_assets_root` folder) you'll have to do this again in the new JATOS version. We recommend re-editing the new version of the file, rather than just overwriting the new with the old version, in case anything in the `production.conf` file has changed.

5. That's it! Start the new JATOS.

What will be transferred:

1. Files and subfolders in study assets folder

2. All your study and components properties

3. All batches, together with their workers, generated links, and results

What will be lost: nothing

Data Privacy and Ethics

Data privacy is a critical issue in online studies. You should be careful when collecting, storing and handling data, regardless of which platform you use to run your studies.

We developed JATOS with data privacy in mind, preventing any breaches of the standard ethical principles in research. However, ultimately you are responsible for the data you collect and what you do with it.
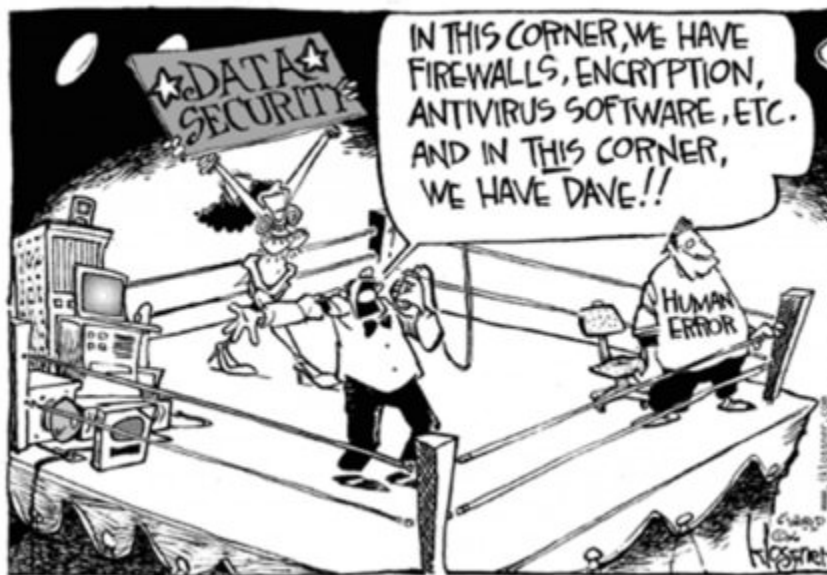
(copyright 2006 John klossner, www.jklossner.com)

Here are a few advantages and limitations of JATOS with regards to data privacy. Please read them carefully before you run any study, and please contact us (page 0) if you find that these are not sufficient, or have suggestions for improvement.

- JATOS' main advantage is that you can store your participants' data in your own server, and not in a commercial server like Amazon or Qualtrics. This means that you have full control over the data stored in your database, and no commercial company has access to it.

- By default, JATOS stores the following data:

  - time (of the server running JATOS) at which the study -and each of its components- was started and finished

  - the worker type (page 16) (MTurk, General single, Personal multiple, etc)

- in cases of MTurk workers, the confirmation code AND the MTurk worker ID. In these cases, if an MTurk worker participated in two of your studies, running in the same JATOS instance, **you will be able to associate the data across these two studies**. This is an important issue: MTurk workers might not be aware that you are the same researcher, and will not know that you have the chance to associate data from different studies. The best way to avoid this is to export all your study's data and delete it from the JATOS database once you are done with it. In this way, JATOS won't know that a worker already participated in another study and will create a new worker ID for them.

- JATOS will **not** store information like IP address or browser type. However, you could access and store this information through your JavaScripts. You could also record whether workers are actively on the browser tab running your study, or whether they have left it to go to another tab, window or program. If you collect any of these data, you should let your workers know.

- Bear in mind: Every file within your study assets folders is public to the Internet. Anybody can in principle read any file in this folder, regardless of how secure your server is. **Thus, you should never store any private data, such as participants' details in the study assets folders.**

Things you should consider in your studies

- You should consider to add some button in your study pages to abort the study. Some ethics demand that any participant should have the **right to withdraw** at any time, without explanation. In this case all data of the participant gathered during the study should be deleted. Conveniently jatos.js offers an abortStudy method (page 67).

- Use **encryption** with your server instance (page 45). Only with encryption no one else in the internet can read the private data from your study's participants.

Example Group Studies

With JATOS 2 you can run group studies. JATOS is versatile and supports fixed, two-user studies (like this Prisoner's Dilemma (https://github.com/JATOS/JATOS_examples/wiki/Example-Studies#prisoners-dilemma)) or open, multi-user studies (like this Snake game (https://github.com/JATOS/JATOS_examples/wiki/Example-Studies#snake)), and everything in between.

Interested? Try one of our pre-made example studies

The easiest way to get started is trying out some pre-made examples. For this, you'll have to run a group study as two different workers.

So you could:

1. Download the Snake game (https://github.com/JATOS/JATOS_examples/wiki/Example-Studies#snake)

2. Get a link for the first worker: Snake game -> Batch Manager -> Default batch -> Worker Setup -> Personal Multiple worker and 'Add' a new one. In the confirmation press 'Run' to start the study.

3. Get the link for the second worker: in a second browser tab (or window), go to Worker Setup in the same Default batch and create a new Personal Multiple worker. Press 'Run' to start the study.

4. In both browser tabs, click through the introduction until you arrive in the waiting room. Click 'Join' and then 'Ready'.

5. Voilà! You'll see two snakes moving around: your own and the one from the other worker.

Further reading about group studies:

- Group Study Properties (page 38)
- Write Your Own Group Studies (page 42)

Group Study Properties

> **Summary:** Lern how to set up a group study and the different ways in how to assign workers to groups.

(If you haven't already, we recommend that you try out some example group studies (page 37).)

Set up group studies

First and common to all group setups is to check the Group study checkbox in the study properties.
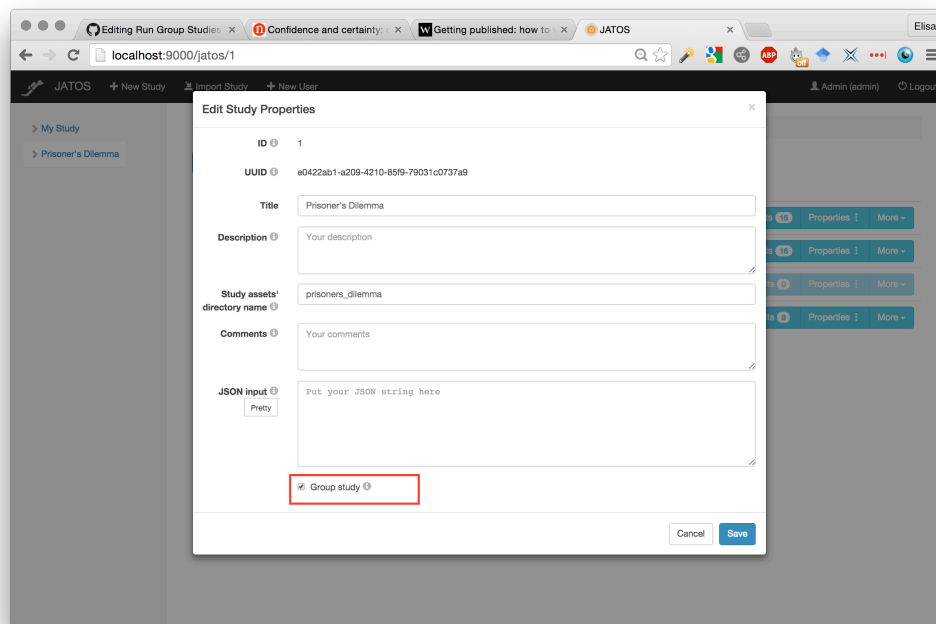
If the Group property is checked, JATOS will assign workers into groups. We'll describe some group properties that you can use to tweak according to whether you want to keep control over worker assignment, or you give JATOS full control.

Group settings in each batch's properties

You can have multiple batches in JATOS, each one with different group settings. There are three important bits of information for a group study:

1. *Max total workers*: This isn't just a property of group studies but can be used in single-worker studies too. It simply limits the total amount of workers who are allowed to run in this batch.

2. *Max total members*: This limits the number of members a single group can have. While there can be multiple groups in a batch, the *Max total members* applies to each separate group.

3. *Max active members*: This limits the number of active members a single group can have. An active member is in the group at this time - in opposite to a past member who already left the group. This number applies to each group separately. Example: In the Prisoner's Dilemma study, you would limit the active members to 2.

By default, all properties have no upper limit.

Group assignment

You can either tell JATOS to assign workers to different groups, or you can keep full control and do it yourself (or something in between). We'll use some example scenarios to explain how this assignment works.

Scenario 1: One group, assign workers manually

If in a batch you set the *Max total worker* to 2 and leave the other two Max parameters empty, JATOS has no other choice than to allow only 2 workers and sort them into the same group. If you then define two Personal Single workers and send the access links (displayed in the batch) to your two participants, you can be sure that they will interact with each other. If you need more groups, just create a second batch with two other workers.

Scenario 2: Several groups, let JATOS assign workers

Say you want to have 3 groups with 2 workers each. You want to leave it to JATOS which workers are paired together. Then, set *Max total workers* to 6 and both *Max active members* and *Max total members* to 2 (remember that these numbers apply to each group separately). Create your 6 workers in the Worker Setup (or use a General Single link) and distribute your link(s) to your workers.
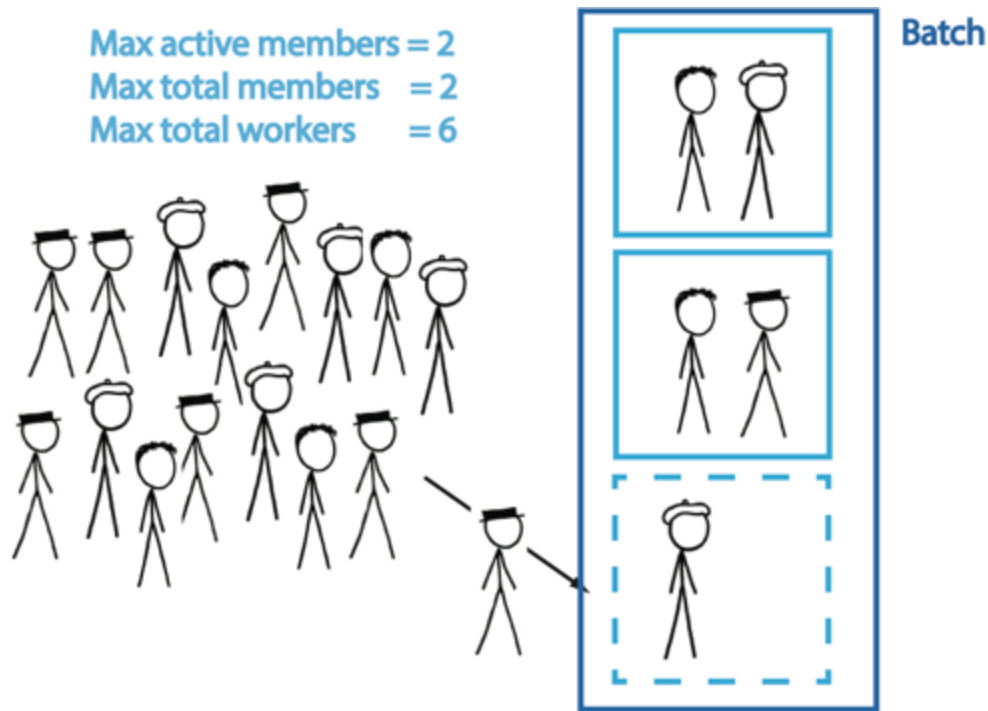
The first two scenarios may apply to the Prisoner's Dilemma Example Study (https://github.com/JATOS/JATOS_examples/wiki/Example-Studies#prisoners-dilemma).

Scenario 3: One open world

This scenario is basically the opposite of the first one. By limiting neither the *Max total worker* nor the *Max total members*, nor the *Max active members* JATOS will sort all workers in one single group that is potentially of unlimited size. Now –to keep it completely open– just create one General Single worker and publish its link (e.g. via a mailing list or on a website). But keep in mind: this way many workers might access your study at the same time and this might overload your JATOS server.

Scenario 4: Multiple open worlds with limited active members

Say you want to have groups with up to 3 members, interacting *at the same time*. But you don't want to actually limit the total number of members per group: you want to allow new workers to join a group if one of its members left. This way each group can have a flow of workers joining and leaving - the only constraint is the maximum members per group at any given time. You also want to let JATOS set the number of groups depending on the available workers. To set up this just use one batch, set the *Max active members* to 3, and leave *Max total worker* and *Max total members* unlimited.

Max active members = 3
Max total members   = [ ]
Max total workers   = [ ]

Batch
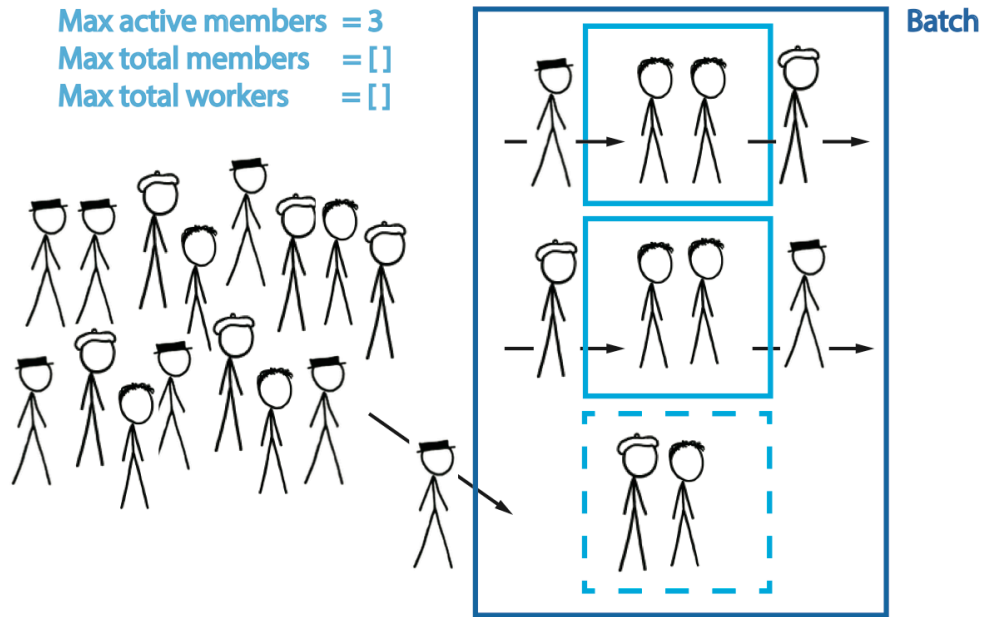
Write Your Own Group Studies

Writing JavaScripts for group studies

Group studies differ from single-worker studies simply in that the JavaScript needs to handle groups and communications between members. The jatos.js library provides some useful functions for this.

If you like to dive right into jatos.js' reference:

- jatos.js functions for group studies (page 68)

- jatos.js group variables (page 64)

- jatos.js group session (page 65)

Joining a group and opening group channels

There are two requisites for allowing group members to interact:

1. Workers can only communicate with members of their own group. So, interacting workers must all join the same group. **A worker will remain in a group until jatos.js is explicitly told to leave the group (or the study run is finished). This means that if a worker moves between components or reloads a page they will remain in the same group.** This feature makes groups much more robust.

2. Communication can only be done if a group channel is open. Although jatos.js opens and closes all necessary group channels so you don't have to care for them directly while writing components. Group channels in turn use WebSockets (https://en.wikipedia.org/wiki/WebSocket). WebSockets are supported by all modern browsers (http://caniuse.com/#feat=websockets).

So here's how a typical JATOS group study run would look like:

Component 1

- *jatos.joinGroup* -> joins group and opens group channel

- *jatos.nextComponent* -> closes group channel and jumps to next component

Component 2

- *jatos.joinGroup* -> opens group channel in the **same group**

- *jatos.nextComponent* -> closes group channel and jumps to next component

Component 3

- *jatos.joinGroup* -> opens group channel **same group**

- *jatos.endStudy* -> closes group channel, leaves group, ends component, and ends study

Notice that by calling *jatos.joinGroup (page 68)* in the second and third component JATOS does not let workers join a new group but just opens a group channel in the already joined group. To make a worker leave a group, use the function *jatos.leaveGroup* (page 69).

Reassigning to a different group

To move a worker from one group to a different one, use *jatos.reassignGroup (page 70)*. This function will make a worker leave their group and join a different one. JATOS can only reassign to a different group if there is another group available. If there is no other group JATOS will not start a new one but put the worker into the same old group again.

Fixing a group

Sometimes you want to stay with the group like it is in the moment and don't let new members join - although it would be allowed according to the group properties. For example in the Prisoner's Example study (https://github.com/JATOS/ JATOS_examples/wiki/Example-Studies#prisoners-dilemma) after the group is assembled in the waiting room component it is necessary to keep the two members as it is. Even if one of the members leaves in the middle of the game, JATOS shouldn't just assign a new member. To do this call jatos.js' function *jatos.setGroupFixed* (page 70).

Communication between group members

JATOS provides two ways for communicating within the group: direct messaging between group members or via the group session.

Direct messaging

Members can send direct messages to other members of the same group with the *jatos.sendGroupMsg* (page 69) and *jatos.sendGroupMsgTo* (page 69) functions. This way of communication is fast but can be unreliable in case of an unstable network connection. We use direct messaging in the Snake example (https://github.com/JATOS/JATOS_examples/wiki/Example-Studies#snake) to

send the coordinates of the snakes on every step. Here, speed is more critical than reliability in the messages, because a few dropped frames will probably go unnoticed.

Group session

Members can set the group session data with the *jatos.setGroupSessionData* (page 70) function. At any time the group session data are available in jatos.js' variable *jatos.groupSessionData* (page 65). The group session data are stored in JATOS' database **only while the group is active. It is deleted when the group is finished.** Communication via group session is slower, but more reliable than group messaging. If one member has an unstable internet connection or does a page reload, the group session will be automatically restored after the member reopens the group channel. Workers communicate via the group session data in the Prisoner's Example study (https://github.com/JATOS/JATOS_examples/wiki/ Example-Studies#prisoners-dilemma), because it is important that each message reaches the other player.

JATOS on a server

Summary: To run studies online, e.g. with Mechanical Turk, JATOS has to be installed on a server. Server instances of JATOS have slightly different configuration requirements than local instances. This text aims at server admins who wants to setup a server running JATOS and who know their way around server management.

There are several ways to bring JATOS to the internet. You can install it

- on your own dedicated server

- in the cloud on an Infrastructure as a Service (IaaS)

- in the cloud with a Docker container

The first two are discussed here in this page. For the last one JATOS provides a .

One word about IaaS. There are many IaaS providers (Digital Ocean, Microsoft Azure, Google Cloud, Amazon's AWS etc.). They all give you a virtual machine (VM) and the possibility to install an operating system on it. I'd recommend to go with a Linux system like Ubuntu or Debian. Another point is to make sure they have persistent storage and not what is often called 'ephemeral storage' (storage that is deleted after the VM shuts down). Since JATOS stores all study assets in the server's file system persistent storage is needed. But apart from that it's the same JATOS installation like on a dedicated server. In we have some advice in how to do it in AWS.

The actual JATOS instance on a server isn't too different from a local one. It basically involves telling JATOS which IP address and port it should use and (optionally) replace the H2 database with a MySQL one. There are other issues however, not directly related to JATOS, that you should consider when setting up a server. These include: setting up automatic, regular backups of your data, an automatic restart of JATOS after a server reboot, encryption, additional HTTP server, etc. The purpose of this page is not to teach you how to set up a server in general (you do need to know a little bit about Internet and security issues and should not try it at home unless you know what you're doing!). Here, we simply cover the steps to server setup that are related to JATOS.

Java

We've produced multiple versions of JATOS. The simplest version is JATOS alone, but other versions are bundled with Java JRE. On a server, it's best (though not necessary) to install JATOS without a bundled Java. This will make it easier to upgrade to new Java releases.

Configuration

If JATOS runs locally it's usually not necessary to change the defaults but on a server you probably want to set up the IP and port or maybe use a different database and change the path of the study assets root folder. This wiki has an extra page how to Configure JATOS on a Server (page 49).

Change admin's password

Every JATOS installation has the same default admin password 'admin'. You must change it before the server goes live. This can be done in the GUI: 1) login as 'admin', 2) click on your user in the header 3) Click 'Change Password'.

Backup

You should set up a regular backup of your data from JATOS. These include (1.) the data stored in the database and (2.) your study assets folder.

1. If you use the default H2 database you can just backup the folder `my-jatos-path/database`. In case you want to restore an older version from the backup just replace the current version of the folder with the backed-up version. If you use a MySQL database you might want to look into the `mysqldump` shell command. E.g., with `mysqldump -u myusername -p mydbname > mysql_bkp.out` you can backup the whole data into a single file. Restore the database with `mysql -u myusername -p mydbname < mysql_bkp.out`.

2. You can just make a backup of your study assets folder. If you want to return to a prior version replace the current folder with the backed-up version.

Remember, a backup has to be done of **both** the database **and** the study assets root folder.

HTTP server

JATOS doesn't need an HTTP server (like Nginx or Apache). It is its own HTTP server. But sometimes it is necessary to have an additional server in front of JATOS (e.g. for HTTPS encryption). In JATOS with Nginx (page 52) we show a sample configuration for Nginx and in JATOS with Apache (page 56) for Apache. And keep in mind that JATOS uses WebSockets for group studies and your HTTP server should be configured to support them.

Auto-start JATOS on Linux/Unix as a daemon

It's nice to have JATOS starts automatically after a start or a reboot of your machine. It's easy to turn the `loader.sh` script that you normally use to start JATOS into an init script for a daemon.

1. Copy the `loader.sh` script to `/etc/init.d/`

2. Rename it to `jatos`

3. Change access permission with `chmod og+x jatos`

4. Edit `/etc/init.d/jatos`

    a. Specify the IP address and port you want to use

    b. Comment out the line
       ```
       dir="$( cd "$( dirname "$0" )" && pwd )"
       ```

    c. Add variable `dir=` with the path to your JATOS installation

       The beginning of your `/etc/init.d/jatos` should look like:

```
#!/bin/bash
# JATOS loader for Linux and MacOS X

# Change IP address and port here
address="127.0.0.1"
port="9000"
dir="/path/to/my/JATOS/installation"

# Don't change after here unless you know what yo
u're doing
###################################
# Get JATOS directory
#dir="$( cd "$( dirname "$0" )" && pwd )"
pidfile=$dir/RUNNING_PID
```

5. Make it auto-start with the command
    ```
    sudo update-rc.d jatos defaults
    ```

Now JATOS starts automatically when you start your server and stops when you shut it down. You can also use the init script yourself like any other init script with `sudo /etc/init.d/jatos start|stop|restart` .

Encryption and HTTPS

Most admins tend to use an additional HTTP server in front of JATOS for encryption purpose. But since JATOS is built with the Play Framework it can be setup JATOS directly by following their Documentation about Configuring HTTPS (https://www.playframework.com/documentation/2.4.x/ConfiguringHttps).

JVM Memory

If your JATOS instance has a high worker load or runs many different studies, you might have to increase the memory that is available to the JVM that runs JATOS. A clear sign that your JVM needs more memory is a `java.lang.OutOfMemoryError` in your log. To increase the maximal available memory use the `-Xmx` parameter of the JVM. You can set it in the environment variable JAVA_OPTS. E.g. to set it to 2048 MB use `export set JAVA_OPTS="-Xmx2048m"` on a Linux/MacOS X system.

Configure JATOS on a Server

Summary: If JATOS runs locally it's usually not necessary to change the defaults. On a server, you probably will want to set up the IP and port, or use a different database and change the path of the study assets root folder.

Restart JATOS after making any changes to the configuration (`loader.sh restart`)

IP / domain and port

By default JATOS uses the address 127.0.0.1 and port 9000. There are two ways to configure the host name or IP address and the port:

1. In `loader.sh` change the values of 'address' and 'port' according to your IP address or domain name and port.

```
address="172.16.0.1"
port="8080"
```

2. Via command-line arguments `-Dhttp.address` and `-Dhttp.port` , e.g. with the following command you'd start JATOS with IP 10.0.0.1 and port 80

```
loader.sh start -Dhttp.address=10.0.0.1 -Dhttp.port=80
```

Study assets root path

By default the study assets root folder (where all your study's HTML, JavaScript files etc. are stored) is located within JATOS installation's folder in `study_assets_root` . There are three ways to change this path:

1. Via the command-line argument `-DJATOS_STUDY_ASSETS_ROOT_PATH` , e.g.

```
loader.sh start -DJATOS_STUDY_ASSETS_ROOT_PATH="/path/t
o/my/assets/root/folder"
```

2. Via `conf/production.conf` : change `jatos.studyAssetsRootPath`

```
jatos.studyAssetsRootPath="/path/to/my/jatos_study_asset
s_root"
```

3. Via the environment variable `JATOS_STUDY_ASSETS_ROOT_PATH` , e.g. the following export adds it to the env variables:

```
export JATOS_STUDY_ASSETS_ROOT_PATH="/path/to/my/assets/
root/folder"
```

Database

By default JATOS uses an embedded H2 database, but it can be easily configured to work with an external H2 or a MySQL database.

You can confirm that JATOS is accessing the correct database by looking in the logs. One of the lines after JATOS starts should look like this (with your JDBC URL).

```
19:03:42.000 [info] - p.a.d.DefaultDBApi - Database [default] c
onnected at jdbc:mysql://localhost/jatos?characterEncoding=UT
F-8
```

JATOS requires MySQL >= 5.5 or H2 >= 1.4.192 (prior versions might work - I just never tested)

Note: We tried JATOS extensively with the H2 database. It's reliable and doesn't have some issues that do exist with MySQL databases such as this one (https://github.com/JATOS/JATOS/issues/111).

There are three ways to set up the database.

1. Via command-line arguments:

   • `-DJATOS_DB_URL` - specifies the JDBC URL to the database

   • `-DJATOS_DB_USERNAME` and `-DJATOS_DB_PASSWORD` - set username and password

- `-DJATOS_DB_DRIVER` - **can be either** `org.h2.Driver` **or** `com.mysql.jdbc.Driver`

- `-DJATOS_JPA` - **can be either** `h2PersistenceUnit` **or** `mysqlPersistenceUnit`

E.g. to connect to a MySQL database running on 172.17.0.2 and a table named 'jatos' use:

```
loader.sh start -DJATOS_DB_URL='jdbc:mysql://172.17.0.2/
jatos?characterEncoding=UTF-8' -DJATOS_DB_USERNAME=sa -D
JATOS_DB_PASSWORD=sa -DJATOS_JPA=mysqlPersistenceUnit -D
JATOS_DB_DRIVER=com.mysql.jdbc.Driver
```

2. Via production.conf (description analog to 1.)

```
db.default.url="jdbc:mysql://localhost/MyDatabase?charac
terEncoding=UTF-8"
db.default.user=myusername
db.default.password=mypassword
db.default.driver=com.mysql.jdbc.Driver
jpa.default=mysqlPersistenceUnit
```

3. Via environment variables (description analog to 1.)

- `JATOS_DB_URL`

- `JATOS_DB_USERNAME`

- `JATOS_DB_PASSWORD`

- `JATOS_DB_DRIVER`

- `JATOS_JPA`

E.g. to set all database environment variables for a MySQL database and table called 'jatos' you could use a command (change the values):

```
export JATOS_DB_URL='jdbc:mysql://localhost/jatos?charac
terEncoding=UTF-8' JATOS_DB_USERNAME='jatosuser' JATOS_D
B_PASSWORD='mypassword' JATOS_DB_DRIVER=com.mysql.jdbc.D
river JATOS_JPA=mysqlPersistenceUnit
```

JATOS with Nginx

This is an example for a configuration of Nginx (https://www.nginx.com/) as a proxy in front of JATOS. It is not necessary to run JATOS with a proxy but it's common. It supports encryption (HTTPS) and WebSockets for JATOS' group studies.

The following is the content of `/etc/nginx/nginx.conf` . Change it to your needs. You probably want to change your servers address (`www.example.com` in the example) and the path to the SSL certificate and its key.

```
user www-data;
worker_processes 1;
pid /run/nginx.pid;

events {
        worker_connections 768;
        # multi_accept on;
}

http {
        sendfile on;
        keepalive_timeout 65;
        client_max_body_size 500M;

        include /etc/nginx/mime.types;
        default_type application/octet-stream;

        proxy_buffering    off;
        proxy_set_header   X-Real-IP $remote_addr;
        proxy_set_header   X-Forwarded-Proto https;
        proxy_set_header   X-Forwarded-Ssl on;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forward
ed_for;
        proxy_set_header   Host $http_host;
        proxy_http_version 1.1;

        upstream jatos-backend {
                server 127.0.0.1:9000;
        }

        # needed for websockets
        map $http_upgrade $connection_upgrade {
                default upgrade;
                ''      close;
        }

        # redirect http to https
        server {
                listen      80;
                server_name www.example.com;
                rewrite ^ https://www.example.com$request_uri?
permanent;
        }

        server {
                listen              443;
                ssl                 on;
```

```
                # http://www.selfsignedcertificate.com/ is usef
ul for development testing
                ssl_certificate      /etc/ssl/certs/localhost.c
rt;
                ssl_certificate_key  /etc/ssl/private/localhos
t.key;

                # From https://bettercrypto.org/static/applie
d-crypto-hardening.pdf
                ssl_prefer_server_ciphers on;
                ssl_protocols TLSv1 TLSv1.1 TLSv1.2; # not poss
ible to do exclusive
                ssl_ciphers 'EDH+CAMELLIA:EDH+aRSA:EECDH+aRSA+A
ESGCM:EECDH+aRSA+SHA384:EECDH+aRSA+SHA256:EECDH:+CAMELLIA256:+A
ES256:+CAMELLIA128:+AES128:+SSLv3:!aNULL:!eNULL:!LOW:!3DES:!MD
5:!EXP:!PSK:!DSS:!RC4:!SEED:!ECDSA:CAMELLIA256-SHA:AES256-SHA:C
AMELLIA128-SHA:AES128-SHA';
                add_header Strict-Transport-Security "max-age=1
5768000; includeSubDomains";

                keepalive_timeout    70;
                server_name www.example.com;

                # websocket location (JATOS' group channel)
                location ~ "^/publix/[\d]+/group/join" {
                        proxy_pass              http://jatos-ba
ckend;
                        proxy_set_header        Upgrade $http_u
pgrade;
                        proxy_set_header        Connection $con
nection_upgrade;
                        proxy_read_timeout      3600; # keep op
en even without any transmission
                }

                # all other traffic
                location / {
                        proxy_pass              http://jatos-ba
ckend;
                }
        }

        access_log /var/log/nginx/access.log;
        error_log /var/log/nginx/error.log;

        include /etc/nginx/conf.d/*.conf;
```

```
        include /etc/nginx/sites-enabled/*;
}
```

JATOS with Apache

This is an example for a configuration of Apache (https://httpd.apache.org/) as a proxy in front of JATOS. It is not necessary to run JATOS with a proxy but it's common.

The following is the content of Apache's `httpd.conf`. Change it to your needs. You probably want to change your servers address (`www.example.com` in the example). If you want to use JATOS with group studies you have to add the mod_proxy_wstunnel module (https://httpd.apache.org/docs/2.4/mod/mod_proxy_wstunnel.html).

```
LoadModule proxy_module modules/mod_proxy.so
…
<VirtualHost *:80>
  ProxyPreserveHost On
  ServerName www.example.com
  ProxyPass  /excluded !
  ProxyPass / http://127.0.0.1:9000/
  ProxyPassReverse / http://127.0.0.1:9000/
</VirtualHost>
```

Install JATOS via Docker

JATOS has a Docker image: hub.docker.com/r/jatos/jatos/ (https://hub.docker.com/r/jatos/jatos/)

Docker is a great technology, but if you never heard of it you can safely ignore this page (it's not necessary to use it if you want to install JATOS, either locally or on a server).

If you have heard of it, you might want to, for example, setup JATOS in a Docker container together with an external H2 or MySQL database in another Docker container. You can easily do that:

Install JATOS locally with a Docker container

1. Install Docker locally on your computer (not covered here)

2. Go to your shell or command line interface

3. Pull the JATOS image: use `docker pull jatos/jatos:x.x.x` and specify the release, e.g. to get version 2.2.4 use
   `docker pull jatos/jatos:2.2.4`

4. Check that you actually downloaded the image: `docker images` should show `jatos/jatos` in one line

5. Now run JATOS (and create a new Docker container) with
   `docker run -d -p 9000:9000 jatos/jatos:x.x.x`, e.g. to for version 2.2.4 use `docker run -d -p 9000:9000 jatos/jatos:2.2.4`. The `-d` argument specifies to run this container as a daemon and the `-p` is responsible for the port mapping.

6. Check that the new container is running: In your browser go to localhost:9000 (http://localhost:9000) - it should show the JATOS login screen. Or use `docker ps -a` - in the line with `jatos/jatos` the status should say `up`.

Troubleshooting: By removing the `-d` argument (e.g.
`docker run -p 9000:9000 jatos/jatos:2.2.4`) you get JATOS' logs printed in your shell - although you don't run it as a daemon in the background anymore.

Change port

With Docker you can easily change JATOS' port (actually we change the port mapping of JATOS' docker container). Just use Docker `-p` argument and specify your port. E.g. to run JATOS on port 8080 use

`docker run -d -p 8080:9000 jatos/jatos:2.2.4`.

Configure with environment variables

All environment variables that can be used to configure a normal JATOS server installation (page 49) can be used in a docker installation. Just use Docker's `-e` argument to set them.

E.g. to setup JATOS with a MySQL database running under IP `172.17.0.2` that uses the table `jatos` use the following command (but change the JATOS version and use username and password of your MySQL account):

```
docker run -e JATOS_DB_URL='jdbc:mysql://172.17.0.2/jatos?chara
cterEncoding=UTF-8' -e JATOS_DB_USERNAME='root' -e JATOS_DB_PAS
SWORD='password' -e JATOS_DB_DRIVER=com.mysql.jdbc.Driver -e JA
TOS_JPA=mysqlPersistenceUnit -p 9000:9000 jatos/jatos:2.2.4
```

JATOS in Amazon's Cloud (without Docker)

On this page is additional information in how to install JATOS on a server in Amazon's Web Services. All general installation advice is in JATOS on a server (page 45) and applies here too.

1. First you need to register at AWS (Amazon Web Services) (https://aws.amazon.com/) (they'll want your credit card).

2. In AWS webpage move to EC2 and launch a new instance with Ubuntu (you can use other Linux too - I tested it with Ubuntu 14.04)

3. During the creation of the new EC2 instance you will be ask whether you want to create a key pair. Do so. Download the file with the key (a *.pem file). Store it in a safe place - with this key you will access your server.

4. Login via SSH:
   `ssh -i /path/to/your/pem_key_file ubuntu@xx.xx.xx.xx` (Use your instance's IP address: In AWS / EC2 / Instances / Description are two IPs 'Private IP' and 'Public IP'. Use the **public** one.)

5. Get the latest JATOS bundled with Java
   `wget https://github.com/JATOS/JATOS/releases/download/v2.1.12-beta/jatos-x.x.x-li`

6. `unzip jatos-x.x.x-linux_java.zip` (You probably have to install 'unzip' first with `sudo apt-get install unzip`.)

7. Configure IP and port in `jatos.sh`: Use the '**Private IP**' from your instance description (the one that starts with 172.x.x.x) and port 80

8. (Optional) make JATOS auto-start (page 47)

9. Change JATOS' admin password

Updating a JATOS server installation

Updating the server instance is equivalent to doing it locally (page 33), but make sure that you know what you're doing; especially if you have paired JATOS with a MySQL database.

To be absolutely safe you can install the new JATOS version and keep the old one untouched. This way you can switch back if something fails. Just remember that only one JATOS can run at the same time.

Note: If you are using a MySQL database and to be on the safe side in case something goes wrong, make a backup of your MySQL database. Dump the database using

` mysqldump -u yourUserName -p yourDatabaseName > yourDatabaseName.out `.

As when updating of a local JATOS installation (page 33) you have two options: 1. Keep your studies but discard all your result data and batches. 2. Keep everything, including your studies and result data (might not always be possible).

First option: quick and dirty (discarding result data)

You can just follow the update instructions for the local installation (page 33). If you use a mySQL database don't forget to configure it with a clean and new one (page 49) (not the one from your old JATOS). Do not use the new JATOS with the old MySQL database unless you choose to keep your data, as described below.

Second option: keeping everything

This means that we have to configure the MySQL database or copy the H2 database files.

However: Occasionally (on major version updates such as 1.X.X to 2.1.1) we will make big changes in JATOS, which will require some restructuring of the database. In these cases, the second way described here will not be possible. We will do our best to prevent these big changes and inform you explicitly in the release description.

1. Stop the old JATOS using ` ./loader.sh stop `

2. Copy the new JATOS version to your server, e.g. copy it into the same folder where your old JATOS is located. Don't yet remove the old JATOS instance.

3. Unzip the new JATOS (` unzip jatos-x.x.x-beta.zip `)

4. From the old JATOS installation copy your assets root folder to the new JATOS installation (Note: By default your assets root folder is called `study_assets_root` and lays in the JATOS folder but you might have changed this (page 49).

5. If you are using the default H2 database: From your the folder of your old JATOS installation copy the folder `database` to the new JATOS installation. For an external MySQL or H2 database you don't have to change anything on the database side.

6. Configure the new JATOS like the old one (page 49)

7. That's it! Start the new JATOS using `./loader.sh start`

jatos.js Reference

> **Summary:** jatos.js is a small JavaScript library that helps you to communicate from JavaScript with your JATOS server (in more technical terms, it wraps calls to JATOS' public REST API). Below we list and describe the variables and functions of the jatos.js library.

Two bits of code (page 20) are mandatory in all your components' HTML files. One of them is the following line in the head section:
`<script src="/assets/javascripts/jatos.js"></script>` which includes the jatos.js library into your HTML file.

All variables or calls to jatos.js start with `jatos.` . For example, if you want to get the study's ID you can use `jatos.studyId` . If you want to submit some result data of your component back to your JATOS server you can call `jatos.submitResultData(resultData)` , where resultData can be any kind text.

And, please, if you find a mistake or have a question don't hesitate to contact us (page 0).

Below we list and describe the variables and functions of the jatos.js library:

jatos.js variables

IDs

You can call any of these variables below at any point in your HTML file after `jatos.onload()` is done. All those IDs are generated and stored by JATOS. jatos.js automatically sets these variables with the corresponding values if you included the `jatos.onLoad()` callback function at the beginning of your JavaScript.

- `jatos.studyId` - ID of the study which is currently running. All the study properties are associated with this ID.

- `jatos.componentId` - ID of the component which is currently running. All the component properties are associated with this ID.

- `jatos.batchId` - ID of the batch this study run belongs to. All batch properties are associated with this ID.

- `jatos.workerId` - Each worker who is running a study has an ID.

- `jatos.studyResultId` - This ID is individual for every study run. A study result contains data belonging to the run in general (e.g. study session).

- `jatos.componentResultId` - This ID is individual for every component in a study run. A component result contains data of the run belonging to the specific component (e.g. result data).

There's a convenient function that adds all these IDs to a given object. See function `jatos.addJatosIds(obj)` below.

Study variables

- `jatos.studyProperties` - All the properties (except the JSON input data) you entered for this study

  - `jatos.studyProperties.title` - Study's title

  - `jatos.studyProperties.uuid` - Study's UUID

  - `jatos.studyProperties.description` - Study's description

  - `jatos.studyProperties.locked` - Whether the study is locked or not

  - `jatos.studyProperties.dirName` - Study's dir name in the file system of your JATOS installation

  - `jatos.studyProperties.groupStudy` - Whether this is a group study or not

- `jatos.studyJsonInput` - The JSON input you entered in the study's properties.

- `jatos.studyLength` - Number of component this study has

Component variables

- `jatos.componentProperties` - All the properties (except the JSON input data) you entered for this component

  - `jatos.componentProperties.title` - Component's title

  - `jatos.componentProperties.uuid` - Component's UUID

  - `jatos.componentProperties.htmlFilePath` - Path to Component's HTML file in your JATOS installation

  - `jatos.componentProperties.reloadable` - Whether it's reloadable

- `jatos.componentJsonInput` - The JSON input you entered in the component's properties.

- `jatos.componentList` - An array of all components of this study with basic information about each component. For each component it has the `title`, `id`, whether it is `active`, and whether it is `reloadable`.

- `jatos.componentPos` - Position of this component within the study starting with 1 (like shown in the GUI)

Study's session data

The session data can be accessed and modified by every component of a study. It's a very convenient way to share data between different components. However, remember that the session data will be deleted after the study is finished (see also Two Types of Session Data (page 19)).

- `jatos.studySessionData`

Batch variables

- `jatos.batchProperties` - All the properties you entered for this batch.

  ○ `jatos.batchProperties.allowedWorkerTypes` - List of worker types that are currently allowed to run in this batch.

  ○ `jatos.batchProperties.maxActiveMembers` - How many members this group can have at the same time

  ○ `jatos.batchProperties.maxTotalMembers` - How many members this group is allowed to have at the same time

  ○ `jatos.batchProperties.maxTotalWorkers` - Total amount of workers this group is allowed to have altogether in this batch

  ○ `jatos.batchProperties.title` - Title of this batch

Group variables

The group variables are part of jatos.js since JATOS 2. They are only filled with values if the current study is a group study.

- `jatos.groupMemberId` - Group member ID is unique for this member (it is actually identical with the study result ID)

- `jatos.groupResultId` - ID of this group result (It's called group result to be consistent with the study result and the component result - although often it's just called group)

- `jatos.groupState` - Represents the state of the group in JATOS; only set if group channel is open (one of STARTED, FIXED, FINISHED)

- `jatos.groupMembers` - List of member IDs of the current members of the

group

- `jatos.groupChannels` - List of member IDs of the currently open group channels

Group's session data

- `jatos.groupSessionData` - Group session data shared in between members of the group (see also Two Types of Session Data (page 19))

Other variables

- `jatos.version` - Current version of the jatos.js library

General jatos.js functions

`jatos.onLoad(callback)`

Defines callback function that jatos.js will call when it's finished initialising. Only mandatory (page 20) call in every component.

`jatos.onError(callback)`

Defines callback function that is to be called in case jatos.js produces an error.

`jatos.log(logMsg)`

Logs an msg in the log of the JATOS installation.

- *param {String} logMsg* - The messages to be logged

`jatos.addJatosIds(obj)`

Convenience function that adds some IDs (page 62) (study ID, study title, component ID, component position, component title, worker ID, study result ID, component result ID, group result ID, group member ID) to the given object.

- *param {Object} obj* - Object to which the IDs will be added

Functions to control study flow

`jatos.startComponent(componentId)`

Starts the component with the given ID. You can pass on information to the next component by adding a query string.

- *param {Object} componentId* - ID of the component to start

`jatos.startComponentByPos(componentPos)`

Starts the component with the given position (# of component within study).

- *param {Object} componentPos* - Position of the component to start

`jatos.startNextComponent()`

Starts the next component of this study. The next component is the one with position + 1.

`jatos.startLastComponent()`

Starts the last component of this study or if it's inactive the component with the highest position that is active.

`jatos.endComponent(successful, errorMsg, onSuccess, onError)`

Finishes component. Usually this is not necessary because the last component is automatically finished if the new component is started. Nevertheless it's useful to explicitly tell about a FAIL and submit an error message. Finishing the component doesn't finish the study.

- *param {optional Boolean} successful* - 'true' if study should finish successful and the participant should get the confirmation code - 'false' otherwise.

- *param {optional String} errorMsg* - Error message that should be logged.

- *param {optional Function} onSuccess* - Function to be called in case of successful submit

- *param {optional Function} onError* - Function to be called in case of error

`jatos.abortStudyAjax(message, success, error)`

Aborts study. All previously submitted data will be deleted.

- *param {optional String} message* - Message that should be logged

- *param {optional Function} success* - Function to be called in case of successful submit

- *param {optional Function} error* - Function to be called in case of error

```
jatos.abortStudy(message)
```

Aborts study. All previously submitted data will be deleted.

- *param {optional String} message* - Message that should be logged

```
jatos.endStudyAjax(successful, errorMsg, onSuccess, onError)
```

Ends study with an Ajax call.

- *param {optional Boolean} successful* - 'true' if study should finish successful and the participant should get the confirmation code - 'false' otherwise.

- *param {optional String} errorMsg* - Error message that should be logged.

- *param {optional Function} onSuccess* - Function to be called in case of successful submit

- *param {optional Function} onError* - Function to be called in case of error

```
jatos.endStudy(successful, errorMsg)
```

Ends study.

- *param {optional Boolean} successful* - 'true' if study should finish successfully, 'false' otherwise.

- *param {optional String} errorMsg* - Error message that should be logged.

Functions for session and result

```
jatos.submitResultData(resultData, onSuccess, onError)
```

Posts resultData back to the JATOS server.

- *param {Object} resultData* - String to be submitted

- *param {optional Function} success* - Function to be called in case of successful submit

- *param {optional Function} error* - Function to be called in case of error

```
jatos.setStudySessionData(sessionData, complete)
```

Posts study session data to the JATOS server. This function is called automatically in the end of a component's life cycle (it's called by all jatos.js functions that start or end a component). So unless you want to store the session data in between a component run it's not necessary to call this function manually (in opposite to the group session's *jatos.setGroupSessionData*).

- *param {Object} sessionData* - Object to be submitted

- *param {optional Function} complete* - Function to be called after this function is finished

Functions for group studies

```
jatos.joinGroup(callbacks)
```

Tries to join a group (actually a group result) and if it succeeds opens the group channel's WebSocket.

- *param {Object} callbacks* - Defining callback functions for group events. All callbacks are optional. These callbacks functions can be:

   ◦ **onOpen**: Is called when the group channel is successfully opened

   ◦ **onClose**: Is be called when the group channel is closed

   ◦ **onError**: Is called if an error during opening of the group channel's WebSocket occurs or if an error is received via the group channel (e.g. the group session data couldn't be updated). If this function is not defined jatos.js will try to call the global *onJatosError* function.

   ◦ **onMessage(msg)**: Is called if a message from another group member is received. It gets the message as a parameter.

   ◦ **onMemberJoin(memberId)**: Is called when another member (not the worker running this study) joined the group. It gets the group member ID as a parameter.

   ◦ **onMemberOpen(memberId)**: Is called when another member (not the worker running this study) opened a group channel. It gets the group member ID as a parameter.

   ◦ **onMemberLeave(memberId)**: Is called when another member (not the worker running his study) left the group. It gets the group

member ID as a parameter.

- ◦ **onMemberClose(memberId)**: Is called when another member (not the worker running this study) closed his group channel. It gets the group member ID as a parameter.

- ◦ **onGroupSession(groupSessionData)**: Is called when the group session is updated. It gets the new group session data as a parameter.

- ◦ **onUpdate()**: Combines several other callbacks. It's called if one of the following is called: *onMemberJoin*, *onMemberOpen*, *onMemberLeave*, *onMemberClose*, or *onGroupSession* (the group session can then be read via *jatos.groupSessionData*).

`jatos.sendGroupMsg(msg)`

Sends a message to all group members with an open group channel. Like with most transmissions in the Internet these messages are send on a best effort basis. This means that if everything (e.g. network, browser, script) works fine the message gets delivered - but if the message transmission encounters a problem and is not delivered neither the sender nor the receiver will be notified. If you want more reliable message transmission use the group session and *jatos.setGroupSessionData* instead. Compared to the transmission via group session the group messaging is fast but less reliable.

- *param {Object} msg* - Any JavaScript object

`jatos.sendGroupMsgTo(recipient, msg)`

Sends a message to a single group member specified with the given group member ID (only if group channel is open).

- *param {String} recipient* - Recipient's group member ID

- *param {Object} msg* - Any JavaScript object

`jatos.leaveGroup(onSuccess, onError)`

Tries to leave the group (actually a group result) it has previously joined. The group channel is not closed in this function - it's closed from the JATOS' side.

- *param {optional Function} onSuccess* - Function to be called after the group is left

- *param {optional Function} onError* - Function to be called in case of error

```
jatos.reassignGroup(onSuccess, onFail)
```

Asks the JATOS server to reassign this study run to a different group.

- *param {optional Function} onSuccess* - Function to be called if the reassignment was successful

- *param {optional Function} onFail* - Function to be called if the reassignment was unsuccessful

```
jatos.setGroupSessionData(groupSessionData, onError)
```

Sends the group session data via the group channel to the JATOS server where it's stored and broadcasted to all members of this group. It either takes an Object as parameter or uses *jatos.groupSessionData* if the groupSessionData isn't provided. jatos.js tries several times to upload the session data, but if there are many concurrent members updating at the same time it might fail. But jatos.js/ JATOS guarantees that it either persists the updated session data or calls the *onError* callback. In this way it is more reliable but slower compared to *jatos.sendGroupMsg* or *jatos.sendGroupMsgTo*. Since the group session is stored in the JATOS server it can be retrieved after a page reload or Internet connection problem to continue at the point of the interruption.

- *param {optional Object} groupSessionData* - An object in JSON; If it's not given take *jatos.groupSessionData*

- *param {optional Object} onError* - Function to be called if this upload was unsuccessful

```
jatos.setGroupFixed()
```

Ask the JATOS server to fix this group. A fixed group is not allowed to take on more members although members are still allowed to leave.

```
jatos.hasJoinedGroup()
```

Returns true if this study run joined a group and false otherwise. It doesn't necessarily mean that we have an open group channel. We can have joined a group in a prior component. If you want to check for an open group channel use *jatos.hasOpenGroupChannel*.

`jatos.hasOpenGroupChannel()`

Returns true if we currently have an open group channel and false otherwise. Since you can't open a group channel without joining a group, it also means that we joined a group.

`jatos.isMaxActiveMemberReached()`

Returns true if the group has reached the maximum amount of active members like specified in the batch properties. It's not necessary that each member has an open group channel.

`jatos.isMaxActiveMemberOpen()`

Returns true if the group has reached the maximum amount of active members like specified in the batch properties and each member has an open group channel.

`jatos.isGroupOpen()`

Returns true if all active members of the group have an open group channel. It's not necessary that the group has reached its minimum or maximum active member size.