






This repository ▾ Search or type a command 🔍

[Explore](#) [Gist](#) [Blog](#) [Help](#)

 mmberg   

 mmberg / nadia

[Unwatch](#) ▾ 3 [Star](#) 0 [Fork](#) 0

[Home](#) [Pages](#) [History](#)[New Page](#)

# Nadia Introduction and First Steps

[Edit Page](#)[Page History](#)[Clone URL](#)

Nadia stands for Natural Dialogue System and is a set of components that deals with the creation of spoken dialogue systems. While common standards like VoiceXML are widely used in industry, it is still quite a challenge to design dialogues that make use of well established theories from research projects. Although it is easy to create simple dialogues, we cannot specify the dialogue strategy, we have no integrated support for language generation and we don't have parsers for common question types at hand. Moreover, we need a complex IVR architecture which means a lot of effort to setup a dialogue development environment on restricted university computers. Alternatively, it is possible to create a dialogue system completely manually based on speech recognition/synthesis APIs like the Microsoft SAPI. Again, there is no support regarding the specification of the dialogue and its behaviour.

Hence, the goal of Nadia is to offer an approach to design dialogues without having to code parsers, dialogue strategies or complex behaviour. The basis is a dialogue model that can be automatically processed by Nadia. The task of the dialogue designer is to only specify the questions and their properties. You will learn more about this model later. Currently, the main usage of Nadia is in dialogue system related university courses.

## Run Modes

Nadia comes as a jar-file that can be run locally. We offer two run modes that can be specified by command line arguments:

- console
- embedded REST service (Jetty)

Moreover, we offer a war-file that can be deployed on a Jetty or Tomcat server.

## Command Line Arguments

```
usage: nadia [-f] [-h] [-i] [-r] [-s]
```

```
-f,--file => specify dialogue path and file, e.g. -f /path/to/dialogue1.xml
```

The root of the path refers to the root of the war-file or the directory where the jar resides in

```
-h,--help => print this message
```

```
-i,--interface => select user interface, e.g. -i rest or -i console
```

```
-r,--resource => load dialogue (by name) from resources, e.g. -r dialogue1
```

No path required. Indicate a file name (without extension) that exists in the dialogue resources directory /res/dialogues.

```
-s,--store => load dialogue (by name) from internal store, e.g. -s dialogue1
```

Loads an in-memory dialogue, i.e. a dialogue that has been designed not as XML but as a Java Object within Nadia and given a name.

## REST

If you have started Nadia as a REST service (locally or on an Java Application Server), you can test the application by calling the URL `http(s)://SERVER:PORT/nadia`, e.g. `https://localhost:8080/nadia`. Please note that the embedded server uses https while the deployed version only uses http (if you don't configure it otherwise).

## Communication

As a first step, any client needs to create a new dialogue instance on the server. This is done by a call to `/nadia/engine/dialog` (we always assume the base URL `http(s)://SERVER:PORT`, e.g. `https://localhost:8080` which would result in `https://localhost:8080/nadia/engine/dialog`). The system will return a new URL (e.g. `/nadia/engine/dialog/d1`) as a `Location` response header that the client must use for every

subsequent communication. This url refers to the created dialogue and adds the dialogue ID, e.g. `d1`. Moreover, the first system utterance is sent to the client in the parameter `systemUtterance`. The first mentioned URL loads the default dialogue. Alternatively you could call </nadia/engine/dialog/load> and pass the content of a Nadia XML Dialogue Description as a `dialogxml` form parameter (file upload).

In the next step, the user wants to answer the question. You will need to get the text from the user and send it to the received URL in the parameter `userUtterance`.

Once a dialogue has been created, you can get more detailed information on

- the loaded dialogue: [/nadia/engine/dialog/DIALOG\\_ID/xml](/nadia/engine/dialog/DIALOG_ID/xml), e.g. </nadia/engine/dialog/d1/xml>
- the status of the dialogue: [/nadia/engine/dialog/DIALOG\\_ID/context](/nadia/engine/dialog/DIALOG_ID/context)

Moreover, you can get information about the active sessions under </nadia/engine/status>.

## cURL

If you want to test the communication process without programming your own client or using the existing web-interface, you may use `curl` like so:

1. First start the server with the embedded REST interface: `java -jar nadia.jar -i rest`
  - Alternatively you may start the REST service by deploying the war-file to your application server
2. Initialise the dialogue with `curl -k -i -X POST https://localhost:8080/nadia/engine/dialog`
  - the parameter `k` disables SSL verification and may be necessary in case of an unsigned SSL certificate
  - the parameter `i` displays not only the content but also the header. Remember that we are interested in the location.
  - please note that we send a POST request without any data. This can also be done with `curl -k -i --data "" https://localhost:8080/nadia/engine/dialog`.
  - you will receive a system utterance (i.e. the first system question), e.g. "How may I help you?"
3. Send your answer, i.e. the user utterance, to the new URL that you received in the Location parameter in the last step, e.g. `curl -k --data "userUtterance=I+want+to+book+a+trip" https://localhost:8080/nadia/engine/dialog/d1`
  - please note that the data should be URL-encoded (separate parameters by `&` and replace spaces by `+`)
  - you should see the next system question, e.g. "Where do you want to start?"
4. Answer the next question as in 3. `curl -k --data "userUtterance=I+want+to+start+in+Edinburgh" https://localhost:8080/nadia/engine/dialog/d1`
5. Have a look at the current dialogue context <https://localhost:8080/nadia/engine/dialog/d1/context>
6. Repeat 4. until the dialogue is finished or you want to stop...
  - please note that your dialogue may have been destroyed if it had not been accessed in a while
  - to see all the current sessions call <https://localhost:8080/nadia/engine/status>

Last edited by Markus Berg, 4 months ago

