Organization of this Guide

A `ziniki` command

The `ziniki` command allows developers to start a local Ziniki instance from the command line.

Ziniki startup is controlled by configuration files. These JSON files control which Ziniki components are configured into a runtime system and how they interact. It is possible to configure Ziniki from scratch by specifying the configuration of all the desired components; however, it is generally easier (and more reliable) to select from some prepackaged options

A.1 Prepackaged configurations

It is not possible to start `ziniki` from the command line without any arguments. Even the simplest prepackaged configuration requires a number of parameters to be specified, such as the hostname and ports it will listen on, and a TLS certificate to handle security. These are the parameters which must be set to identify the environment in which Ziniki will be operated.

An example parameter file is provided in the file `config/env/domain.com.json`, and the parameters specified here are described in full in the section "Environment Parameters" below.

Every configuration file to be passed to Ziniki needs to be specified with the `--configFile` option which takes an appropriate path. Thus, the simplest way to start Ziniki is to use the example configuration file:

```
bin/ziniki --configFile config/env/domain.test.json
```

Initially, this will not work because the Ziniki needs TLS in order to function, and the configuration file references a non-existent PKCS12 keystore, `keys/domain.test.p12`. This file needs to be created using `ziniki-cert` before Ziniki will start. You may, however, wish to make other changes: this file uses the domain `domain.test`, which may not be what you want. If you want to run multiple Ziniki servers on the same node, you will need to change the port numbers, etc.

If you want to change the name of the domain, it is recommended to copy all of the parameters to a new file reflecting the name of the domain.

A.2 Common Options

There are a number of common options that control how Ziniki is run independent of the configuration.

Because Ziniki is a Java program, it is possible to specify properties to the Java virtual machine. Some of these control underlying behaviors (such as the versions of TLS that can be used) and others can be used to control aspects of Ziniki which cannot be configured using the standard configuration file mechanism.

All parameters specified on the `ziniki` command line beginning with `-D` are collected and passed directly to the Java virtual machine without any checks.

Specify a Java Properties file to configure the tracing mechanism in Ziniki. By default, tracing will be sent to the console and only warning and error messages will be displayed.

If neither the `--custom` or `--trace` options are used, then this specifies that the default tracing will be used, but messages will be issued at the `INFO` level.

If neither the `--custom` or `--trace` options are used, then this specifies that the default tracing will be used, but messages will be issued at the `DEBUG` level.

A.3 Packaging Options

By default, Ziniki starts up as a single process, operating entirely in memory and offering Ziniki, admin, authentication and identity services as well as a content store (all on different ports). The packaging options enable all of these configurations to be changed fairly easily, although it may require additional options to be provided in the environment.

Offer the Ziniki service. Although this is provided by default, this option is regressive: specifying it turns off the default offering and means that (unless they are also explicitly specified) the authentication and identity services will *not* be offered.

Offer the authentication service. Although this is provided by default, this option is regressive: specifying it turns off the default offering and means that (unless they are also explicitly specified) the Ziniki and identity services will *not* be offered.

Offer the identity service. Although this is provided by default, this option is regressive: specifying it turns off the default offering and means that (unless they are also explicitly specified) the Ziniki and authentication services will *not* be offered.

By default, Ziniki operates entirely in memory, with a clean initialization at the start of every run. However, it can be useful to load in an existing, prepared state. The `--dbfile` option offers the ability to load a ZIP file into memory before the run starts. Note that this is a read-only operation: Ziniki will never update this file. To obtain a ZIP file containing the current memory contents, it is necessary to use the `/zip` operation on the admin interface.

If you do not want to run Ziniki entirely in memory, it is possible to connect to an existing AWS environment and use the Dynamo and S3 resources provided there. The `--aws` option incorporates the relevant options to replace the in-memory database and content storage options with their AWS equivalents. This may require additional environment configuration to operate. It will also require manual setup and configuration of the Ziniki server.

A.4 Environment Parameters

The various packaging options do most of the "heavy lifting" to set up and configure a Ziniki instance. However, they depend on certain values which must be specified by the user. These are provided in the "environment file" - the file which describes the environment in which Ziniki is to run. Most importantly, these describe the domain on which Ziniki will respond, the ports it will use and the TLS configuration.

Like all Ziniki configuration files, the environment file is in JSON format.

This is the alleged server virtual host name (more complex Ziniki configurations can support multiple virtual hosts). This is a DNS name which should be routed to the local machine using a `hosts` file or other routing technology.

In order for Ziniki to bootstrap itself, it must have at least one user. In a live environment, this user is configured during the setup process. For simplicity during development, the setup process is automated and creates an initial `Domain` object corresponding to the value of `host` and associates that with the specified `firstUser`. This should be a valid OpenID or OAuth user URI. If you are using the builtin Ziniki authentication server, this will be an openID of the form:

```
https://ziniki.<host>:<idPort>/id/<name>
```

If you are using an external authentication source, you will need to consult its documentation to find out what your userid is.

The standard Ziniki configuration uses a total of six ports. The "main" port is the `zinikiPort`, which is where Ziniki will install its `HTTPS` server for handling requests for login, requesting applications and the like.

When Ziniki has its own authentication server enabled, the authentication server runs on this port, to offer the login window.

When Ziniki has its own authentication enabled, this is the port that will be used in `id` URIs.

The `adminPort` is used to access the administration functions of Ziniki. Exactly what will be available on this port depends on which modules have been installed, but with the standard configuration of the Ziniki in developer mode, this enables you to access the internals of the memory store at runtime.

Ziniki depends on access to a content store. In live environments, this is provided by a cloud storage service such as S3. In developer mode, an internal memory storage is used and served directly from the same Ziniki instance. To differentiate the services, the `contentPort` is used to identify requests to the content store.

Once logged in, most of the communication between Ziniki clients and servers takes place over a websocket. This connection takes place over a distinct hostname (`wsapi.`*`domain`* rather than `ziniki.`*`domain`*) and with the port number specified by `wsapiPort`.

In order to configure TLS for Ziniki, it is necessary to specify the appropriate certificate information. Currently, Ziniki offers two mechanisms: "standard" JKS keystores, or PKCS12 files. Both are configured the same way, as shown in the `domain.test.json` example:

```
"tlsconfig": {
"class": "org.ziniki.server.main.grizzly.PKCS12Config",
"root": "root",
"file": "keys/domain.test.p12",
"password": "password"
}
```

If you want to use a JKS keystore, replace the value of the `class` key with `org.ziniki.server.main.grizzly.JKSConfig`. Both require the same set of parameters. Note that no keystore files are shipped with Ziniki; you will need to create your own. The `ziniki-cert` script is provided to assist with this.

A.5 The `--custom` option

For full flexibility, it is possible to specify the `--custom` option, which stops any of the default options being used. It is then necessary to specify individual options to configure Ziniki. All of the pre-packaged options can still be used, but the following options will generally be needed to achieve a working configuration. Note that these options can also be used in combination with the pre-packaged options to obtain more complex configurations.

By default, the Ziniki instance uses the install directory (i.e. the parent directory of the `bin` directory in which the script is located) as its "root" directory. It uses this to find all of its standard resources. The `--root` options allows you to explicitly specify another directory.

You can specify arbitrary JSON configuration directly on the command line using the `--config` option. This is generally only practical for specifying individual configuration options. Note that even if you are specifying just a single parameter, it still needs to be in JSON object syntax, including having the field names and string arguments enclosed in double quotes.

Read a JSON configuration from a specified file.

Create a user during initial setup. This option may be specified multiple times to create multiple users.

Note that since users are associated with domains, all the domains associated with these users will also be automatically created. Each domain will only be created once, no matter how many users are associated with it.

In general, all FLAS code to be included in Ziniki is uploaded using the compiler at runtime. However, for convenience, existing FLIM directories may be loaded into development Ziniki instances directly from the command line using this option.

Multiple directories may be specified using this option. In this case, all of the packages in all of the directories will be loaded.

A.6 Notes

You need to add the domain you wish to use to your local host resolution mechanism (`hosts` file or otherwise).

If you are using an external OpenID source (such as Yahoo!) can you still use zinlogin? (I need to check this and document the result).

More environment options may exist for things like dynamo and s3 storage

There should be something to configure the content store.

B `zinlogin` command

During development, especially for automated testing, it is often inconvenient to deal with the necessities of logging in; however, Ziniki is designed to function almost exclusively with registered users. Thus the `zinlogin` command allows users to automate the login process *for a development server*. This script does *not* work with live servers.

The `zinlogin` script takes two parameters:

It is necessary to specify the URI of the Ziniki admin server. When Ziniki starts up it shows this information to the trace file at `INFO` level, but in general it will be `https://ziniki.<domain>:<adminPort>/`. For example, the admin port for the default `domain.test` configuration would be `https://ziniki.domain.test:18083/`.

This is the ID URI of the user to be logged in, for example, the `firstUser` entry in the environment file.

C `ziniki-cert` command

In order to use TLS effectively, it is necessary to generate certificates. `ziniki-cert` enables users to generate (free, unlimited) certificates *for local use* using a private CA whose root certificate can be added to common browsers (Chrome, Firefox, etc). These can also be shared with other users.

None of this is intended to be secure or for production use. It is suitable for use by individual developers or development teams on internal networks with test data.

The `ziniki-cert` command operates in the Ziniki installation directory, and creates files under the `keys` subdirectory; if that is not present, it creates it.

The first time you run `ziniki-cert` in an installation, it will create a root certificate, along with a new, private key. This certificate, called `rootCA.pem`, is the one you need to install and distribute in order to avoid having browser errors. Installing certificates is beyond the scope of this manual, and depends on browser, operating system and version, but is supported in all major browsers on all major operating systems.

The only required option is the domain name for which you wish to create a certificate. `ziniki-cert` automatically generates a wildcard certificate (since it wants to use `ziniki.<domain>`, `wsapi.<domain>` and possibly `id.<domain>` and `auth.<domain>), so you only need to specify the base domain name, such as` domain.test$.

By default, the certificate is issued with the IP address 127.0.0.1; it is possible to specify an alternative address, particularly if you are going to be using the certificate on a team server to be accessed remotely (or if you want to test mobile devices).

By default, the very secure password "password" is used to encrypt the keystore. Since it is assumed that all of this is just to enable TLS on an internal system, there is no concern about security or fraud here. However, if you wish to use another password for some reason, it is possible to change it using this option.