Command Line Access

For Unix and Linux users as well as those developers using command line tools, you can launch and administrate Wakanda Server, and evaluate a JS file.

Using Wakanda Shell

Wakanda Shell is a command line tool for Wakanda and is available for Macintosh and Linux. Here are some of the key features in Wakanda Shell:

- Connect to/disconnect from a Wakanda Server,
- Run JavaScript code server-side, and
- Send and execute a local JavaScript file on the server.

For more details, refer to the Wakanda Shell's man file using the following instruction:

```
man wakanda
```

Compatibility

Wakanda Shell requires a minimum of Wakanda version 9.

Note for Linux users

Here are the naming conventions in Wakanda Shell for Linux users:

Description	WAK 9	**WAK 10**
Wakanda Shell		wakanda
Wakanda Server (Community edition)	wakanda	wakanda-server
Wakanda Server (Enterprise edition)	wakandaenterprise	wakanda-enterprise-server

Using Wakanda Shell

In Wakanda Shell, you can use many shell commands, shell variables, and internal variables.

Shell commands and variables

Here are the Wakanda Shell commands and variables:

Command	**Description**
.connect	Connect to a Wakanda Server (alias for '.server connect')
.disconnect	Disconnect from the current session to Wakanda Server (alias for '.server disconnect')
.echo <text>	Display a text in the console (even if --silent has been provided). Like all shell commands, it supports environment variable expansion. See **Internal variables** for displaying additional information.
.exit	Quit the interactive mode
.help	Display this help
.history clear	Clear the history in Wakanda Shell (no past successful commands are accessible)
.include	Include a local script file
.info	Display information about the current session

.quit	Quit the interactive mode
.server connect	Connect to a Wakanda Server
.server disconnect	Disconnect from the current session to Wakanda Server
.set <variable> = <value>	Set the value of a console variable.

Here are a few examples of variables that you can set:

Variable to set	Description
.set verbose = [true|false]	Display additional information for all Wakanda Shell commands
.set colors = [true|false]	Allow output to be displayed with color
.set ps1 = <prompt>	Format the prompt based on the **Internal variables**. To set it back to its default, you pass .set ps1=""

Internal variables

You can insert these internal variables when using any of the Wakanda Shell commands or when customizing the prompt:

Variable	Description
${project}	Name of the project for the current session or empty if not connected.
${url}	Actual URL (e.g., https://jsmith@www.example.com:8083) or empty if not connected. The password will be omitted from this variable.
${host}	Host name used to connect to the current session (ex: 127.0.0.1 or example.com) or empty if not connected.
${user}	User name used to connect to the current session or empty if not connected.
${port}	Port used to connect to the current session or empty if not connected.
${time}	Current time (HH::MM).
${black|red|green|yellow|blue|purple|cyan|gray}	Set the text color to the corresponding value.
${bold}	Make the text bold or increased intensity.
${faint}	Make the text faint or decreased intensity. *Note: this setting might not be supported by all terminals.*
${reverse}	Activate the reverse video mode based on the current color.
${reset}	Reset the text color and text style back to default after formatting it using any of the color internal variables or ${bold}.

If you want the variable to be interpreted each time, you must escape the $, e.g., /${time}. In this way, the time is updated each time the prompt is displayed.

You can also include other escape characters like \r (carriage return) or \n (line feed).

***Note**: The ${bold} and ${faint} variables only take effect after declaring one of the color internal variables. See the **Modifying the prompt** example below.*

Using a shell

In your shell, you can use some of the standard environment variables as well as a variety of options that refer to your use of Wakanda Shell.

Standard environment variables

Here are some of the standard environment variables:

Variable	Description
WAKANDA_URL	Default server URL, including username, URL-encoded password, and port (e.g., https://jsmith:pw1234@www.example.com:8083)
WAKANDA_PS1	The Wakanda Shell prompt when connected to a Wakanda Server
WAKANDA_CONNECT_AT_STARTUP	Try to connect to the remote Wakanda Server at startup if set to "1" or "true" (see "--connect")
HTTP_PROXY | http_proxy	Hostname or IP address of the proxy server
HTTPS_PROXY | https_proxy	Hostname or IP address of the proxy server for SSL remote Wakanda Servers

Options

Here are the options:

Option	Description
-f, --file=VALUE	Input script file (which may contain shell commands)
-i, --interactive	Force interactive mode
-x, --execute=VALUE	Input script (which may contain shell commands). If other script files are provided, this script will be executed after all others if no errors have occurred.
Session management	
-c, --connect	Connect to Wakanda Server at startup (using the default host information)
-u, --url=VALUE	Default server URL, including username, URL-encoded password, and port (e.g, https://jsmith:pw1234@www.example.com:8083)
Proxy settings	
--no-proxy	Don't use proxies, even if the appropriate htpp_proxy or HTTP_PROXY environment variables are defined
Output	
--no-color	Disable color output
Help & Messages	
--verbose	Display verbose messages for debugging
--silent	Do not display information messages (ignored if --verbose is used)
--version	Display the version of wakanda shell and exit

Modes

You can use Wakanda Shell in either interactive or non-interactive mode.

Interactive Mode

Here are the keyboard equivalents when using the interactive mode.

Keyboard Shortcuts

Here are Wakanda Shell's keyboard shortcuts:

Keyboard shortcut	Description
<ctrl+a>	Go to the beginning of the line on which you are currently typing
<ctrl+c>	Abort the current user input and start a new input
<ctrl+d>	Exit Wakanda Shell if the user input is empty
<ctrl+e>	Go to the end of the line on which you are currently typing
<ctrl+k>	Clear the line after the cursor
<ctrl+l>	Clear the screen
<ctrl+r>	Allows you to search through previously used commands
<ctrl+u>	Empty the current user input

Non-interactive Mode

Wakanda Shell allows shebang in input files (even included ones). If a script file has the program "wakanda" as an interpreter directive (and if it has execute permissions), a script file may become executable.

Interacting with the project

You can execute some of Wakanda's functions to retrieve information or even data from the project.

Wakanda keywords/functions	Description
ds	Returns the structure of the model
ds.{*DataclassName*}.{*AttributeName*}	Information regarding the attribute

Here is an example of the information displayed for the ds.Employee.firstName attribute:

```
{
    "kind": "storage",
    "relatedDataClass": null,
    "name": "firstName",
    "fullTextIndexed": false,
    "uuid": null,
    "scope": "public",
    "indexType": "",
    "indexed": false,
    "type": "string"
}
```

Examples

Here are some examples on how to use Wakanda Shell.

Connecting to Wakanda Server

If you want to specify a URL and user with his/her password, you write:

```
.connect http://jsmith:pw1234@www.example.com:8081
```

```
.connect https://jsmith:pw1234@www.example.com:443
```

***Note**: In the current version of Wakanda, you can only login with a user in the Admin group. The username and password must be URL encoded.*

To disconnect from the remote Wakanda Server and close its opened session:

```
WakandaProject > .disconnect
```

Running JavaScript code server-side

You can directly write JavaScript code and execute it on Wakanda Server:

```
WakandaProject > var i = { example: "Hello world" };
```

To output the *i* variable, you write:

```
WakandaProject > i;
```

The result appears as shown below:

```
{
    "example": "Hello world"
}
```

Executing Wakanda functions

In the following example, you can execute the following code once connected:

```
WakandaProject > ds.Company.query("name == :1", 'w*')
```

In our case, only one entity was returned:

```
[
    {
        "__KEY": "DCBC758008A8D8469FC5D8377C59A15E",
        "__STAMP": 1,
        "ID": "DCBC758008A8D8469FC5D8377C59A15E",
        "name": "WT Services Corp",
        "address": "6544 Canton Rd.",
        "city": "Charlotteville",
        "stateProv": "NC",
```

```
            "country": "USA",
            "telephone": "2340987645",
            "url": "wtsvcscorp.com",
            "industry": "Medical",
            "logo": {
                "width": 542,
                "meta": {

                },
                "size": 8925,
                "height": 344,
                "length": 8925
            },
            "sales": 389000,
            "employees": {
                "__COUNT": 4
            },
            "numberOfEmployees": 4,
            "cityState": "Charlotteville, NC"
        }
    ]
```

Because the entities found are returned in an array, you can access the first entity found in the search by writing:

```
WakandaProject > ds.Company.query("name == :1", 'a*')[0]
```

You can also just retrieve a value in the attribute by writing:

```
WakandaProject > ds.Company.query("name == :1", 'a*')[0].name
```

In the above case, the following value is returned:

```
"Apple"
```

Working with script files

To send and execute one or more script files:

```
wakanda ./myfile1.js ./myfile2.js
```

To insert the results of a query in a JSON file:

```
wakanda -u "http://127.0.0.1:8081" -c -x 'ds.Company.query("name == :1",
"w*")' > company.json
```

Retrieving information

In the following example, you can retrieve information about the project to which you are connected:

```
WakandaProject > .info
```

This command returns the following output:

```
PROJECT
url               http://127.0.0.1:8081
project           WakandaProject

SERVER
server            Wakanda Server 9 build 9.162538 [Linux]
hostname          127.0.0.1
port              8081
ssl               false
username          adminUser
wasid             AABF510D369227488506BFF71EF9D154

STASTICS
sessions          7
cache             200 MiB (free: 198 MiB, 99%)
debugger          false
entity sets       0
```

Modifying the prompt

You can modify the prompt after connecting to your project by using the **.set** command along with the shell's internal variables:

```
WakandaProject > .set ps1 = "${red}[\${project}]${green}${bold} \${time}
${reset}> "
```

After executing this line, your prompt is replaced once and appears as shown below:

```
[WakandaProject] 12:12 >
```

To set the prompt back to its default, you pass:

```
WakandaProject > .set ps1 = ""
```

Administrating Wakanda Server (Unix)

You can use a command line to launch and administrate Wakanda Server on a Unix-based system, i.e. OS X or Linux. Thanks to this feature, you can open a solution automatically, for example, at startup. You can also launch several Wakanda Servers with different default administration projects.

This section covers command line arguments for Unix-based systems only. For information on command line arguments on Windows, please refer to section **Administrating Wakanda Server (Windows)**.

Syntax

The basic syntax to use on Unix systems is:

```
<Wakanda_server_path> [--option=value] [-option]...
OR
<Wakanda_enterprise_server_path> [--option=value] [-option]...
```

... where <*Wakanda_server_path*> or <*Wakanda_enterprise_server_path*> is the full pathname of the server application.
Example on OS X: "**Volumes/Mac\ HD/Applications/Wakanda/Wakanda\ Server.app/Contents/MacOS/Wakanda\ Server**"
Example on Linux: **wakanda-enterprise-server**

Options are described below.

Launching Several Instances

You can launch several instances of a Wakanda Server from the same bundle. For this, you just need to call several command lines and specify the HTTP and SSL ports for the ServerAdmin project into each command line, using ***admin-port*** and ***admin-ssl-port*** arguments. You have to ensure that there are no port conflicts between the different projects you are opening.

The purpose of these parameters is to resolve potential HTTP and SSL port conflicts between the different ServerAdmin projects. Remember that there is one default ServerAdmin project per solution. Default values are 8080 for the HTTP port and 4433 for the SSL port.

Options

Option	Description	Version information
Opening a file		
-s, --solution: <solution_path>	Full pathname of Wakanda solution to open. If you do not pass this parameter, the default solution is opened.	
-s, --solution:<js_file>	Full pathname javascript file to execute. For more information on javascript file argument, please refer to the **Evaluating a JS script** section.	
Administration		
--admin-port: <number>	Force the HTTP port number of the built-in ServerAdmin project. The ServerAdmin project is the default administration project, available through the **Wakanda Server Administration** page. It is published on port 8080 by default (available for local connections only). By setting a	

	different value, you can publish this default project on another port, allowing you to launch several Wakanda Servers running the default administration project. The HTTP port that you set is used during the entire server session, even if another solution is opened. If the opened solution already contains an administration project (project with key administrator="true" in the *myproject.waSettings* file), the *admin-port* parameter is ignored.	
--admin-ssl-port: <number>	Force the HTTPS port number of the built-in ServerAdmin project (use of SSL/TLS protocol is mandatory for remote connections to the administration project). It is published on port 4433 by default. By setting a different value, you can publish this default project on another port. The HTTPS port that you set is used during the entire server session, even if another solution is opened.	
--admin-password: <password>	Set a password to the default administrator user, automatically added to the default solution. By default, a user with the login "admin" and an empty password is created in this solution, and added to the "Admin" group. Once a password is set, the default solution and thus the ServerAdmin project, is protected. For security reasons, assigning a password to the default administrator user is highly recommended.	
Debugger		
-g, --debugger: <value>	Allow you to define the debugger to launch at startup. The available values are "--debugger:remote" to activate the **Remote Web Debugger**, "--debugger:wakanda" to activate the **Wakanda Debugger**", or "--debugger:none" (default) no debugger is activated at startup. Ignored if --debug-off is specified.	
--debug-off	Disable the Debugger features.	
Service discovery		
--no-discovery	Turns off the service discovery protocol in Wakanda Server. This protocol is enabled by default and allows Wakanda Studio to list Wakanda Server solutions broadcasted over the local network. This service is used for remote debugging (see **Starting Remote Debugging**) or git push and pull features (see **Access to the remote Solution Server**). You may want to disable this service if you don't need it.	Added in v9
Information		
--syslog	Forward Wakanda Server's log messages to the Syslog daemon (see below)	
--verbose	Verbose mode	Added in v9
--version	Display the version and exit	
-h, --help	Display the help and exit	
Specific options		
--job-id:<id>	Specify the server job id	
--system-workers: <systemWorker_path>	Load the *systemWorker.json* configuration file at the specified path (see **Configuring systemWorkers.json file**)	Added in v10

Forwarding Log Messages to Syslog

You can forward Wakanda Server's log messages to the Syslog daemon using the **--syslog** option (see above). You also have to configure your system:

- **Ubuntu configuration**: You can use the *syslog-ng* package. This feature works without any changes to the syslog-ng configuration file.
  Log messages are written in the **/var/log/user.log** file. You can also use the log viewer application.

- **OS X configuration**: Make sure that you have this line in the **/etc/syslog.conf** file:
  user. */var/log/user.log*
  Restart the syslog daemon (rebooting the Mac is the simplest way).
  Log messages are written in the **/var/log/user.log** file. You can also use the "Console" application.

Start / Stop / Status Service on Linux

If you have installed Wakanda Server for Linux through the All-In-One installer, you benefit from a start / stop / status service that you can use to manage Wakanda Server.

Command lines for this service are the following:

```
- sudo service wakanda start
- sudo service wakanda stop
- sudo service wakanda status
```

Examples

- (OS X) Launching Wakanda Server and opening the "invoices" solution. If this solution does not contain an administration project (administrator="false" in settings), the ServerAdmin project is used and published on the default HTTP port (8080)

```
/Volumes/Mac\ HD/Applications/Wakanda/Wakanda\
Server.app/Contents/MacOS/Wakanda\ Server /Volumes/Mac\
HD/Solutions/invoices.waSolution
```

- (Linux) Launching Wakanda Server and opening the default solution (containing the ServerAdmin project, published on port 8080)

```
./wakanda
```

- (Linux) Launching Wakanda Server and opening the "invoices" solution. If this solution does not contain an administration project (administrator="false" in settings), the ServerAdmin project is used and published on the HTTP port 8080

```
./wakanda /home/AdminUserName/invoices.waSolution
```

Administrating Wakanda Server (Windows)

You can use a command line to launch and administrate Wakanda Server on Windows. Thanks to this feature, you can open a solution automatically, for example, at startup. You can also launch several Wakanda Servers with different default administration projects.

This section covers command line arguments for Windows only. For information on command line arguments on Unix-based systems (OS X or Linux), please refer to section **Administrating Wakanda Server (Unix)**.

Syntax

The basic syntax to use on Windows is:

```
<Wakanda_server_path> [/option:value] [/option]...
OR
<Wakanda_enterprise_server_path> [/option:value] [/option]...
```

... where <*Wakanda_server_path*> or <*Wakanda_enterprise_server_path*> is the full pathname of the server application.

Examples:
"**C:\ProgramData\Wakanda\Server\Wakanda Server.exe**"
"**C:\ProgramData\Wakanda Enterprise\Server\Wakanda Server.exe**"

The options are described below.

Launching Several Instances

You can launch several instances of a Wakanda Server from the same bundle. For this, you just need to call several command lines and specify the HTTP and SSL ports for the ServerAdmin project into each command line, using ***admin-port*** and ***admin-ssl-port*** arguments. You have to ensure that there are no port conflicts between the different projects you are opening.

The purpose of these parameters is to resolve potential HTTP and SSL port conflicts between the different ServerAdmin projects. Remember that there is one default ServerAdmin project per solution. Default values are 8080 for the HTTP port and 4433 for the SSL port.

Options

Option	Description	Version information
Opening a file		
/s, /solution: <solution_path>	Full pathname of Wakanda solution to open. If you do not pass this parameter, the default solution is opened.	
/s, /solution:<js_file>	Full pathname of the JavaScript file to execute. For more information on the JavaScript file argument, please refer to the **Evaluating a JS script** section.	
Administration		
/admin-port: <number>	Force the HTTP port number of the built-in ServerAdmin project. The ServerAdmin project is the default administration project, available through the **Wakanda Server Administration** page. It is published on port 8080 by	

	default (available for local connections only). By setting a different value, you can publish this default project on another port, allowing you to launch several Wakanda Servers running the default administration project. The HTTP port that you set is used during the entire server session, even if another solution is opened. If the opened solution already contains an administration project (project with key administrator="true" in the *myproject.waSettings* file), the *admin-port* parameter is ignored.	
/admin-ssl-port: <number>	Force the HTTPS port number of the built-in ServerAdmin project (use of SSL/TLS protocol is mandatory for remote connections to the administration project). It is published on port 4433 by default. By setting a different value, you can publish this default project on another port. The HTTPS port that you set is used during the entire server session, even if another solution is opened.	
/admin-password: <password>	Set a password to the default administrator user, automatically added to the default solution. By default, a user with the login "admin" and an empty password is created in this solution, and added to the "Admin" group. Once a password is set, the default solution and thus the ServerAdmin project, is protected. For security reasons, assigning a password to the default administrator user is highly recommended.	
Debugger		
/g, /debugger: <value>	Define the debugger to launch at startup. The available values are "/debugger:remote" to activate the **Remote Web Debugger**, "/debugger:wakanda" to activate the **Wakanda Debugger**", or "/debugger:none" (default) to have no debugger activated at startup. Ignored if /debug-off is specified.	
/debug-off	Disable the Debugger features.	
Service discovery		
/no-discovery	Turns off the service discovery protocol in Wakanda Server. This protocol is enabled by default and allows Wakanda Studio to list Wakanda Server solutions broadcasted over the local network. This service is used for remote debugging (see **Starting Remote Debugging**) or git push and pull features (see **Access to the remote Solution Server**). You may want to disable this service if you don't need it.	Added in v9
Information		
/verbose	Verbose mode	Added in v9
/version	Display the version and exit	
/?, /help	Display the help and exit	
Specific options		
/job-id:<id>	Specify the server job id	
/system-workers: <systemWorker_path>	Load the *systemWorker.json* configuration file at the specified path (see **Configuring systemWorkers.json file**)	Added in v10

Examples

- Launching Wakanda Server and opening the default solution (containing the ServerAdmin project, published on port 8080 and SSL port 4433)

```
"C:\ProgramData\Wakanda\Server\Wakanda Server.exe"
```

- Launching another Wakanda Server instance and opening the default solution (containing the ServerAdmin project, published on HTTP port 80 and SSL port 443):

```
"C:\ProgramData\Wakanda\Server\Wakanda Server.exe" /admin-port=80
/admin-ssl-port=443
```

- Launching Wakanda Server, opening the default solution (containing the ServerAdmin project) and publishing the ServerAdmin project on the HTTP port 8090

```
"C:\ProgramData\Wakanda\Server\Wakanda Server.exe" /admin-port=8090
```

- Launching Wakanda Server and opening the "invoices" solution. If this solution does not contain an administration project (administrator="false" in settings), the ServerAdmin project is published on the HTTP port 8090

```
"C:\ProgramData\Wakanda\Server\Wakanda Server.exe"
C:\solutions\invoices.waSolution /admin-port=8090
```

- Launching several Wakanda Server instances and solutions:

```
"C:\ProgramData\Wakanda\Server\Wakanda Server.exe"
"C:\ProgramData\Wakanda\Server\Wakanda Server.exe"
C:\solutions\invoices1.waSolution /admin-port=81 /admin-ssl-port=444
"C:\ProgramData\Wakanda\Server\Wakanda Server.exe"
C:\solutions\invoices2.waSolution /admin-port=80 /admin-ssl-port=443
```

***Note**: Quotes needs to be used when a path contains space characters.*

Evaluating a JS script

You can execute any JavaScript file with Wakanda Server using a command line. This feature is available on all platforms (Windows, Mac OS and Linux).

Basically, the running sequence is:

1. **You execute a command line that contains the Wakanda Server path and a JavaScript file path.**
   Wakanda Server should not be already running.
2. **An instance of Wakanda Server is launched and evaluates the script.**
   Note that the context is outside of any solution or project (application). All APIs that are not solution-dependent or project-dependent can be used, for example the **console** object or the [#cmd id="100030"/] method (see below).
3. **The server quits.**

The syntax to use is:

```
<Wakanda_server_name> <JS_File_path>
```

where:

- ***<Wakanda_server_name>*** is the full pathname of the server application ("Wakanda Server.exe" on Windows and "Wakanda Server.app" on Mac OS)
- ***<JS_File_path>*** is the full pathname of the JavaScript file to execute. Only one file can be passed. The path should be expressed in the system syntax. If you do not pass this parameter or if the JavaScript file is not found, the Wakanda Server is launched and opens the default solution (see **Administrating Wakanda Server (Unix)**).

Warning: The shells do not accept spaces or / symbols in command lines. To avoid interpretation errors, insert parameters between double quotes "" (see examples).

During execution:

- The file is evaluated outside of any solution or project (application) context. All APIs that are not solution-dependent or project-dependent can be used.
- The **Console** object sends messages in the terminal application from where the Wakanda Server was launched.
- Parsing or execution errors are sent to the terminal as well.

After the execution:

- A **null** value is sent.
- The Wakanda Server instance quits.

Example

On Windows, we want to print the classic "Hello World" message in the console.

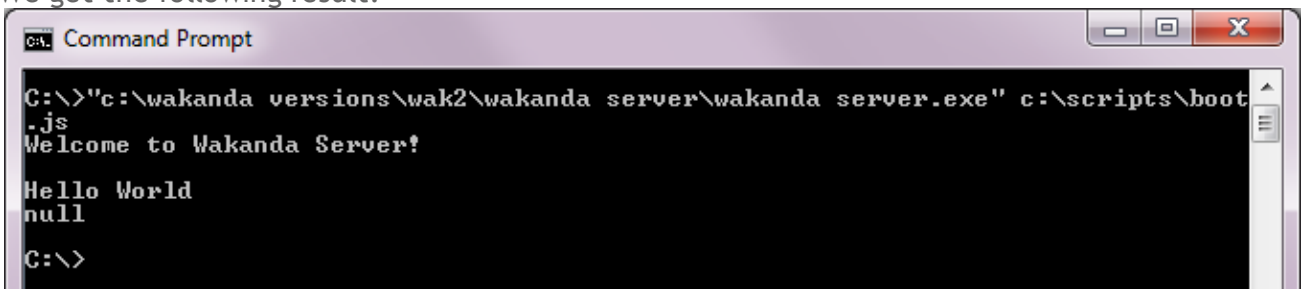
- We create a simple "boot.js" file, at the following location: "C:\scripts\boot.js". The file only contains the following code:

```
var myText = "Hello World";
console.info(myText);
```

- We open the *Command prompt* window and execute the following command line:

```
"c:\wakanda versions\wak2\wakanda server\wakanda server.exe"
"c:\scripts\boot.js"
```

We get the following result:



Available Wakanda APIs

Here are the main server-side methods and objects available when Wakanda Server evaluates a .js file through a command line:

- console
- os
- process
- BinaryStream( )
- Buffer( )
- clearInterval( )
- clearTimeout( )
- close( )
- createDataStore( )
- dateToIso( )
- displayNotification( )
- exitWait( )
- File( )
- Folder( )
- garbageCollect( )
- generateUUID( )
- getURLPath( )
- getURLQuery( )
- getWalibFolder( )
- include( )
- isoToDate( )
- JSONToXml( )
- loadImage( )
- loadText( )
- open4DBase( )
- openDataStore( )
- saveText( )
- setInterval( )
- setTimeout( )

SharedWorker( )
SystemWorker( )
TextStream( )
wait( )
Worker( )
XMLHttpRequest( )
XmlToJSON( )