

Wakanda Server Administration

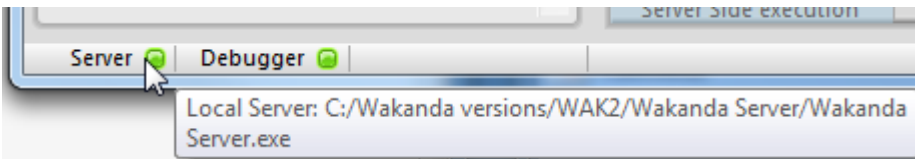
Managing Wakanda Server

Configuring the Connection between the Studio and the Server

You can manage Wakanda Server from Wakanda Studio. The actions that you are allowed to do depend on the way the two applications are connected:

- **Local connection** (both Wakanda Server and Wakanda Studio are running on the same computer and share the same solution on the file system). In this case, you can:
 - Start and stop the Wakanda Server
 - Load a solution and edit all parts of the solution and its project(s).
 - Open and close a debug connection
- **Remote connection** (Wakanda Server and Wakanda Studio are not running on the same computer). Server and Studio must use the same solution on each machine. You need to make sure that both solutions are identical (for example, using standard synchronization tools).
In this case, the only action available from the Studio is:
 - Open and close a debug connection

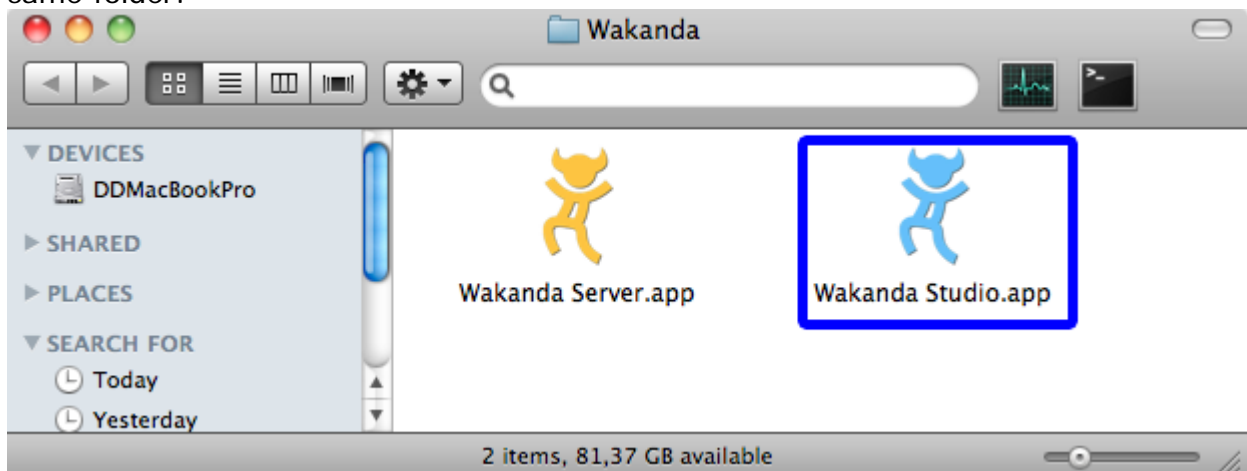
In Wakanda Studio, you can get information about the running server by placing the cursor above the green icon at the bottom left of your window. For a local connection, the path to the server application is displayed in the tip:



Configuring a Local Connection

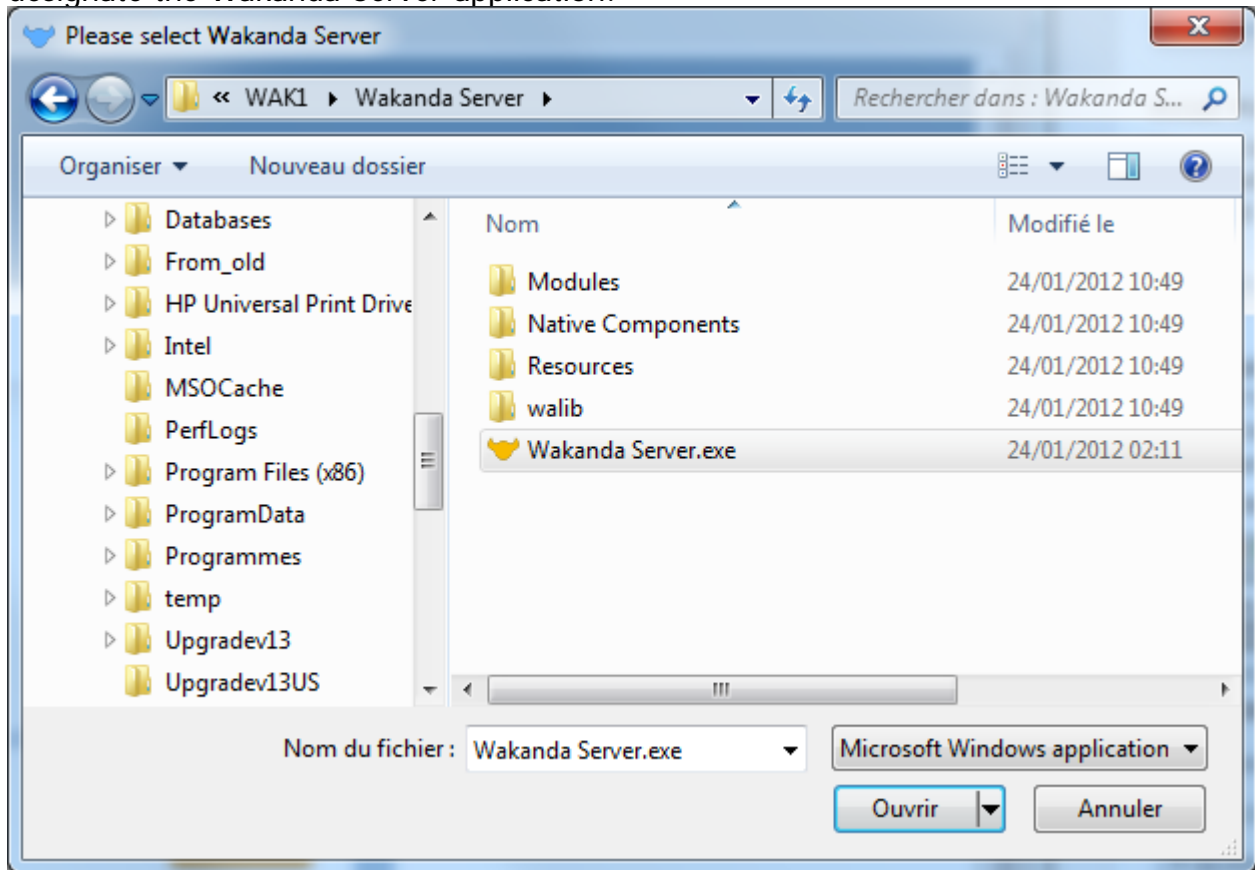
To be able to open a local connection to Wakanda Server from Wakanda Studio (both applications are running on the same machine), you have two possibilities:

- Install Wakanda Server and Wakanda Studio subfolders (*packages* on Mac OS) in the same folder and name them "Wakanda Server" and "Wakanda Studio". Here is the typical installation on Mac OS with Wakanda Server and Wakanda Studio in the same folder:



In this case, both applications will be connected automatically (unless another "preferential server" has been set for a solution, see [Defining a Preferential Server](#)).

- Install Wakanda Server and Wakanda Studio subfolders wherever you want on the same machine and launch Wakanda Studio. In this case, the first time that the Wakanda Studio will need to launch the server, a standard open dialog box will be displayed, allowing you to designate the Wakanda Server application:



Debugging on a Remote Server

You can connect to a remote Wakanda Server from Wakanda Studio (when the applications are running on different machines). However, this functionality only allows you to open a debug connection with the server. You can trace code execution at runtime and add/remove break points.

Configuration

To be able to use the remote debugging feature, you need to have the following elements:

- Two computers: A (running Studio for debug) and B (running server).
The following configurations are supported:

Server on Mac OS Server on Windows Server on Linux

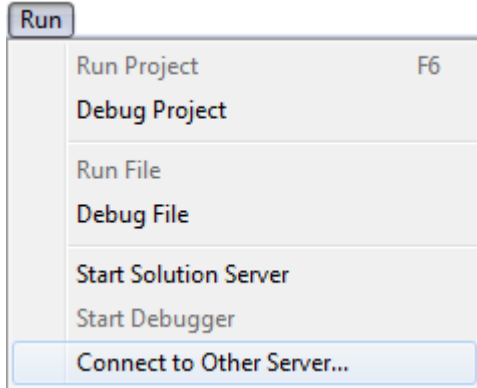
Studio on Mac OS	x	x	x
Studio on Windows	x	x	x
Studio on Linux	n/a	n/a	n/a

- A copy of the same solution must be installed on each computer. The solution should contain at least one server-side JavaScript file.
- On the server machine, all the following TCP ports should not be blocked by any network devices (firewall, proxy...) for the Studio:
 - admin port (usually 8080)
 - SSL admin port (usually 4433)
 - application port(s) (usually 8081, 8082...)
 - remote debugger port(s) (usually 1919)

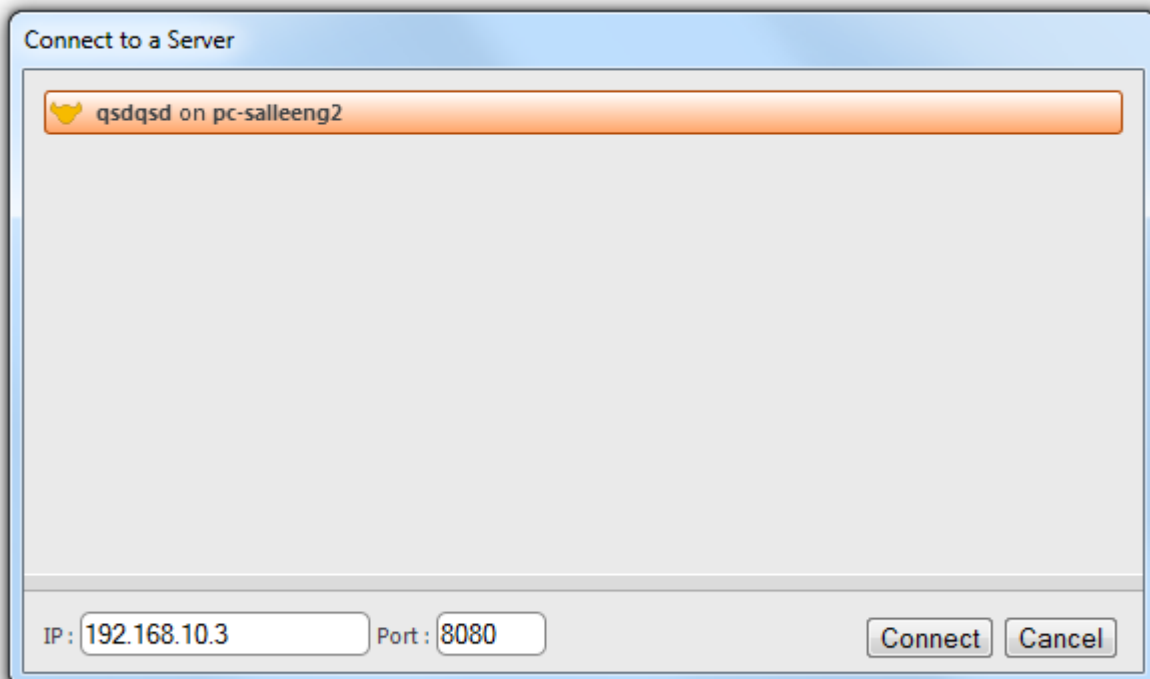
Starting Remote Debugging

To run a remote debugging session between computer A (debugger) and computer B (server):

1. Launch the Wakanda Server on computer B and open the solution.
If the server is launched through a Studio, make sure the local debugger is NOT launched (you may have to stop it -- you can even close the Studio).
2. Open the same solution with Wakanda Studio on computer A.
Do not accept to start the server if you are prompted to (it depends on local settings).
3. On computer A, select the **Connect to Other Server...** item in the **Run** menu:



The following dialog box is displayed:

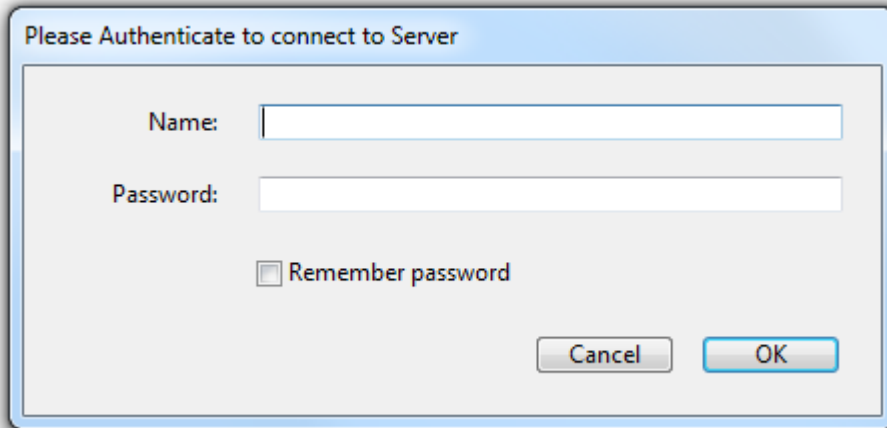


This dialog box lists all the Wakanda Server solutions broadcasted over the local network (this feature uses the *Bonjour* protocol).

4. Fill the IP address and Port of the server (computer B).
You have to enter the SSL Admin port (usually 4433).
5. Click **Connect**.
Wakanda Studio opens the debug socket on the server and, if the Studio and the Server are running the same solution, a debug connection is opened. You will get the message: "Studio is connected to xxx.xxx.xxx.xxx:xxxx"
Note: The debugging port will be used automatically. Keep in mind that, unlike other ports, the debugging port is not defined in a setting file. It is dynamically assigned by the server: first, it tries to open port 1919; if this fails, it then tries 1920, 1921, and so on until an available port is found.
6. On computer A, you can add a break point in the JavaScript file then execute **Run File**.

To close the connection, you just need to click on the **Stop debug** button of the toolbar.

Note: If Wakanda's [Controlled Admin Access Mode](#) has been activated for your solution, you will be prompted to enter a username and a password to connect to the server.

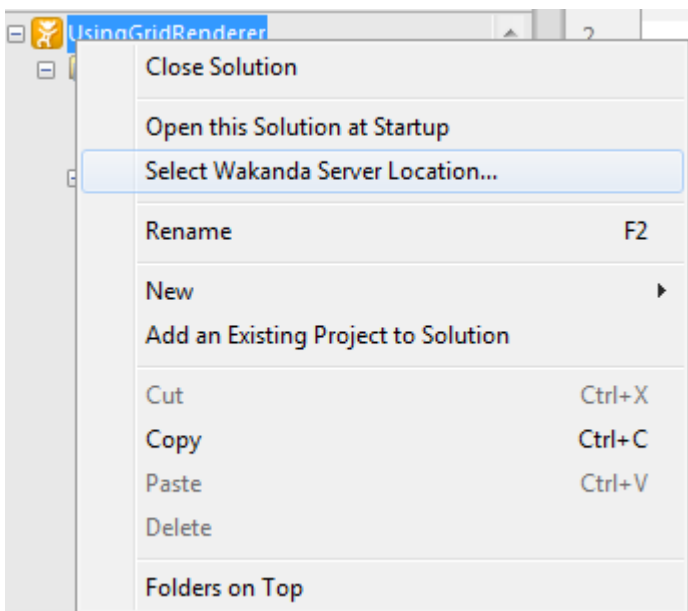


Only users belonging to the "Admin" group can connect remotely to a secured solution. For more information, refer to the [Configuring Admin Access Control](#) section.

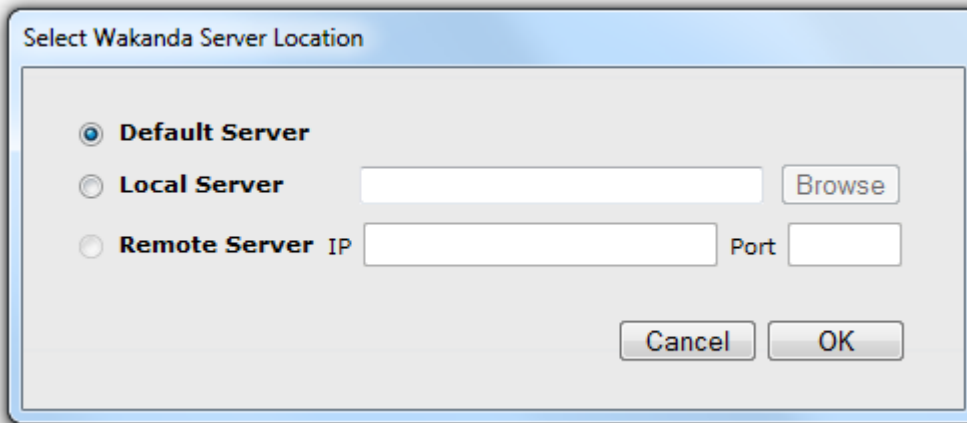
Defining a Preferential Server

You can define a "preferential server" (local or remote) for a given solution. Once set, the preferential server is stored in the solution settings and will be used by default with the solution.

To define a preferential server for a solution, right-click on the solution name in the Explorer area and choose **Select Wakanda Server Location...** in the contextual menu:



The following dialog box is displayed:

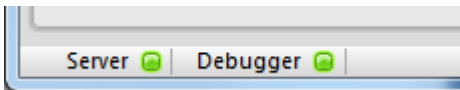


Three options are available to define the preferential server:

- **Default Server**: use a local Wakanda Server located beside the Wakanda Studio on the same machine (see [Configuring a Local Connection](#))
- **Local Server**: use a local Wakanda Server but not the Default Server. You need to define the location of the Wakanda Server application file on your disk by clicking the **Browse** button.
- **Remote Server**: use the remote Wakanda Server to which Wakanda Studio is currently connected through a debug connection (see [Connecting to a Remote Server](#)). This option is not available if no debug connection is currently open.

Starting and Stopping Wakanda Server

In Wakanda Studio, a green icon at the bottom left side of your window indicates if the server (as well as the debugger) is running:



Wakanda server is running

Specific features allow you to:

- Start and stop the Wakanda Server automatically when opening or closing a solution
- Start and stop the Wakanda Server manually

Note: For these features to work, the Wakanda Studio must know the current location of the Server. For more information, please refer to the [Configuring the Connection between the Studio and the Server](#) section.

You can also launch Wakanda Server using a command line, without having to use Wakanda Studio (see the [Launching Wakanda Server using a Command Line](#) section).

Note: If the Wakanda admin access control is activated for the solution, you will be prompted to enter a login and a password to start or stop the server. Only users belonging to the "Admin" group can administrate the server. For more information, please refer to the [Configuring Admin and Debug Access Control](#) section.

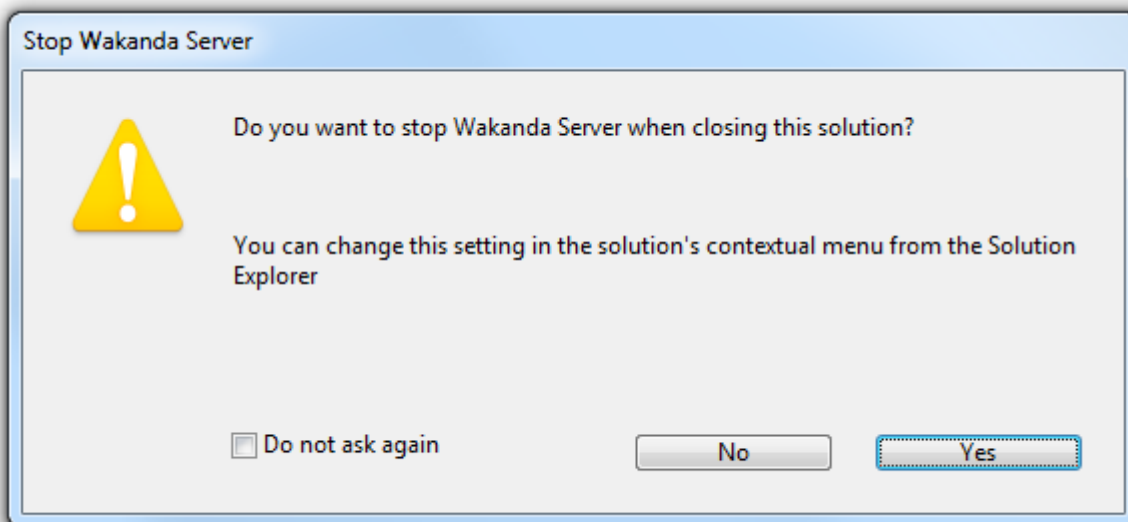
Automatic Handling

When you first open a solution with Wakanda Studio, you are asked to start automatically the server when opening the solution in the Studio:



- If you click **Yes**, the server will be automatically launched when you open the solution
- If you click **No**, the server will not be automatically launched when you open the solution. You will need to launch it manually.

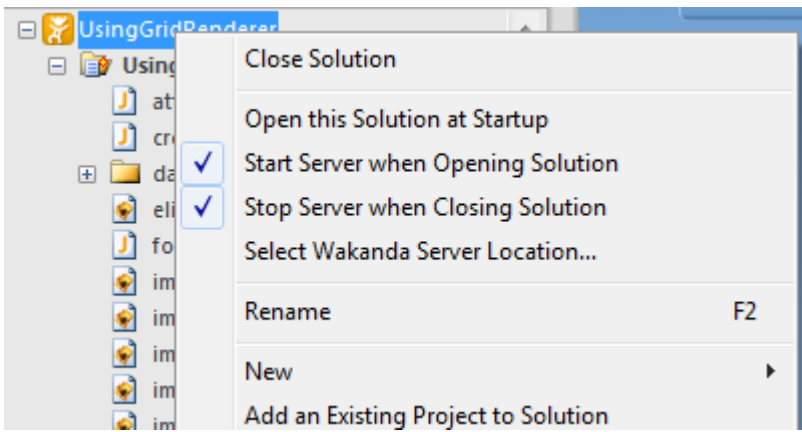
The same setting can be set when you close the solution. The following dialog box is displayed:



- If you click **Yes**, the server will be automatically stopped when you close the solution
- If you click **No**, the server will not be automatically stopped when you close the solution. You will need to stop it manually.

If you check "Do not ask again", the corresponding dialog will not be displayed anymore for any solution.

You can also manage the automatic opening and closing of the Wakanda Server for each solution using the contextual menu in the explorer:



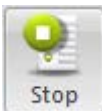
Manual Handling

You can start and stop the Wakanda Server at any time when the Wakanda Studio is launched and when a valid solution is open.

To start the server, you select **Start Solution Server** in the **Run** menu or click on the Start button:



Once the server is running, you can select **Stop Solution Server** in the **Run** menu or click on the Stop button:



Launching Wakanda Server using a Command Line

You can use a command line to launch the Wakanda Server on all platforms (Windows, Mac OS and Linux). As parameters, you can pass the solution to open as well as the ServerAdmin project HTTP port.

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.

The syntax is:

```
Wakanda_server_name [solution_path] [--admin-port=http_port_number] [--admin-ssl-port=http_port_number] [--debug-off]
```

where:

- *Wakanda_server_name* is the full pathname of the server application ("Wakanda Server.exe" on Windows and "Wakanda Server.app" on Mac OS)
- *solution_path* is the full pathname of the solution to open. The path should be expressed in the system syntax. If you do not pass this parameter, the default solution is opened.
- **--admin-port=http_port_number** is an optional parameter that sets the HTTP port number of the built-in ServerAdmin project. The ServerAdmin project is the default administration project. It is published on port 8080 by default. 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**=*http_port_number* is an optional parameter that sets the HTTPS port number of the built-in ServerAdmin 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.
- **--debug-off** is an optional parameter that disables [Debugger](#) features in Wakanda Server. When this parameter is passed, the debugging interface is not launched on the server side. This parameter is useful when the solution is used in a production environment.

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

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 your different opened projects.

The purpose of these parameters is to resolve potential HTTP and SSL ports conflicts between the different ServerAdmin projects (there is one default ServerAdmin project per solution).

Default values are 8080 for the HTTP port and 4433 for the SSL port (see Examples 2)

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

- (Windows) Launching the Wakanda server and opening the default solution (containing the ServerAdmin project, published on port 8080)
`"C:\Wakanda\Wakanda Server.exe"`
- (Windows) Launching the Wakanda server, opening the default solution (containing the ServerAdmin project) and publishing the ServerAdmin project on the HTTP port 8090
`"C:\Wakanda\Wakanda Server.exe" "--admin-port=8090"`
- (Windows) Launching the 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:\Wakanda\Wakanda Server.exe" "C:\solutions\invoices.waSolution" "--admin-port=8090"`
- (Mac OS) Launching the 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 the Wakanda server and opening the default solution (containing the ServerAdmin project, published on port 8080)
`./wakanda`
- (Linux) Launching the 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`

Examples 2

- (Windows) Launching a Wakanda server and opening the default solution (containing the ServerAdmin project, published on HTTP port 8080 and SSL port 4433):
`"C:\Wakanda\Wakanda Server.exe"`
- (Windows) Launching another Wakanda server and opening the default solution (containing the ServerAdmin project, published on HTTP port 80 and SSL port 443):
`"C:\Wakanda\Wakanda Server.exe" --admin-port=80 --admin-ssl-port=443`
- (Windows) Launching another Wakanda server and opening a solution. The ServerAdmin project is published on HTTP port 81 and SSL port 444:
`"C:\Wakanda\Wakanda Server.exe" --admin-port=81 --admin-ssl-port=444`

Configuring secure connections (SSL/TLS)

The Wakanda HTTP server provides support of secure connections through HTTPS. To implement secure connections in Wakanda, you need to:

- get a SSL/TLS certificate
- install the appropriate files in your Wakanda project folder
- configure the relevant settings for your project

How to get a certificate?

A Wakanda server working in secure mode means that you need a digital SSL/TLS certificate. This certificate references various information such as the site ID as well as the public key used to communicate with the site. This certificate is transmitted to the Web clients (browsers) connecting to this site. Once the certificate has been identified and validated, the communication is made in secure mode.

SSL/TLS certificates can be delivered by a Certification Authority or be self-signed. A well known Certification Authority (such as Verisign® or Thawte®) will provide certificates that will be authorized by many browsers (i.e. automatically validated), but the price will be expensive. Note however that the certificate only manages the 'authentication' part of the secured connection. When a certificate is not referenced in the browser's properties (this is the case for self-generated certificates), the user is just asked to validate it manually. Once the certificate is accepted, the connection is established in secure mode and is encrypted. During the development phase, it is a good idea to use self-signed certificates.

Applying for a certificate

To get a SSL certificate from a certification authority, you need to:

- generate a private key on your Wakanda server
- generate a CSR (Certificate Signing Request)
- send the CSR to a certification authority or use it to generate a self-signed certificate

To achieve this, you can use a tool such as [openSSL](#). For example, the following command line uses openSSL to generate both the private key file and a certificate signing request (CSR) file:

```
openssl req -new -nodes -newkey rsa:2048 -keyout myServer.key -out myServer.csr
```

.... where *myServer* is the name of your server. You should get *myServer.key* and *myServer.csr* files. You can then send the *myServer.csr* to the chosen certificate authority or use it to generate a self-signed certificate.

- If you send it to a certification authority, you will get in return a certificate to install in your Wakanda project (see below). You can receive a certificate in different ways (usually by E-mail or HTML form). The Wakanda Web Server accepts all platform-related text formats for certificates (Mac OS, PC, Linux, etc.). However, the certificate must be in **PKCS format**. If you get a file in a different format (.crt for example), you need to convert it into .pem format.
- If you want to generate a self-signed certificate, you can use [openSSL](#) again. For example, the following command line uses openSSL to generate a certificate ready for Wakanda:

```
openssl x509 -req -days 15 -in myServer.csr -signkey myServer.key -out cert.pem
```

Generating a key and a certificate

You can generate a private key and a self-signed certificate in a single call using [openSSL](#). For example, the following command line generates both valid *key.pem* and *cert.pem* files:

```
openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout key.pem -out cert.pem
```

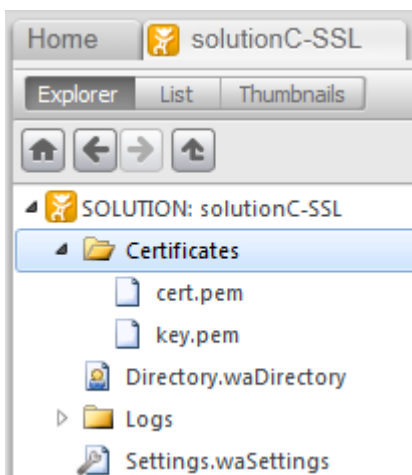
Note: Wakanda does not support encrypted private keys.

Installing SSL/TLS files in Wakanda

If you want to use the SSL/TLS protocol in your Wakanda application, the following files must be installed:

- **key.pem**: This is the file containing the private key.
If necessary, rename your private key file name manually to "key.pem".
- **cert.pem**: This is the file containing the certificate.
It must be named "cert.pem" and be in PKCS format. If you get a file in a different format, you may need to convert it into .pem format (see above).

Both files must be installed in the **Certificates** subfolder of your solution folder:



This subfolder is added by default in solutions created with Wakanda starting from version 4.

Compatibility Note: If the Certificates folder is missing or empty, Wakanda will look for cert.pem and key.pem files at the first level of your project folder.

Configuring the Settings

In order for HTTPS connections to be accepted by the Wakanda Web server, you must make sure that SSL/TLS is activated and configured in the project settings.

Several parameters are accessible in the "Secure Connections" area of the `{projectName}.waSettings` file:

Secure Connections (SSL -TLS)

Enable secure connections

Port Number: + -

Mandatory secure connections

- **Enable secure connections:** By default, secure connections are allowed. You can uncheck this option if you do not want to use HTTPS functionality with your Web server, or if another Web server allowing secure connections is operating on the same machine.
- **Port Number:** The TCP port dedicated to SSL/TLS data exchange is 443 by default. This port number can be modified in order, for example, to reinforce the security of the Web server or to resolve conflicts on the machine. The TCP port 443 is used for standard mode Web server connections.
- **Mandatory secure connections:** Check this option to force the use of the SSL protocol for all resources in the application. When checked, only HTTPS connections to the server will be allowed.

Configuring Hosting

The Wakanda HTTP Server supports multi-domain hosting, that is, a solution can contain several projects available through different domain names.

By default, Wakanda Server runs your different projects on different TCP ports, but you can easily configure your projects to support virtual multi-domain hosting.

Default Configuration for Multiple Projects

For each project in a solution, you can specify:

- a hostname or IP address
- a port number

The basic rule is that for any project in the same solution, at least one of these parameters must be different so that the server can know to which project each request should be routed.

By default when you create a new project in a new solution, it is assigned to the port 8081 and Wakanda Server increments this number for each project added in the solution. Since these ports are different from the default port (80), this means that you need to add the port number to the hostname in the URL. For example: **http://127.0.0.1:8081**

The reasons that Wakanda uses these specific port numbers are:

- to prevent conflict with another HTTP server on the developer's machine
- to make all the projects accessible via different URLs without having to specify (and buy) a

domain name

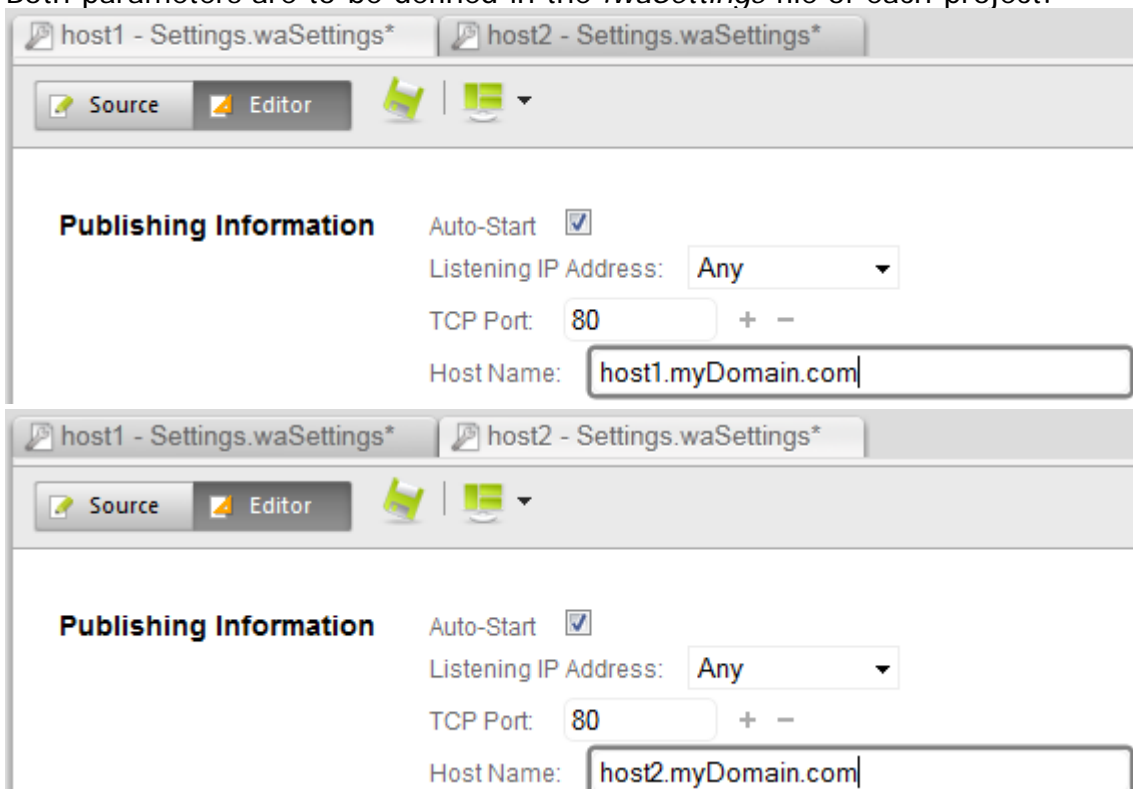
- to not rely on an internet connection to reach the server using a domain name

However, in a production context, you would use the TCP port 80 for incoming HTTP connections, even if your Wakanda Server runs several projects. In this case, you need to configure the virtual multi-hosting features of the Wakanda server.

Creating Virtual Hosts

This paragraph explains the steps required to configure a single Wakanda Server and the DNS to support several virtual hosts. Below you will find a screencast that illustrates the whole process.

1. Create a solution containing at least two projects (for example host1 and host2).
2. For each project:
 - select the same listening TCP port. For example, you can choose the port **8081**. For a production server, you would choose the port **80**, which is the default HTTP port.
 - enter a different hostname, for example **host1.myDomain.com** and **host2.myDomain.com**.Both parameters are to be defined in the `.waSettings` file of each project:



3. Configure the DNS accordingly.
In this case, you will configure **host1.myDomain.com** and **host2.myDomain.com** to route requests to the IP address of your server.
Note that domain names can be tested on a developer machine, even in offline mode. You just need to configure the local "hosts" file (for more information, please refer to <http://support.ecenica.com/domain-names/edit-hosts-file-mac-windows/> or watch [this part](#) of the screencast below).
4. To connect to your projects, enter in a browser:
 - **host1.myDomain.com** (if using default port 80) or **host1.myDomain.com:8081** (if using specific port) to connect to the first project
 - **host2.myDomain.com** (if using default port 80) or **host2.myDomain.com:8081** (if using specific port) to connect to the second project

Mime Types Support

All file MIME types handled by Wakanda Server are defined in the `MimeType.xml` file, stored at the following location:

{Wakanda Server}/Native Components/HTTPServer.bundle/Contents/Resources/

This file contains all the MIME types recognized by Wakanda Server, along with their associated extension(s) and the compressibility option. For example:

- `<contentType contentType="image/jpeg" extensions="jpeg;jpg;jpe"/>`
-> JPEG files, not compressible
- `<contentType compressible="true" contentType="text/html" extensions="html;htm;shtml;shtm" parsable="true"/>`
-> HTML files, compressible

Note: You can ignore the "parsable" option.

Configuring Settings Files

Settings files are loaded by Wakanda Server and allows you to define many parameters for your Web application. Two main settings files are available:

- **Settings.waSettings** located in the **Solution folder**: This XML file defines the settings for your solution.
- **Settings.waSettings** located in the **Project folder**: This XML file defines the customizable settings for your project.

Note: In Wakanda Studio, the Auto Hide Wakanda Extension option can be set using the Explorer contextual menu. In this case, extensions are hidden and the settings file only show "Settings".

As settings are XML files, they can be edited with any text or XML editor.

In Wakanda Studio, XML elements can be edited in the [Code Editor](#) but most settings are also available through a graphical interface. For more information, refer to the [Solutions](#) and [Projects](#) sections in the *Wakanda Studio Reference Guide*.

Solution Settings File

The Settings file for your Wakanda solution is an XML file containing several properties that are used for your entire solution. The Settings file is named **Settings.waSettings** and is located in your solution's folder. The XML file appears as shown below with the description of the different properties in the following sections:

```
<?xml version="1.0" encoding="UTF-8"?>
<settings>
  <solution>
    <serverStartup stopIfProjectFails="true"/>
    <directory authenticationType="basic"/>
  </solution>
  <database adaptiveCache="false" memoryForOtherApplications="512" memoryForCache="50"
minimumSize="100"
maximumSize="400" fixedSize="200" keepCacheInMemory="true"
flushDataCacheInterval="15"/>
</settings>
```

solution

The "solution" root element has the following elements and attributes:

serverStartup

Attribute	Default	Description
stopIfProjectFails	true	Defines if server should be stopped when Wakanda fails to open one of the projects in the solution. Accepted values: "true" or "false".

directory

Attribute	Default	Description
authenticationType	basic	Defines the authentication mode for your solution. The accepted values are: "basic," "digest," or "custom". For more information, please refer to the " Authenticating Users " chapter in the " Data Security and Access Control " manual.

database

The "database" element has the following attributes:

Attribute	Default	Description
adaptiveCache	false	Activation of the server's adaptive cache. Accepted values: "true" or "false". If it is set to "false," the fixedSize will be used.
memoryForOtherApplications	512	Memory to be reserved for other applications and the system (in MB)
memoryForCache	50	Percentage of the remaining memory allocated to the cache by default
minimumSize	100	Minimum amount of memory reserved for the cache (in MB)
maximumSize	400	Maximum amount of memory that can be used by the cache (in MB)
fixedSize	200	Fixed size of memory to be used by the cache when adaptiveCache is false (in MB)
keepCacheInMemory	true	Allows you to force the cache to be kept in the physical memory of the machine. Accepted values: "true" or "false"
flushDataCacheInterval	15	Specifies the time period between each automatic saving of the data cache (in seconds)

Project Settings File

Your project's settings file is an XML file that contains the settings for your project whose elements are defined in the following sections.

Your Wakanda project's settings file is an XML file containing several properties that are used for your project. The settings file is named **Settings.waSettings** and is located in your project's folder. Here is the contents of your project's settings file (in XML):

```
<?xml version="1.0" encoding="UTF-8"?><settings>
<project publicName="" listen="0" hostName="localhost" responseFormat="json"
administrator="false">
  <database>
    <journal enabled="true" journalFolder="."/"/>
    <autoRecovery integrateJournal="true" restoreFromLastBackup="true"/>
  </database>
</project>
<http autoStart="true" port="8081" SSLMandatory="false" SSLPort="443" useCache="false"
pageCacheSize="5242880" cachedObjectMaxSize="524288" acceptKeepAliveConnections="true"
keepAliveMaxRequests="100" keepAliveTimeOut="15" logFormat="ELF" logTokens="BYTES-SENT;C-
DNS;C-IP;CS(COOKIE);CS(HOST);CS(REFERER);CS(USER-AGENT);USER;METHOD;CS-SIP;STATUS;CS-
URI;CS-URI-QUERY;CS-URI-STEM;DATE;TIME;TRANSFERT_TIME;" logPath="Logs/"
```

```

logFileName="HTTPServer.waLog" logMaxSize="10240" allowCompression="true"
compressionMinThreshold="1024" compressionMaxThreshold="10485760"/>
<service name="webApp" modulePath="services/webApp" enabled="true" autoStart="true"/>
<service name="rpc" modulePath="services/rpc" enabled="true" autoStart="true"
proxyPattern="^/rpc-proxy/" publishInClientGlobalNamespace="false"/>
<service name="dataStore" modulePath="services/dataStore" enabled="true"
autoStart="true"/>
<service name="upload" modulePath="services/upload" enabled="true" autoStart="true"/>
<service name="remoteFileExplorer" modulePath="services/remoteFileExplorer"
enabled="false" autoStart="false"/>
<service name="Git HTTP Service" modulePath="services/waf-git/waf-GitService"
enabled="true"/>
<resources location="/walib/" lifeTime="31536000"/>
<javascript reuseContexts="true"/>
</settings>

```

project

The "project" element has the following attributes:

Attribute	Default	Description
publicName		Name of the project for "bonjour" service (see Debugging on a Remote Server).
listen	0	The IP addresses for the corresponding project. The server can listen to several IP for one project (=application). On localhost this value is "0". See Configuring Hosting section.
hostName	localhost	Hostname associated to the project of the solution. Hostnames may be simple names consisting of a single word. See Configuring Hosting section.
responseFormat	json	Format of the response from the server, can be "json", "text", or "XML".
administrator	false	Set this value to "true" if you want to use the current project as the administration project for the current solution. The "Administration" project provided by Wakanda will be used by default if none of the projects in the solution have this attribute set to "true".

database

Attributes in the "database" element allow configuring journal and automatic recovery settings. For more information about the backup features in Wakanda, please refer to the [Backup and Restore](#) section.

journal

Attribute	Default	Description
enabled	true	Allows you to enable/disable database journal.
journalFolder	./	Path to the database journal file (or "." if in the DataFolder).

autoRecovery

Attribute	Default	Description
integrateJournal		Integrate journal when datastore is not up to date with the journal (true/false).
restoreFromLastBackup		Restore damaged datastore with last backup (true/false).

http

The "http" element configures the Wakanda HTTP server and has the following attributes:

Attribute	Default	Description
autoStart	true	Enables/disables the HTTP server for the project at launch. You can manage this property at runtime using methods from the HttpServer class. Accepted values: true/false
port	8081	The TCP/IP port to be used when the HTTP server is started. This value will be incremented by one for each project added to the solution. 8080 is used for the default administration project. See Configuring Hosting section.
allowSSL	false	Allows you to activate/deactivate the SSL protocol usage for the current application ("true" or "false") (see Configuring secure connections (SSL/TLS) section)
SSLMandatory	false	Force the use of the SSL protocol for all resources in application ("true" or "false")
SSLPort	443	Sets the TCP/IP port used by the HTTP server for secured HTTP connections over SSL (HTTPS protocol).
useCache	false	Use Wakanda's cache for pages (see also cache property)
pageCacheSize	5120	Size for the HTML page cache (in Kb)
cachedObjectMaxSize	524288	Maximum object size in bytes
acceptKeepAliveConnections	true	Allows you to enable/disable the keep-alive connections ("true" or "false")
keepAliveMaxRequests	100	Maximum number of requests by connection
keepAliveTimeOut	15	Maximum timeout (in seconds) for keep-alive connections
logFormat	ELF	Allows you to set the log format. This value can be "ELF" (Extended log format), "CLF" (Common log format), and "DLF" (Combined log format).
logTokens		If you select "ELF" as the logFormat, you must define the log tokens. For more information, refer to the Web Log section.
logPath	Logs/	Path to the HTTP server log file
logFileName	HTTPServer.waLog	File name for the HTTP server log
logMaxSize	10000	Maximum log file size in bytes.
allowCompression	True	Enable text compression
compressionMinThreshold	1024	Minimum compression threshold (size in bytes)
compressionMaxThreshold	10485760	Maximum compression threshold (size in bytes)

service

The "service" elements manage the various Wakanda services and has the following attributes:

Attribute	Default	Description
name		Name of the service to configure. The available services are: "webApp": allows you to server static Web pages "dataStore": handles access to the REST interface in the

Wakanda application (internal service)

"rpc": handles access to the JSON-RPC services in the Wakanda application

"upload": upload services are required for the [File Upload](#) widget

"Git HTTP Service": handles access to git service (Added to Wakanda 3)

"remoteFileExplorer": allows you to browse files on the server (Added to Wakanda 4)

You may also have custom services if they are defined as SSJS modules [Using Custom Services](#)

modulePath	services/ <i>serviceName</i>	Path to the SSJS module file that handles the service (within the Modules folder)
enabled	true	Allows you to enable/disable the service for the current project. Either "true" or "false".
autoStart	true	Allows you to automatically start the service for the current project. Either "true" or "false".

For the "upload" server, the following properties were also added:

Attribute Default Description

maxSize	"kb"	Maximum file size to upload
maxFiles		Maximum number of files to upload
sizeUnity	"kb"	File size type (kb, mb, or byte)

resources

This setting defines the lifetime of the resources in the client and server cache. Typically, the goal is to avoid receiving too many requests for resources that rarely change. The "resources" element has the following attributes:

Attribute Default Description

location	/walib/	Location of the resources to store in cache
lifeTime	31536000	Expressed in seconds (the default is equivalent to one year)

JavaScript

This setting allows you to reuse JavaScript contexts from one request to another. Everything that is loaded in a request ([include\(\)](#), functions, etc.) remains available in the context, which is very practical for RPC requests (cf. [Using JSON-RPC Services](#)). The "JavaScript" element has the following attribute:

Attribute Default Description

reuseContexts	true	Specify if you want contexts to be reused.
---------------	------	--

Note: When the context is reused, Wakanda verifies if the loaded files were modified. If they were modified, the context is invalid and therefore not reused and another context is generated.

Wakanda Server Features

Evaluating a JS script using a Command Line

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 [openDataStore\(\)](#) 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 [Launching Wakanda Server using a Command Line](#)).

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

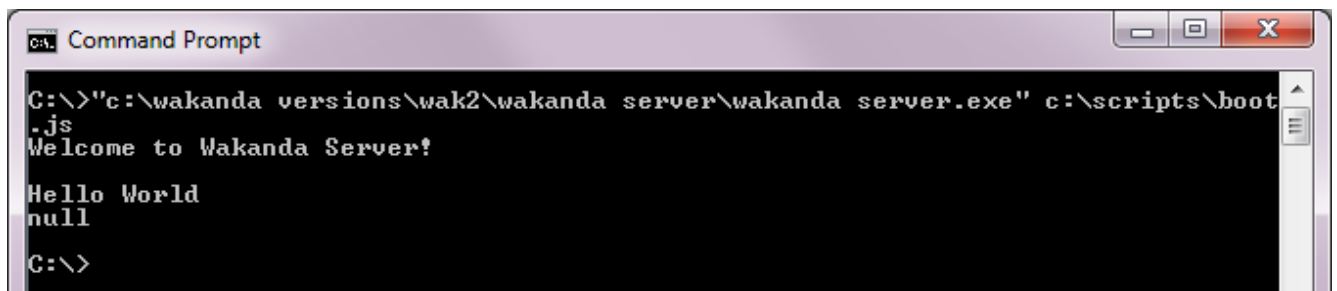
On Windows, we want to print the classic "Hello World" message in the console.

- 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:



```
ca. Command Prompt
C:\>"c:\wakanda versions\wak2\wakanda server\wakanda server.exe" c:\scripts\boot
.js
Welcome to Wakanda Server!

Hello World
null
C:\>
```

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\(\)](#)

/cache and /debugInfos URLs

Wakanda Server accepts two specific URLs: `/cache` and `/debugInfos`. These URLs can be helpful for establishing diagnostics about the built-in HTTP server status.

- **/cache**: returns the current contents of the HTTP server cache. Note that you can handle the server cache in JavaScript through the `HttpServer.cache` property and the [HttpServerCache](#) class.
- **/debugInfos**: returns internal information about the server and the running solution. This information is mainly useful for the Wakanda technical team.

`/debuginfos` and `/cache` URLs are only available to a user belonging to the "Admin" group (if the Wakanda admin access control is activated for the solution, refer to the [Configuring Admin Access Control](#) section). If the Wakanda admin access control has not been activated, these URLs are available to all users.

Syntax and Scope

To call the `/cache` or `/debugInfos` URL, you must use the following syntax:

```
http://{ project address }:{ project port }/urlName[?options]
```

For example :

```
http://www.myapp.com:8080/cache?format=xml
```

Although you have to pass a project identifier, both URLs return global information relative to all projects opened by the Wakanda Server (including the default admin project, usually published on port 8080).

options

You can use the *options* part to set the return format and the indentation of the returned text:

- **?format=xml** or **?format=json**
Format of the returned text.
By default (if omitted), `format=json`
- **?pretty=yes** or **?pretty=no**
Sets auto indentation for the returned text.
By default (if omitted), `pretty=no`

Example of returned cache information in JSON format, indented, and displayed in a browser:

```
{
  "cacheUsage" : 1,
  "numOfLoads" : 13,
  "currentSize" : 145431,
  "maxSize" : 10485760,
  "objectMaxSize" : 524288,
  "enabled" : true,
  "nbCachedObjects" : 12,
  "cachedObjects" :
  [
    {
      "url" : "/walib/WAF/widget/css/images/emptyImage.png",
      "mimeType" : "image/png",
      "isStatic" : true,
      "nbLoads" : 2,
      "lastModified" : "2012-05-14T08:56:04Z",
      "dataSize" : 110,
      "entryDate" : "2012-09-03T09:47:40Z",
      "maxAge" : 0,
      "expirationDate" : "2013-09-03T09:47:37Z"
    },
    {
      "url" : "/scripts/index.js",
      "mimeType" : "application/javascript",
      "encoding" : "gzip",
      "isStatic" : true,
      "nbLoads" : 1,

```