



**8.5**

# Using ServerDev

# ***content***

<b>introduction</b>	<b>1</b>
Overview of the ServerDev Module	1
How the ServerDev Module Works	1
<b>configuration</b>	<b>3</b>
<b>issuing commands and getting results</b>	<b>5</b>
Connecting to the ServerDev Process	5
Format of a ServerDev Command	6
Processing Results	7
Format of a Result	8
<b>list of ServerDev commands</b>	<b>9</b>
System Commands	10
Room Commands	11
Site Commands	15
Messaging Commands	18
<b>troubleshooting</b>	<b>20</b>
Usage Problems / Questions	20
Support	21



# **introduction**

---

## **Overview of the ServerDev Module**

The ServerDev module is a powerful add-on component to the ChatBlazer architecture. Using the ServerDev module, you can issue commands to the ChatBlazer server and retrieve information on your current chat status without having to go through any of ChatBlazer's administrative client applets. The ability to perform administrative functions through this avenue adds a new dimension of dynamic integration with your web site. In fact, you can even build an entirely HTML-based form to control the ChatBlazer server!

Usage of the ServerDev module can be slightly technical at times and may require certain technical knowledge in some areas. Nonetheless, this manual serves to explain as clearly as possible what you can do with the ServerDev module and will attempt to elucidate those areas which are perhaps more difficult to understand.

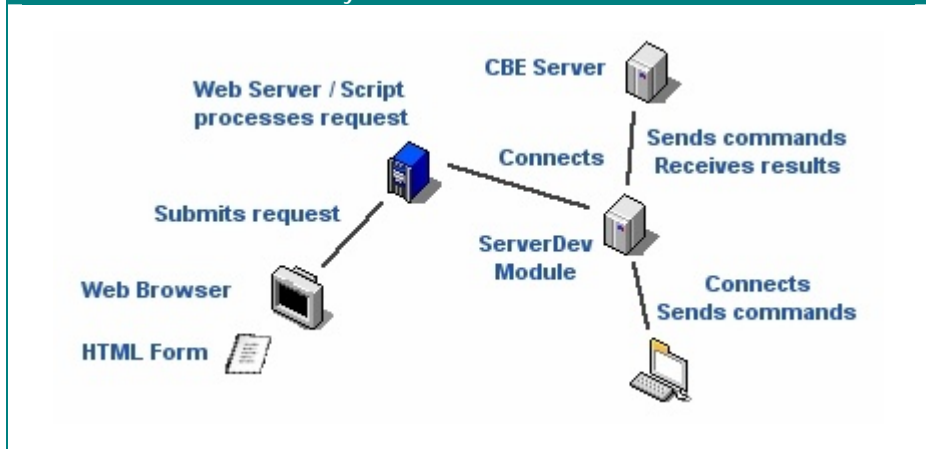
---

## **How the ServerDev Module Works**

Essentially, the ServerDev module is another server-side process started by the main ChatBlazer server that listens specifically for ServerDev commands. This server does not interact with any of ChatBlazer's Java applet clients. Most of the time, you will most probably use a server-side script, program (e.g. Perl, Bash, Tcl, Java servlet, etc.) or a web scripting language (e.g. ASP, PHP, ColdFusion, JSP, etc.) to communicate with the ServerDev process. Thus the ServerDev module is platform and language neutral.

The ServerDev server runs on your machine and only accepts local connections. Local connections can be made by the abovementioned scripts or programs when they are executed by their respective compilers or application servers. You can even Telnet and open a local connection to the server and issue commands using your Telnet client.

Figure 1. ServerDev architecture  
 Note that the web server, the ServerDev module and the ChatBlazer server are actually on the same machine.



This restriction is necessary due to potential security problems if connections are accepted from any IP address. In the next update to this module, we will most likely include a trusted list of IP addresses where connections will also be accepted.

Once a connection is received by the ServerDev module, it processes your commands and arguments much like how a HTTP POST and GET requests are processed. Once a command is pre-processed by the ServerDev module, it dispatches the command to the ChatBlazer Server. The ChatBlazer server acts on it and returns any result to the module. Subsequently, those results are returned to the calling script or program.

### No Additional Installation

The ServerDev module does not require any additional installation over and above those you may have already performed for your licensed copy of ChatBlazer 8. The module is inbuilt into the ChatBlazer server program binary file, `cb8svr.jar`, found in `chatblazer8/server/lib` (default installation). No other files are required.



## configuration

The ServerDev module's configuration parameters are found together with the ChatBlazer server's configuration file, `conf.xml`.

In `conf.xml`, you should see a module parameter section for the ServerDev. The attribute "active" can be set to "0" to disable the module.

```
<module name="serverdev" active="true">
  <param name="port" value="18002" />
  <param name="hosts.allow" value="" />
</module>
```

The table below lists the parameters that are pertinent to the module.

Parameter	Value	Description
port	any number between 0 and 65535, excluding 0	<p>This is the port number where the ServerDev module will listen for incoming connections. This number should not be the same as the port number used by the ChatBlazer server for normal connections.</p> <p>If you do not specify a valid port number here, the ServerDev process will not start. You can put a value of "-1" (in fact any invalid number) to disable the module.</p> <p>You should take note of the value for this parameter as this will be the port number that your script or program will be making a connection to.</p>

---

hosts.allow	list of hostnames and addresses separated by commas	This list of hostnames and IP addresses are the permitted remote addresses that can connect to the ServerDev process. You can use a '*' to indicate that all hosts are accepted but this is not recommended.  Leaving the value blank means that you will only accept connections made from "localhost".
-------------	---	--

---

If your port setting for the ServerDev module is correct and the port is not already bound to another process, you will see an indication that the module has started when you launch the ChatBlazer server.

When the ChatBlazer server starts, look out for this line.

```
[NORMAL]ServerDev started on port: 18002
```



## issuing commands and getting results

### Connecting to the ServerDev Process

Before you can issue commands to the ServerDev module, you need to first make a connection to it so that you can send commands and receive any desired result.

In the simplest case, you can make a Telnet connection on localhost.

```
>telnet localhost 18002
```

For the most part you should be making a connection with your script or program, like the Perl fragment given below.

```
$iaddr = inet_aton('localhost');
$paddr = sockaddr_in($SERVERDEV_PORT, $iaddr);
socket(SOCK, PF_INET, SOCK_STREAM, getprotobyname('tcp')) or
print_error("socket: $!");
connect(SOCK, $paddr) or print_error("connect: $!");
SOCK->autoflush();
```

In the case of this Perl fragment, your original request could actually have started on some HTML page that contains a <FORM> element that submits its requests to this Perl script. When the Perl script is run, the Perl interpreter will make a connection to the ServerDev module as instructed by the lines of code above.

After you have made a connection to the ServerDev module, you are now ready to start issuing it commands. You can use a connection to issue more than one command. Every connection is actually a session of commands, whereby preceding commands can affect following commands.

You can terminate a connection simply by sending an empty line to the process. In fact, any line sent to the module that does not conform to the format described hereafter will cause the connection to be terminated.

## Format of a ServerDev Command

To issue commands to the ServerDev module, you need to understand the format of commands, so called ServerDev commands, which the process expects to receive.

In simple BNF (Backus Naur Form), the format of a ServerDev command is this.

```
command ::= <command_code> <LF> { <argument> <LF> } <LF>
command_code ::= digit digit digit
argument ::= <key> { '=' <value> }
key ::= string
value ::= string
LF ::= line-feed character, i.e. '\n'
```

From the grammar above, every ServerDev command starts with a three digit command code, which is listed in the table below. Essentially this means that the command codes define the possibilities of what you can accomplish with the ServerDev module. The next chapter will describe every ServerDev command in detail.

After sending the command code, you can now send any necessary argument. Each argument should occupy a single line. Argument keys will never have a line-feed character. If your argument value has a line-feed character, you need to encode the value in a similar way to how URLs are encoded, line-feed characters are substituted by "%20". When values are encoded, you need to firstly indicate in your connection session that values are encoded. As stated by the grammar, not all arguments are key-value pairs. An argument can solely be a key value.



Arguments do not need to be sent in a specific order.

Do not forget to send an empty line (a single line-feed character) after you are done sending all your arguments. This tells the ServerDev module that it should not expect any more arguments and that it can start to process your issued command.

Following are some examples showing valid commands being issued to the module.

1. Change character-encoding henceforth to Windows Cp1256. (Note the command-terminating line-feed at the end.)

```
100
enc=Cp1256
```

2. Indicate that values are encoded. Set the welcome message of a room. You can issue multiple commands in a single connection session.

```
102
203
site=CBS1
room=The Lobby
welcome=Hello and welcome!%20Have a nice day!
```

Once you have issued a valid (in terms of conforming to the format, not correctness of your arguments) command, the ServerDev module will process it and return you the results. The following section will describe the format of results that are returned.

Do note that every issued command will always have a result returned, even though the command may seem to be a non-interactive one, e.g. toggling of values being encoded.



If you have sent an invalid (non-conforming) command, your connection will actually be terminated.

---

## Processing Results

When you are done with issuing your command and expect some results to be returned, you need to open a receiving I/O stream that you will use to read results that are being returned.

The following Perl fragment is an example of this.

```
# Put messages sent by server into array.
my $line = "";
READ: while ($line = <SOCK>) {
    chomp $line;
    push @reply, ($line);
    last READ if ($line eq "");
}
```

The example actually shows a “greedy” method of processing results, that is all results are actually read into a data structure (array in this case) before they are processed by the script. You can choose to process the results as you read them.

## Format of a Result

Again, in simple BNF, the format of any result returned by the module follows the grammar below.

```

result ::= <result_code> '-' <op_code> <LF>
{ <result_body> } <LF>
result_code ::= digit digit digit
op_code ::= digit | digit digit
result_body ::= error | result
error ::= string
result ::= string
LF ::= line-feed character, i.e. '\n'

```

The table below gives the set of result codes.

Result Code	Description
900	Command executed successfully.
902	Command failed to execute.
904	Command contained missing or illegal arguments.

Op codes are single or two-digit values. They are not absolute in their meaning, they depend on the commands that you have issued. You need to refer to the detailed description of a command to understand the possible op code values that are returned.

Each result returned may have a result body. Some may not, like the toggling of encoded values or changing of character-encoding. If you expect a list of values to be returned to you (list of user names, rooms, etc.), each value will occupy a single line. This means results are delimited by line-feed characters. You should read the entire result body until you encounter an empty line. The empty line terminates the returned result.

If your command is valid but incorrect, you will always get an error message indicating the problem with your issued command. If the server encountered some unexpected problems when processing your command, you will also get an error message.



## ServerDev commands

This chapter enumerates the currently supported set of commands that are understood by the ServerDev module.

For certain commands, some arguments are necessary while others may be optional.

Below is the index of available commands.

Command Code	Description
100	Change current character-encoding of the ServerDev process.
101	Toggle encoded values processing.
201	Create room
202	Delete room
203	Set room's property
204	Get room's property
205	Get the list of rooms in chat site
206	Rename room
207	Refresh room (reload from database)
210	<i>File uploaded (system-use only)</i>
212	Create private chat
301	Get the list of users in chat site
302	Logout user
303	Check to see if user exists
304	Get number of users in a chat site
305	Get buddy list of a member
306	Refresh / reload user group
307	Delete user group
309	Create user group
310	Create user
311	Update user
401	Send an administrator message to a site or a room
501	Create site
502	Set site's configuration

## System Commands

These commands affect how other commands are processed and how results are returned.

### Change character encoding

<i>Command Code</i>	100
<i>Required Argument(s)</i>	enc=new encoding value
<i>Optional Argument(s)</i>	None
<i>Op Code(s)</i>	None

#### *Description*

Changes the character-encoding that is received and sent by the ServerDev module. This change is permanent and will take effect for all commands issued henceforth and for future connection sessions.

Be careful when using this command. If you have changed the encoding, you need to be able to send commands and arguments in that set encoding or else your commands may become garbled when received by the ServerDev module and may result in your connections being closed by the module.

### Toggle encoded values

<i>Command Code</i>	101
<i>Required Argument(s)</i>	None
<i>Optional Argument(s)</i>	None
<i>Op Code(s)</i>	None

#### *Description*

This command tells the ServerDev module to decode arguments first before processing them. Decoding arguments means that the value portion of an argument is decoded. It also encodes results before returning them to your calling script or program. Encoding and decoding works exactly like how URLs are encoded and decoded.

Unlike the change of character encoding, this command persists on a session-basis. Each session starts with this disabled. Each issue of this command toggles the previous value, e.g. send for first time, encoding is enabled; send again and encoding is disabled.

## Room Commands

These commands perform room administrative functions. You need to provide at least two arguments, the site which the room belongs to, and the room name itself.

### Create room

<i>Command Code</i>	201
<i>Required Argument(s)</i>	site=site id room=new room name
<i>Optional Argument(s)</i>	room.creator room.parent=ID of parent group room.type="group"
<i>Op Code(s)</i>	0 – Failed when trying to save the room, most probably database error 1 – Success 2 – Room with same name exists 3 – Room limit exceeded 4 – Parent room group does not exist (specified by <i>room.parent</i> )

#### Description

Creates a new room in the site specified. An optional "room.creator" argument allows you to create a room from a user's perspective. If "room.parent" is not specified, the room is created at the root level. To create a room group, the value of the "room.type" argument must be "group". Any other value will mean a normal room is to be created.

The room ID of the new room is returned in the result field if the room is successfully created.

### Delete room

<i>Command Code</i>	202
<i>Required Argument(s)</i>	site=site id room.id=ID of room / room group
<i>Optional Argument(s)</i>	None
<i>Op Code(s)</i>	0 – Failed when trying to delete the room or room group, most probably database error 1 – Success

#### Description

Deletes a room or a room group in the specified site.

**Rename room**

---

**Command Code** 206

---

**Required Argument(s)** site=site id  
room.id=ID of room to change  
room.newName=new name of the room

---

**Optional Argument(s)** None

---

**Op Code(s)** 0 – Failed when trying to rename the room  
1 – Success

---

**Description**Renames a room in the site specified.

---

### Set a room's property

<i>Command Code</i>	203
<i>Required Argument(s)</i>	site=site id room.id=ID of room
<i>Optional Argument(s)</i>	allow.guest=0 1 allow.all=0 1 allow.spectator=0 1 private=0 1 toggle.profanity=0 1 schedule.active=0 1 close.logout=0 1 schedule.list=list of room opening times, following the format given in the admin client history=number of messages in the history list to keep pass=new room password welcome=welcome message url=room logout URL banner=room banner slideshow room.limit=room's user limit mod.set=permission set for moderated room mod.pass=pass ratio mod.limit=limit number of room moderators closed=0 1
<i>Op Code(s)</i>	0 – Failed when saving changes to the room, most probably database error 1 – Success

#### *Description*

Changes a room property and saves it to the database. You can set multiple properties with a single command.

For "mod.set" argument, the format of the permission set is this.

*ABCDEF*

Each character is a binary 0 or 1.

- A – set room moderated
- B – pass on exit
- C – cannot talk unless allowed
- D – allow emoticon/audio
- E – allow whisper
- F – queue messages

E.g. '100111' means set room as moderated, allow emoticon/audio, allow whisper and queue messages when there are no moderators.

**Get a room's property**

<i>Command Code</i>	204
<i>Required Argument(s)</i>	site=site id room.id=ID of room
<i>Optional Argument(s)</i>	user.count – Number of users in the room user.list – Names of all users in the room closed – Is room closed?
<i>Op Code(s)</i>	None

*Description*

Gets one or more room properties with the specified optional arguments. The results are returned in the order that you have provided the arguments. For multi-value results, each result occupies a line.

**Get site's room list**

<i>Command Code</i>	205
<i>Required Argument(s)</i>	site=site ID
<i>Optional Argument(s)</i>	None
<i>Op Code(s)</i>	None

*Description*

Returns the names of all the rooms in the specified chat site.

**Refresh room (reload from database)**

<i>Command Code</i>	207
<i>Required Argument(s)</i>	site=site ID room.id=ID of room or group
<i>Optional Argument(s)</i>	room.type=1 (room)   2 (group)
<i>Op Code(s)</i>	None

*Description*

Reloads a room's or group's properties from the database. If a group is specified, all rooms under the group will also be reloaded. The "room.type" argument does not need to be specified for refreshing of a room as it is the default.

**Create private chat**

<i>Command Code</i>	212
<i>Required Argument(s)</i>	site=site ID user.n=name of participant (n starts from 0)
<i>Optional Argument(s)</i>	av=1 (enabled)   0 (disabled, default)
<i>Op Code(s)</i>	None

*Description*

Creates a private chat session. Arguments user.0, user.1, and so on, give the names of the participants who will be joining the private chat session after it has been created. The "av" argument determines if audio-video chatting is allowed in the private chat.

## User Commands

These commands are used to manage users within chat sites.

### Get site's user list

<i>Command Code</i>	301
<i>Required Argument(s)</i>	site=site id
<i>Optional Argument(s)</i>	user.type=n, where n is the flag for the two types of users, room-based or Messenger-based. 1 for room, 2 for Messenger, add them together, 3, to get both. Default value is 3.
<i>Op Code(s)</i>	None
<i>Description</i>	Returns the names of the users (of the specified type) in the chat site. This is across all rooms.

### Logout user

<i>Command Code</i>	302
<i>Required Argument(s)</i>	site=site id user.name=user to be logged out
<i>Optional Argument(s)</i>	None
<i>Op Code(s)</i>	0 – User does not exist 1 – User logged out
<i>Description</i>	Logs out a user based on his login name.

### Check if user exists

<i>Command Code</i>	303
<i>Required Argument(s)</i>	site=site id user.name=user to find
<i>Optional Argument(s)</i>	None
<i>Op Code(s)</i>	0 – User does not exist 1 – User exists
<i>Description</i>	Looks for a user. If the user is in a room, the room's name is returned as the result.

**Get number of users in a site**

<i>Command Code</i>	304
<i>Required Argument(s)</i>	site=site id
<i>Optional Argument(s)</i>	user.type=n, where n is the flag for the two types of users, room-based or Messenger-based. 1 for room, 2 for Messenger, add them together, 3, to get both. Default value is 3.
<i>Op Code(s)</i>	None
<i>Description</i>	Returns the number of users in the chat site.

**Get buddy list of a member**

<i>Command Code</i>	305
<i>Required Argument(s)</i>	site=site ID user.name=name of member
<i>Optional Argument(s)</i>	None
<i>Op Code(s)</i>	1 – database error 2 – member does not exist
<i>Description</i>	Returns the buddy list of a member in the specified site, only for the ChatBlazer protocol (does not include Public IM networks). Each buddy's name is returned in the result.

**Refresh / reload user group**

<i>Command Code</i>	306
<i>Required Argument(s)</i>	site=site ID userGroup.id=ID of user group
<i>Optional Argument(s)</i>	None
<i>Op Code(s)</i>	0 – database error 1 – OK 2 – user group not found
<i>Description</i>	Loads an externally created user group into the ChatBlazer server memory. This can also be used to reload any existing user group. An external application can create the database record for a new user group in the database directly and then use this command to have the ChatBlazer server load this new user group.

**Delete user group**

<i>Command Code</i>	307
<i>Required Argument(s)</i>	site=site ID userGroup.id=ID of user group
<i>Optional Argument(s)</i>	None
<i>Op Code(s)</i>	0 – OK 1 – user group not found
<i>Description</i>	
Deletes the specified user group.	

**Create user group**

<i>Command Code</i>	309
<i>Required Argument(s)</i>	site=site ID userGroup.name=name of new user group
<i>Optional Argument(s)</i>	None
<i>Op Code(s)</i>	0 – OK 1 – name exists 2 – database error
<i>Description</i>	
Creates a new user group with the given name in the specified site.	

**Create user**

<i>Command Code</i>	310
<i>Required Argument(s)</i>	site=site ID name=name of user to be created password=password of user
<i>Optional Argument(s)</i>	None
<i>Op Code(s)</i>	userID of newly created user
<i>Description</i>	
Creates a new user for the site specified. The user's ID (in the database) is returned in the Op Code.	

### Update user

**Command Code** 311

**Required Argument(s)** site=site ID  
user.name=name of user to be updated

**Optional Argument(s)** save="true" | "false"  
password  
timeLeft  
(any other profile field)

**Op Code(s)**

**Description**

Updates an existing user. This command can be used for real-time updating of a user's chat usage time. The *save* attribute indicates whether the changes should be committed to the database.

## Messaging Commands

### Send administrator message

*Command Code*            401

*Required Argument(s)*   site=site id  
                                   message=your message to be sent

*Optional Argument(s)*   room=send message to this room only  
                                   sender=named sender of this message  
                                   target=targeted user this message will be sent to

*Op Code(s)*                None

#### *Description*

This command sends out a message either to all users in the specified chat site, or to a targeted room. The source of the message can be specified with the optional "sender" argument. If it is not supplied, an administrator message is sent. If the "sender" argument is given, the message is treated as a normal user message and the "room" argument is required. The "target" argument when used is to be given with the "sender" argument or else it is ignored.

## Site Commands

These commands are used to manage chat sites.

### Create site

<i>Command Code</i>	501
<i>Required Argument(s)</i>	name=name of site domain=domain assigned to this site
<i>Optional Argument(s)</i>	None
<i>Op Code(s)</i>	0 – Success 1 – no permission 2 – site limit exceeded 3 – database error

#### *Description*

Creates a new chat site. A new chat site's ID is automatically assigned by the server and is returned in the result field.

### Set site's configuration

<i>Command Code</i>	502
<i>Required Argument(s)</i>	site=ID of site
<i>Optional Argument(s)</i>	At least one of the following: userLimit roomLimit
<i>Op Code(s)</i>	0 – Success 1 – database error

#### *Description*

Updates a site's configuration.



We are continuously expanding the functionalities of the ServerDev module. If you would like to perform a certain task but it is not possible with the current set of commands, please drop us a suggestion. We will definitely look into the issue.



# troubleshooting

Common usage problems are listed below. If you find any issues that are not addressed here, please do not hesitate to contact us for assistance.

---

## Usage Problems / Questions

My script fails to connect to the ServerDev module. When it does, it is almost immediately disconnected.

Firstly, check that the ChatBlazer server has started the ServerDev process.

Secondly, check that you are making a connection from the list of allowed hostnames or IP addresses you have configured. If you have left the "hosts.allow" parameter blank, the ServerDev module will only accept connections made from "localhost" or an IP address which is identical to the residing machine. If a connection is made from anywhere not permitted, it will be disconnected immediately.

There is no authentication when I make a valid connection to the ServerDev module. Is this a potential security risk?

The ServerDev module currently does not provide any authentication mechanism, apart from the list of allowed hosts that you can specify to accept connections from. We recommend that you protect the resource that accesses the ServerDev module.

---

## Support

As you have read in our foreword, purchasing ChatBlazer is just the first step to having a real time chat solution for your business or organization. Quality support plays the other half of the equation. Pendulab is committed to providing you with fast and helpful personal support in the event that you require so.

If you have any suggestion to improve this user manual, we will appreciate your feedback.

You can reach us in the following ways.

Email      [support@pendulab.com](mailto:support@pendulab.com)

Customer      <http://www.chatblazer.com/support/index.htm>  
Portal\*

Forum      <http://customer.chatblazer.com/forum/>

Phone      United States  
              415 954 7103 (office hours)  
              866-265-1884 (toll free, available after office hours from  
              8am - 1pm ET, 9pm to next day 6am ET)

              International  
              (65) 6448-1846

\* Recommended.