פרוטוקול system cloud

[1 תקשורת עם השרת 2](#_Toc453198623)

[2 פורמט הפרוטוקול 3](#_Toc453198624)

[3 הודעות על PORT 5000 4](#_Toc453198625)

[3.1 Get version 4](#_Toc453198626)

[3.2 Sign up 5](#_Toc453198627)

[3.3 Sign in 6](#_Toc453198628)

[3.4 Log out 7](#_Toc453198629)

[3.5 Delete 8](#_Toc453198630)

[3.6 Download 9](#_Toc453198631)

[3.7 Download folder 10](#_Toc453198632)

[3.8 Get folders 11](#_Toc453198633)

[3.9 Rename 12](#_Toc453198634)

[3.10 Upload 13](#_Toc453198635)

[3.11 Create 14](#_Toc453198636)

# תקשורת עם השרת

* הפרוטוקול יהיה מעל **IP/TCP**.
* מספר ה-PORT של השרת יהיה 5000 (השרת ירוץ על 127.0.0.1)
* על כל PACKET של הלקוח, תינתן PACKET תשובה ע"י השרת.
* תשובת השרת תשלח ל-PORT ול-IP שממנו התקבלה הבקשה.

# פורמט הפרוטוקול

* כל הודעה תכיל צמדים מסוג של VALUE=**KEY**
* כל צמד יופרד ע"י התו '|' (PIPE)

לדוגמא:

**MSG**=GAME\_CREATE|**GAME\_NAME**=Yair’s War|**PLAYERS\_COUNT**=4

* כל רשימה תוקף ע"י סוגריים מרובעים: [ ], ופסיקים יפרידו בין איברי הרשימה.

לדוגמא:

**MSG**=GAMES\_LIST\_REPSONSE|**GAME\_NAME\_LIST**=[Yair’s War, Hagai’s Game, Shaul’s bloodbath]

* לכל הודעה ישנו KEY בשם **MSG**, שמכיל את סוג ההודעה.
* כל תשובה מהשרת תכיל את סוג ההודעה המקורית שנשלחה ע"י הלקוח בתוספת RESPONSE\_

לדוגמא, על הודעה MOVE\_GAME, השרת יענה עם הודעת RESPONSE\_MOVE\_GAME

* התווים **, | =** **] [** (פסיק, PIPE, שווה, פתיחת וסגירת סוגריים מרובעים) אינם תווים חוקיים למעט במקומות שהוגדרו לעיל.
* הן הלקוח והן השרת, **יתעלמו** מכל צמד **KEY**=VALUE, שהם אינם מכירים.
* במקרה של שגיאה שהשרת מצליח לזהות את מקורה, השרת יוסיף מפתח בשם **TEXT**=, עם תיאור חופשי של הבעיה בה הוא נתקל.
* כאשר השרת יזהה בעיה אך ללא הצלחה בניתוח הסיבה לבעיה, השרת יוסיף מפתח בשם **STATUS**=, שיכיל את ההודעה הבאה: FAILED\_UNKNOWN\_PROBLEM. במקרה כזה הלקוח ידפיס הודעה כי יש ליצור קשר עם בעלי השרת ויעצור את התוכנית.
* כאשר השרת יקבל הודעה שאיננה קיימת בפרוטוקול השרת ישלח בתגובה הודעה בשם ERROR ויוסיף לה מפתח בשם **STATUS**=, שיכיל את ההודעה הבאה: FAILED\_COMMEND\_NOT\_FOUND
* כאשר השרת יקבל הודעה ויכשל בביצועה עקב שגיאה כלשהי שהשרת זיהה, השרת יחזיר הודעה ובמקום STATUSS=SUCCESS הוא יחזיר ב STATUSS את סיבת השגיאה למשל: FAILED\_ALREADY\_EXIST .
* כאשר הלקוח יבצע לעשות פעולות שמצריכות כניסה לחשבון משתמש מבלי שהלקוח נכנס לחשבון השתמש שלו בפרוטוקול השרת ישלח בתגובה הודעה בשם ERROR ויוסיף לה מפתח בשם **STATUS**=, שיכיל את ההודעה הבאה: FAILED\_NO\_ACCESS\_PERMISSION .

# הודעות על PORT 5000

## Get version

|  |  |
| --- | --- |
| Get server version  Client -> Server  Server always responds with GET\_VERSION\_RESPONSE MSG. | |
| MSG | **GET\_VERSION** |

Example: **MSG**=GET\_VERSION

|  |  |
| --- | --- |
| Game server version Response  Server -> Client  Response to Game List | |
| MSG | **GET\_VERSION\_RESPONSE** |
| VERSION | <Integer>  Version number |

## Sign up

|  |  |
| --- | --- |
| Creates a new user account on the server.  (user will be signed in automatically after creating an account)  Client -> Server  Server always responds with SIGN\_UP\_RESPONSE MSG. | |
| MSG | **SIGN\_UP** |
| USER\_NAME | <string>  The username of the user being created  Forbidden to contain one of: [/ \ @ > < ? \* : !" ' ] |
| PASSWORD | <string>  The password of the user being created  Forbidden to contain one of: [/ \ @ > < ? \* : !" ' ] |

Example: **MSG**=SIGN\_UP|**USER\_NAME**=shaked|**PASSWORD**=167daniel

|  |  |
| --- | --- |
| Sign up Response  Server -> Client | |
| MSG | **SIGN\_UP\_RESPONSE** |
| STATUS | <string>  One of [SUCCESS, FAILED\_ALREADY\_EXISTS, FAILED\_ILLEGAL\_CHAR,FAILED\_UNKNOWN\_PROBLEM [ |

## Sign in

|  |  |
| --- | --- |
| Signing in with a user account that already exists  Client -> Server  Server always responds with SIGN\_IN\_RESPONSE MSG. | |
| MSG | **SIGN\_IN** |
| USER\_NAME | <string>  The username of the user being signed in |
| PASSWORD | <string>  The password of the user being signed in |

Example: **MSG**=SIGN\_IN|**USER\_NAME**=shaked|**PASSWORD**=167daniel

|  |  |
| --- | --- |
| Sign in response  Server -> Client | |
| MSG | **SIGN\_IN\_RESPONSE** |
| STATUS | <string>  One of [SUCCESS, FAILED\_PASSWORD\_INCORRECT, FAILED\_ UNRECOGNIZED\_USER\_NAME, FAILED\_UNKNOWN\_PROBLEM] |

## Log out

|  |  |
| --- | --- |
| Logging out from the server  Client -> Server  Server always responds with LOG\_OUT\_RESPONSE MSG. | |
| MSG | **LOG\_OUT** |

Example: **MSG**=LOG\_OUT

|  |  |
| --- | --- |
| Log out Response  Server -> Client | |
| MSG | **LOG\_OUT\_RESPONSE** |
| STATUS | One of [SUCCESS, FAILED\_UNKNOWN\_PROBLEM] |

## Delete

|  |  |
| --- | --- |
| The client deletes a folder or a file from his synchronization folders. As a result of that the client sends a delete message that announces to the cloud about the delete action.  Client -> Server  Server always responds with DELETE\_RESPONSE MSG. | |
| MSG | **DELETE** |
| PATH | <string>  Path of the file or folder to be deleted. The path starts at one of the main sync folders. |

Example: **MSG**=DELETE|**PATH\_NAME**=”c:\”

|  |  |
| --- | --- |
| delete Response  Server -> Client | |
| MSG | **DELETE\_RESPONSE** |
| STATUS | One of [SUCCESS, FAILED\_UNKNOWN\_PROBLEM, FAILED\_PATH\_DOES\_NOT\_EXIST] |

## Download

|  |  |
| --- | --- |
| Download a file from the server  Client -> Server  Server always responds with DOWNLOAD\_RESPONSE MSG. | |
| MSG | **DOWNLOAD** |
| PATH | <string>  The path of the file\folder found in the server. |

Example: **MSG**=DOWNLOAD|**PATH**=c:\

|  |  |
| --- | --- |
| download Response  Server -> Client | |
| MSG | **DOWNLOAD \_RESPONSE** |
| FILE | <binary> (compressed) (represented by string)  The file that was asked to be downloaded  (is being returned just if status == success) |
| STATUS | One of [SUCCESS, FAILED\_UNKNOWN\_PROBLEM, FAILED\_PATH\_DOES\_NOT\_EXISTS] |

## Download folder

|  |  |
| --- | --- |
| Download a directory from the server  Client -> Server  Server always responds with DOWNLOAD\_FOLDER\_RESPONSE MSG. | |
| MSG | **DOWNLOAD\_FOLDER** |
| PATH | <string>  The path of the file\folder found in the server. |

Example: **MSG**=DOWNLOAD\_FOLDER|**PATH**=c:\

|  |  |
| --- | --- |
| Download folder Response  Server -> Client | |
| MSG | **DOWNLOAD\_FOLDER \_RESPONSE** |
| FOLDER\_DATA | <binary> (compressed) (represented by string)  The folder that was asked to be downloaded zipped  (is being returned just if status == success) |
| STATUS | One of [SUCCESS, FAILED\_UNKNOWN\_PROBLEM, FAILED\_PATH\_DOES\_NOT\_EXISTS] |

## Get folders

|  |  |
| --- | --- |
| Get the folder that belong to the user\_name that has been signed in.  Client -> Server  Server always responds with GET\_FOLDERS\_RESPONSE MSG. | |
| MSG | **GET\_FOLDERS** |

Example: **MSG**=GET\_FOLDERS|**PATH**=c:\

|  |  |
| --- | --- |
| Get folders Response  Server -> Client | |
| MSG | **GET\_FOLDERS \_RESPONSE** |
| FOLDERS\_DICT | <dict> (represented by string)  A dict of the folders that belongs to the user. The key is their virtual name and the value is their guid name |
| STATUS | One of [SUCCESS, FAILED\_UNKNOWN\_PROBLEM] |

## Rename

|  |  |
| --- | --- |
| The client renamed a folder or a file in the sync folder. As a result of that the clients sends to the server a message to inform the server about the file or folder that was renamed.  Client -> Server  Server always responds with RENAME\_RESPONSE MSG. | |
| MSG | **RENAME** |
| PATH | <string>  The original path of the renamed file including its old name. the path starts at one of the main sync folders |
| NEW\_NAME | <string>  The new name of the folder\file |

Example: **MSG**=RENAME|**PATH\_NAME** =”c:\”|**NEW\_NAME=**”new folder”

|  |  |
| --- | --- |
| Rename Response  Server -> Client | |
| MSG | **RENAME\_RESPONSE** |
| STATUS | One of [SUCCESS, FAILED\_UNKNOWN\_PROBLEM, FAILED\_FOLDER\_NAME\_ALREADY\_ EXISTS] |

## Upload

|  |  |
| --- | --- |
| Upload a file from the sync directory to the server  Client -> Server  Server always responds with UPLOAD\_RESPONSE MSG. | |
| MSG | **UPLOAD** |
| PATH | <string>  The original path of the uploaded file\folder. The path starts at one of the main sync folders. |
| FILE\_DATA | <binary> (compressed)  The content of the file |

Example: **MSG**=UPLOAD|**PATH\_NAME** =c:\|**FILE\_DATA**=file

|  |  |
| --- | --- |
| upload Response  Server -> Client | |
| MSG | **UPLOAD\_RESPONSE** |
| STATUS | One of [SUCCESS, FAILED\_UNKNOWN\_PROBLEM, FAILED\_DATA\_CORUPTED, FAILED\_NO\_SYNC\_FOLDER] |

## Create

|  |  |
| --- | --- |
| The client creates a main sync folder. As a result the client is informing to the server about the new main sync folder  Client -> Server  Server always responds with CREATE\_RESPONSE MSG. | |
| MSG | **CREATE** |
| PATH | <string>  The original path of the uploaded file\folder. The path starts at one of the main sync folders. |
| NEW\_NAME | <string>  New folder/file name |
| IS\_FILE | <Boolean>  If the created variable is a folder: False  If the created variable is a folder: True |

Example: **MSG**=CREATE|**PATH** =c:\|**NEW\_NAME=**file.py|**IS\_FILE=**True

|  |  |
| --- | --- |
| create Response  Server -> Client | |
| MSG | **CREATE\_RESPONSE** |
| STATUS | One of [SUCCESS, FAILED\_UNKNOWN\_PROBLE, FAILED\_ALREADY\_EXISTS] |