NAME

smsync — smsync protocol

DESCRIPTION

This manual page describes the **smsync** protocol used by the **smsync** client and server to communicate with each other. All communication between the client and the server falls into one of two categories:

control messages	These are messages sent between client and server that govern the general flow of the program.
smsync packets	Any actual data transferred is encapsulated in an smsync packet.

CONTROL MESSAGES

Control messages are loosely based on the SMTP protocol control messages and are used to govern control flow between the client and the server. A control message is a string in the format

```
XYZ MESSAGE\n
```

and is at most 128 bytes long. The three digits of the reply each have a special significance. The first digit denotes whether the response is good, bad or incomplete:

1yz	A request.
2yz	A positive completion reply. The requested action has been successfully completed. A new request may be initiated.
3yz	Unused.
4yz	A transient negative completion reply. The command was not accepted, and the requested action did not occur. However, the error condition is temporary and the action may be requested again.
5yz	A prmanent negative completion reply (ie failure). The command was not accepted and the requested action did not occur.

The second digit encodes responses in specific categories:

x0z	Syntax: These replies refer to syntax errors, syntactically correct commands that do not fit any functional category, and unimplemented or superfluous commands.
x1z	Information: These are replies to requests for information, such as status or help.
x2z	Connections: These are replies referring to the transmission channel.
x3z	Authentication: These are replies referring to the authentication handshake.

CONTROL MESSAGES REFERENCE

The full list of control messages in numerical order is:

120 Send Root	Sent by the client to provide the server with the root under which to start synchronization. The root is given as an absolute path.
121 Begin Filelist	Sent by the client to initiate the transfer of the filelist.
123 Begin Indexlist	Sent by the server to initiate the transfer of the indeces.
125 Send File	Sent by the client to signal the beginning of a file transfer.
129 Terminate	Sent by the client to close the session.
130 Request Authorization	Sent by the client to request an authorization challenge.

131 Auth Response	Sent by the client to return the auhtorization response.
220 SMSYNC_VERSION [0-9.]∗ OPTIONAL	Sent by the server when a new connection is made.
221 Goodbye	Sent by the server to signal termination of the connection.
222 Accept Root	Sent by the server to signal that the requested root can be used to synchronize the file hierarchy.
223 Accept Filelist	Sent by the server to signal that the entire filelist was accepted.
224 Accept Indexlist	Sent by the client to signal that the entire indexlist was accepted.
224 Accept File	Sent by the server to signal that the last file was accepted.
229 OK	Sent to acknowledge the previous request.
230 Auth Challenge	Sent by the server to signal the start of the authorization challenge.
232 Auth Ok	Sent by the server to signal that authorization was successful.
410 Illegal Response	Sent if the previous request was not expected.
440 Root Unacceptable	Sent by the server if the requested root can not be found or permissions do not allow synchronization.
530 Access denied	Sent by the server if the authorization response did not match.

SMSYNC PACKETS

All data exchanged (other than control messages) is encapsulated in so-called smsync packets. A packets consists of:

type An 8 bit `type` field used to specify what kind of data is being transmitted. Possible types are AUTHCHALLENGE (0), AUTHRESPONSE (1), FILELIST (2), INDEXLIST (3), or FILEDATA (4).

flag An 8 bit `flag` field used to differentiate among packets of the same type. Possible flags are:

IGNORE (0)	if the flag is not relevant to this packet.
FILECOMPLETE (1)	`type` is FILEDATA and this packet is the last packet for the given file
ROOT (2)	if `type` is FILELIST and the client is sending the root to synch under to the server
END (3)	if `type` is either FILELIST or INDEXLIST. After all entries have been sent, then a zero-data smsync packet is sent with `flag` set to END.

index A 4 byte `index` field used to send a specific index during filelist or indexlist transmissions. This field is set to 0 and is ignored unless `type` is FILELIST or INDEXLIST.

length A 4 byte `length` field used to indicate how much data payload is following. This field is set to 0 and the following `data` field ignored unless `type` is FILEDATA or FILELIST.

data A variable size payload containing the actual data to be sent.

When the client is sending the filelist to the server, it sends exactly one such packet per file. When the server is sending the list of indeces to the client, it sends exactly one such packet for index. When the client is sending a file to the server, it will send as many packets as necessary.

AUTHENTICATION

The *`130 Request Authorization`* control message initiates the authentication handshake. After the server responded with a *`230 Auth Challenge`* control message and the client acknowledged the challenge with *`229 OK`*, it sends a 16 byte random string to the client, encrypted with the pre-shared secret (itself a 16 byte string). The client decrypts the message, calculates its *`md5`* checksum and sends that checksum back to the server encrypted with the pre-shared secret.

The server then decrypts the message from the client, calculates the *`md5`* checksum of the random string it generated earlier and compares it to the decrypted string it received from the client. If they match, the server sends back *`232 AUTH OK`* and both sides use the random string as the *`initialization vector`* for the subsequent CBC encryption (no initialization vector is used for the previous exchange).

EXAMPLE COMMUNICATION

S:	220 SMSYNC_VERSION 0.1 smsync-765-a
C:	130 Request Authorization
S:	230 Auth Challenge
C:	229 OK
S:	...
C:	131 Auth Response
S:	229 OK
C:	...
S:	232 AUTH OK
C:	120 Send Root
S:	224 OK
C:	...
S:	222 Accept Root
C:	121 Begin filelist
S:	229 OK
C:	... ...
S:	223 Accept filelist
S:	123 Begin index list
C:	229 OK
S:	... ...
C:	224 Accept index list
C:	125 Send file
S:	229 OK
C:	... ...
S:	225 accept file

C:	129 terminate
S:	221 Goodbye

SEE ALSO

`smsync`(1), `smsyncd`(8), `mtree`(1)

HISTORY

The **`smsync`** protocol was developed as part of the the final project for the class CS765 “Advanced Programming in the UNIX Environment” during the fall semester of 2005 at Stevens Institute of Technology.