HomeSome application layer protocol for client-server TCP communication

Communication nodes:

PS = Public Server
H = Hub
G = Gadget
AC = Android Client
WC = Web Client
C = Client (both AC & WC)

Not relevant	⬛ 0
Not implemented	🟥 1
Ongoing implementaion	🟧 2
Implemented, not tested / approved	🟦 3
Implemented, tested and approved	🟩 4

Information			Syntax (by index)					Comment	Revision control					
Function	Precondition	Direction	0	1	2	3	4							v.
C login & logout									AC	WC	PS	H	G	
C maual login.		C → PS	101	C_nameID	C_pwd			If valid: H generates new *C_sessionKey*.	🟩	🟩	🟩	⬛	⬛	1.0
Successful result of C manual login attempt.	#101.	PS → C	102	C_nameID	C_isAdmin	H_alias	C_sessionKey	C strores *C_sessionKey* locally. Note: Unsuccessful login result: #903	🟩	🟩	🟩	⬛	⬛	1.0
C automatic login (reconnect)	C has made previuos successful manual login	C → PS	103	C_nameID	C_sessionKey				🟩	🟩	🟩	⬛	⬛	1.1
Successful result of C automatic login attempt.	#103.	PS → C	104					Note: Unsuccessful login result: #903	🟩	🟩	🟩	⬛	⬛	1.1
C logout - this device.		C → PS	105					PS destroys *C_sessionKey* for current session.	🟩	🟩	🟩	⬛	⬛	1.1
C logout - all devices.		C → PS	106					PS destroys C_sessionKey for all C's recorded sessions.	🟩	🟩	🟩	⬛	⬛	1.1
PS confrims logout.	#105, #106.	PS → C	107	Logout msg					🟩	🟩	🟩	⬛	⬛	1.1
H login									AC	WC	PS	H	G	
H login	H boot && settings: *remoteAccessEnable* == true	H → PS	120	H_hubID	H_pwd	H_alias			⬛	⬛	🟩	🟩	⬛	1.0
Successful result of H manual login.	#120	PS → H	121					Unsuccessful login result: #903	⬛	⬛	🟩	🟩	⬛	1.0
New remote access hub [Only here: WC = Promotional web site]									AC	WC	PS	H	G	
WC requests remote access credentials.		WC → PS	201	C_req_nameID	C_req_pwd			WC prompts user to select name and pwd.	⬛	🟥	🟥	⬛	⬛	?
PS returns new remote access credentials.	#201.	PS → WC	202	H_hubID	H_pwd	C_nameID		Remote access credentials for hub, and login crednetials for one admin user. Exception = #901.	⬛	🟥	🟥	⬛	⬛	?
Request all gadgets									AC	WC	PS	H	G	
C requests all gadgets		C → PS	301						⬛	🟩	🟩	⬛	⬛	1.0
PS forwards request of all gadgets to associated H.	~~#102, #104,~~ #301.	PS → H	302	C_sessionID					⬛	⬛	🟩	🟩	⬛	1.0
H returns all gadgets (that is present).	#302.	H → PS	303	C_sessionID	nbr_of_gadgets	[See below: A2: Gadget array format]		[Purple area] x [nbr_of_gadgets] *C_sessionID* = session ID of user requesting all gadgets (#302)	⬛	⬛	🟩	🟩	⬛	1.0
PS forwards all gadgets.	#303.	PS → C	304	nbr_of_gadgets	[See below: A2: Gadget array format]			[Purple area] x [nbr_of_gadgets]	🟩	🟩	🟩	⬛	⬛	1.0
Alter gadget state									AC	WC	PS	H	G	
C requests to alter a gadget's state.		C → PS	311	G_id	new_state				🟩	🟩	🟩	⬛	⬛	1.0
PS forwards request to alter a gadget's state.	#311.	PS → H	312	G_id	new_state				⬛	⬛	🟩	🟩	⬛	1.0
H requests gadget to alter state.	#312.	H → G	313	new_state	request_spec / "null"			request_spec: Request specific data from a gadget. Eg temp/hum.	⬛	⬛	⬛	🟩	🟩	1.0
Gadget reports its current (new) state.	#313, #341.	G → H	314	new_state					⬛	⬛	⬛	🟩	🟩	1.0
H reports a gadget state change.	#314, #341.	H → PS	315	G_id	new_state				⬛	⬛	🟩	🟩	⬛	1.0
PS notifies all C's of associated H that gadget state change has occured.	#315	PS → C	316	G_id	new_state				🟩	🟩	🟩	⬛	⬛	1.0
Poll gadgets									AC	WC	PS	H	G	
Poll gadget state	*pollDelaySec* interval has expired	H → G	341	request_spec / "null"				Gadget responds with #314. If state change detected: #315	⬛	⬛	⬛	🟩	🟩	1.1
Gadget presence reports									AC	WC	PS	H	G	
New gadget detected	H successfully poll new gadget, or gadget that was previously not responding.	H → PS	351	[See below: A1: Gadget format]				H notifies PS that an additional gadget should be passed to C.	⬛	⬛	🟩	🟩	⬛	1.1
PS forwards new gadget to C	#351	PS → C	352	[See below: A1: Gadget format]				C dynamically adds a new gadget to the view.	🟩	🟩	🟩	⬛	⬛	1.1
Gadget connection lost	H is not able to poll gadget that was previoulsy responding.	H → PS	353	G_id				H notifies PS that a gadget should be removed from C's list of gadgets.	⬛	⬛	🟩	🟩	⬛	1.1
PS forwards gadget removal request	#353	PS → C	354	G_id				C removes gadget from list.	🟩	🟩	🟩	⬛	⬛	1.1
Visual gadget grouping									AC	WC	PS	H	G	
C request list of gadget groups		C → PS	370						⬛	🟩	🟩	⬛	⬛	1.1
PS forwards gadget group request	#370	PS → H	371	C_sessionID					⬛	⬛	🟩	🟩	⬛	1.1
H sends list of gadget groups	#371	H → PS	372	C_sessionID	[See below: A4: Group array]				⬛	⬛	🟩	🟩	⬛	1.1
PS forwards list of gadget groups	#372	PS → C	373	[See below:A4: Group array]					⬛	🟩	🟩	⬛	⬛	1.1
Admin client features: Gadget setttings									AC	WC	PS	H	G	
WC requests to alter gadget alias.		WC → PS	401	G_ID	G_new_alias				⬛	🟩	🟩	⬛	⬛	1.1
PS forwards request to alter gadget alias.	#401.	PS → H	402	C_sessionID	G_ID	G_new_alias			⬛	⬛	🟩	🟩	⬛	1.1
H report gadget alias change.	#402.	H → PS	403	G_ID	G_new_alias				⬛	⬛	🟩	🟩	⬛	1.1
PS report gadget alias change.	#403.	PS → C	404	G_ID	G_new_alias			To all clients (not only admin).	🟩	🟩	🟩	⬛	⬛	1.1
WC requests to edit/create gadget group		WC → PS	410	[See below:A3: Gadget group]				groupName exists == group will be edited. Else: Group will be created.	⬛	🟥	🟥	🟥	⬛	?
Delete gadget group		WC → PS	411	groupName					⬛	🟥	🟥	🟥	⬛	?
Edit gadget poll dely									⬛	🟥	🟥	🟥	⬛	?
Admin client features: User setttings									AC	WC	PS	H	G	
Request all users	?								⬛	🟥	🟥	⬛	⬛	?
Add user	?								⬛	🟥	🟥	⬛	⬛	?
Delete user	?								⬛	🟥	🟥	⬛	⬛	?
AC specific									AC	WC	PS	H	G	
AC reports its geo location	Android device is not logged in to public server.	AC → PS	501	C_nameID	C_sessionKey	AC_longitude	AC_latitude	Used for AC background operation. Does not require previous login.	🟩	⬛	🟩	⬛	⬛	1.2
AC reports its geo location	Android device is logged in to public server.	AC → PS	502	AC_longitude	AC_latitude				🟩	⬛	🟩	⬛	⬛	1.2
PS forwards AC's geo location.	#501.	PS → H	503	C_nameID	AC_longitude	AC_latitude		H calculates home/away status	⬛	⬛	🟩	🟩	⬛	1.2
G Plug-and-Play: Gadget_Basic (Socket based)									AC	WC	PS	H	G	
AC verifies H address	AC connected to LAN	AC → H	601					Used in port scanning of local network to find and verify H.	🟩	⬛	⬛	🟩	⬛	1.2
H confirms its presence.	#601	H → AC	602	H_alias				AC now knows the hub address.	🟩	⬛	⬛	🟩	⬛	1.2
AC verifies connection to G	G running as WiFi AP. AC connected to G.	AC → G	611						🟩	⬛	⬛	⬛	🟩	1.2
G confirms connection.	#611	G → AC	612					AC knows its connected to a HoSo G AP.	🟩	⬛	⬛	⬛	🟩	1.2
AC requests G to blink.	#612	AC → G	613					Used by AC to confirm it's the correct physical WiFi module.	🟩	⬛	⬛	⬛	🟩	1.2
AC sends credentials for WLAN and local hub.		AC → G	614	LAN_SSID	LAN_pwd	H_IPv4	H_tcp_port	Credentials needed forG to connect to LAN and locate H.	🟩	⬛	⬛	⬛	🟩	1.2
G confirms credentials is received.	#614	G → AC	615					G may now connect to LAN. AC may disconnect from G AP.	🟨	⬛	⬛	⬛	🟩	1.2
G requests to add gadget(s) to H.	G connected to LAN.	G → H	620	G_MAC	G_tcp_port	nbr_of_gadgets	[See below: A5: Gadget array format]	G_MAC for hub to verify if it already has this gadget(s). Next for hub: #351.	⬛	⬛	⬛	🟩	🟩	1.2
H confirms that request #620 has been received.	#620	H → G	621					If G does not alredy exist in hub, it is added. G can now transiation to duty mode as Socket server within the LAN.	⬛	⬛	⬛	🟩	🟩	1.2
Global commands									AC	WC	PS	H	G	
Exception msg		PS → C	901	Exception message					🟩	🟩	🟩	⬛	⬛	1.0
Exception msg		H → PS	902	C_sessionID	Exception message			Can be used by H to target specific C. PS will create a #901 and send to target C.	⬛	⬛	🟥	🟥	⬛	?
Exception msg: Failed login	#101, #103, #120.	PS → C	903	Exception message					🟩	🟩	🟩	⬛	⬛	1.0
Exception msg: Hub disconnected		PS → C	904	Exception message					🟩	🟥	🟦	⬛	⬛	1.2
C pings PS		C → PS	ping					Sent periodically to keep connection alive.	⬛	🟩	🟩	🟩	⬛	1.1

ATTACHEMENTS		
A1: Gadget format	[G1_id]::[G1_alias]::[G1_type]::[G1_valueTemplate]::[G1_state]::[G1_pollDelaySec]	
A2: Gadget array format	[G1_id]::[G1_alias]::[G1_type]::[G1_valueTemplate]::[G1_state]::[G1_pollDelaySec]::[G2_id]::[G2_alias]::[G2_type]::[G2_valueTemplate]::[G2_state]::[G2_pollDelaySec]... G(n)	
A3: Gadget **group** format	[groupName]:[G_id]:[G_id]:[G_id]	
A4: Gadget **group** array format	[groupName]:[G_id]:[G_id]:[G_id]::[groupName]:[G_id]:[G_id]	**Note:** Groups are sperated by double colon [**::**] while internal group items are spearated by single colon [**:**]
A5: Gadget array format	[G1_alias]::[G1_type]::[G1_request_spec]::[G2_alias]::[G2_type]::[G2_request_spec]... G(n)	

VARIABLE EXPLANATIONS	
C_sessionKey	Used for automatic login to PS. Unique hash generated value returned to clients after successful manual login. Stored in client device memory (cache/cookies). Used for automatic (behind-the scenes) login henceforth, without forcing any client input. The sessionKey is used for automatic logins until the client manually logs out. The sessionKey is then destroyed in the client device -> Forcing manual login again.
C_sessionID	Unique client TCP connection ID. Initiated by PS as a refrence to the client session, which is also mapped to data about the associated client. Used for correct back-and forth routing of data between client device and PS. Allows the same user to be logged in on multiple devices simultanously and to be logged in on more than one insance on the same machine. Think of C_sessionID as an identifier for the user client issuing a request.
G_valueTemplate	Instance variable of gadgets. A String value that corresponds to a register used by frontend instances to know and specify how a gadget's state should be represented, in words and icons. The valueTemplate registers are implemented seperately in Android and web server. E.g. valuetTemplate record in Android : "light": {state 1="ON"} {state 0="OFF"} {iconON="aaa"} {iconOFF="bbb"}. E.g. valuetTemplate record in web server : "light": {state 1="ON"} {state 0="OFF"} {iconON="ddd"} {iconOFF="eee"}.
requst_spec	A way for H to request specific data from gadget units providing multiple services (associated with multiple gadget ID's). E.g. a sensor unit providing both temperature and humidity data, that should be mapped to two individual HomeSome gadget records (be represented and treated as two individual gadgets).