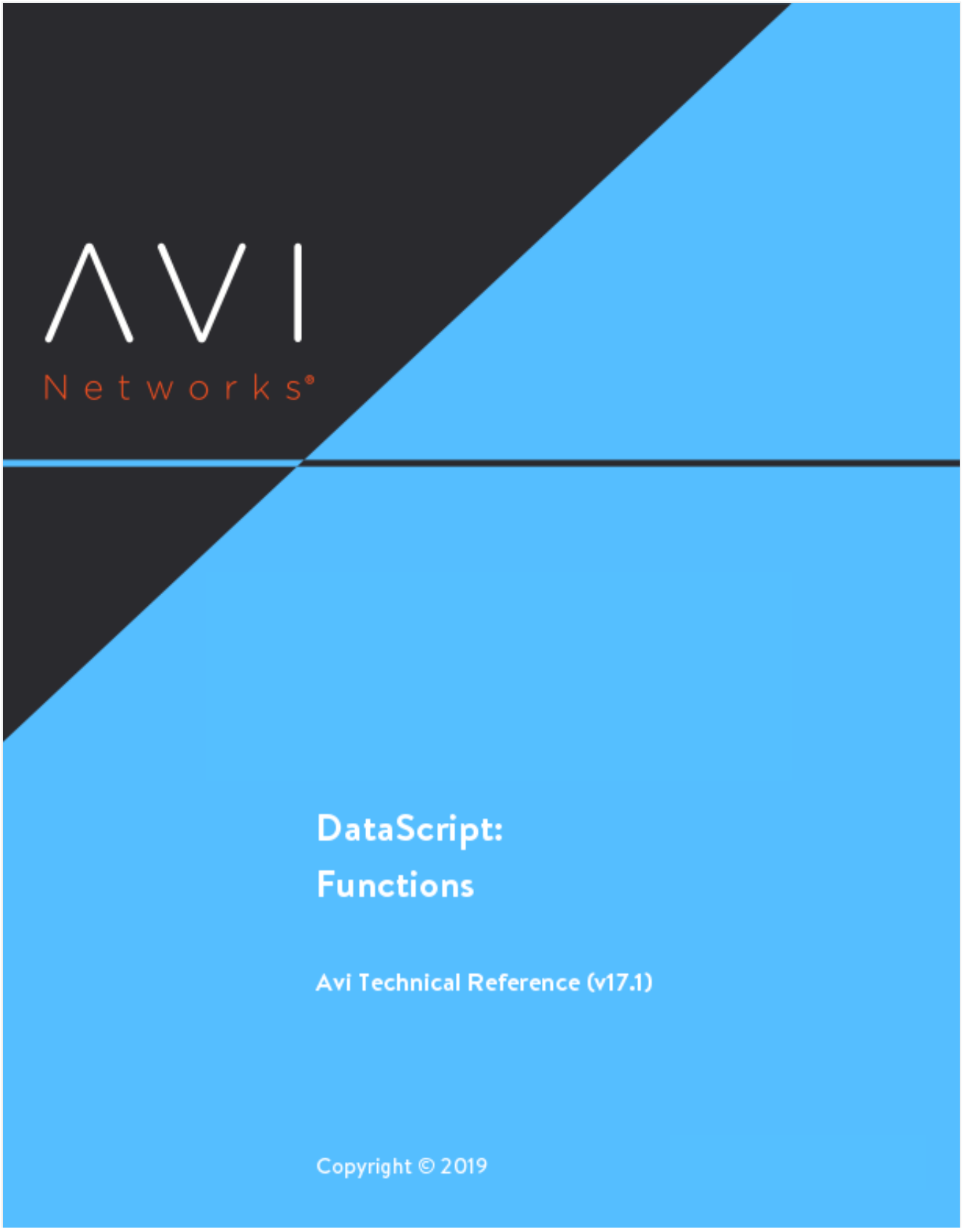

DataScript: Functions

Avi Technical Reference (v17.1)

DataScript: Functions

view online

DataScripts are comprised of any number of function or method calls which can be used to inspect and act on traffic flowing through a virtual service. DataScript's functions are exposed via Lua libraries and grouped into modules: *string, vs, http, pool, ssl and crypto.* Other Lua libraries may also be used, following the documentation from lua.org. The following functions are available:

String	Description
string.beginswith(source, target)	Search for string in beginning of a string
string.contains(source, target)	Search contains a string in another string
string.endswith(source, target)	Search for string at the end of a string
string.len(source)	Returns number of characters in string
string.lower(source)	Change a string to lower case
string.sub(source, begin, [end])	Extract a sub-string from a string
string.upper(source)	Change a string to upper case
VS	
avi.vs.client_ip( )	Returns the client IP address
avi.vs.client_port( )	Returns the client source port
avi.vs.ip( )	Returns IP address of the VS
avi.vs.log( )	Write a custom log to VS > client logs
avi.vs.name( )	Returns the name of the VS
avi.vs.port( )	Returns the VS port of the connection
avi.vs.rate_limit(type, string_to_limit, [defer_action=False])	Rate limits entities of various kinds.
avi.vs.reqvar.*	Set a global variable usable across events
avi.vs.table_insert([table_name,] key, value [, lifetime])	Store custom data in a time based table
avi.vs.table_lookup([table_name,] key [, lifetime_exten])	Lookup data in a table
avi.vs.table_refresh([table_name,] key [, lifetime_exten])	Update the expire time for a table entry
avi.vs.table_remove([table_name,] key)	Remove data from a table
HTTP	
avi.http.add_cookie(table)	Insert a new cookie
avi.http.add_header(name, value)	Insert a new header and value
avi.http.close_conn([reset])	Close or reset a TCP connection
avi.http.cookie_exists(name, [context])	Validate if a cookie already exists
avi.http.disable()	Upgrade (disable) HTTP processing for the current connection, which will subsequently be treated as layer 4 TCP
avi.http.get_cookie(name [, context])	Return the values of a cookie
avi.http.get_cookie_names ([context])	Return the names of cookies
avi.http.get_header([[name] [context]])	Return header names or their values
avi.http.get_host_tokens([start [, end]])	Return a subsection of the host

avi.http.get_path([false])	Returns the URI's path /path.index.htm
avi.http.get_path_tokens([start [, end]])	Return a subsection of the path
avi.http.get_query([arg_name | avi.QUERY_TABLE] [, decode])	Returns the URI's query ?a=1&b=2
avi.http.get_req_body([size_in_kb])	Returns part or all of the client request body.
avi.http.get_reqvar()	Gets (reads) data stored in a variable via the avi.http.set() function.
avi.http.get_uri([false])	Returns the URI (path plus query)
avi.http.get_userid( )	Returns the user ID for the session
avi.http.hostname( )	Return the hostname requested by client
avi.http.internal_status()	Returns HTTP status as a string
avi.http.method( )	Return the client's request method
avi.http.protocol( )	Returns the session protocol, http or https
avi.http.redirect(uri [,status])	Redirect a request
avi.http.remove_cookie(name1, [name2, ...])	Remove an existing cookie
avi.http.remove_header(name)	Remove all instances of a header
avi.http.replace_cookie(table)	Replace an existing cookies values
avi.http.replace_header(name, value)	Replace an existing header's value
avi.http.response(status, [headers, [body]])	Send a defined HTTP response page
avi.http.scheme( )	Returns http:// or https://
avi.http.secure( )	Returns on for https, nil for http
avi.http.set_path(new_uri)	Modify the path of a request
avi.http.set_query(integer | string | table)	Modify the query of a request
avi.http.set_reqvar()	Sets (write) arbitrary data from an HTTP request event into a variable. These variables have scope across the HTTP_REQ and HTTP_RESP events.
avi.http.set_uri(new_uri)	Change the URI
avi.http.set_userid( )	Sets the user ID for the session
avi.http.status( )	Returns status code to be sent to client

Pool

avi.pool.get_server_status(pool, server, port)	Returns the up/down status of the server
avi.pool.get_servers(pool)	Returns up and total server count
avi.pool.select(pool [, server [, port]])	Pick a specific pool
avi.pool.server_ip( )	Returns the IP of the selected server
avi.poolgroup.select(pool)	Pick a specific pool group

Groups

avi.ipgroup.contains(ipgroup, ip-address)	Compare an IP address against IPs within an IP group
avi.stringgroup.beginswith(stringgroup, string)	Compare a string against a list of strings within a string group
avi.stringgroup.contains(stringgroup, string)	Compare a string against a list of strings within a string group
avi.stringgroup.equals(stringgroup, string)	Compare a string against a list of strings within a string group

avi.stringgroup.endswith(stringgroup, string)	Compare a string against a list of strings within a string group

SSL

avi.ssl.cipher([true])	Return the SSL ciphers and settings
avi.ssl.client_cert([[avi.CLIENT_CERT] [, avi.CLIENT_CERT_FINGERPRINT] [, avi.CLIENT_CERT_SUBJECT] [, avi.CLIENT_CERT_ISSUER] [, avi.CLIENT_CERT_SERIAL]])	Returns the client's certificate, or part of it
avi.ssl.protocol( )	Return the SSL version
avi.ssl.server_name( )	Return SNI name field

Crypto

avi.crypto.decrypt(ciphertext, key [, iv [, algo]])	Decrypt content
avi.crypto.encrypt(plaintext, key [, iv [, algo]])	Encrypt content

Utilities

avi.utils.base64_decode(string)	Decode content
avi.utils.base64_encode(string)	Encode content
avi.utils.murmur_hash (string)	Hash content
avi.utils.sha1_hash (string)	
avi.utils.md5_hash (string)	

Note: Starting with Avi Vantage 17.1.10 and 17.2.3, DataScripts can hash arbitrary data using one of three functions supporting their corresponding named hashing methods: MurmurHash, SHA-1, and MD5.
For more information, refer to DataScript: avi.utils.murmur_hash (), avi.utils.sha1_hash (), avi.utils.md5_hash ().