



# DHCP Extension Dictionary

This appendix describes the DHCP extension dictionary entries and the application program interface (API) to the extension dictionary. It describes the data items available in the request and response dictionaries, and the calls to use when accessing dictionaries from Tcl extensions and shared libraries.

## Extension Dictionary Entries

A dictionary is a data structure that contains key-value pairs. There are two types of dictionaries: the attribute dictionaries the request and response dictionaries use, and the environment dictionary. This section describes the request and response dictionaries; the environment dictionary entries are described in the [“Tcl Environment Dictionary Methods”](#) section on page C-15.

## Decoded DHCP Packet Data Items

The decoded DHCP packet data items represent the information in the DHCP packet, and are available in both the request and response dictionaries. These dictionaries provide access to considerably more internal server data structures than just the decoded request and decoded response.

All of the options followed by an asterisk (\*) are multiple, which means that there may be more than one value associated with each option. In the DHCP/BOOTP packet, all of these data items appear in the same option. However, in the extension interface, these multiple data items are accessible through indexing.

You can access options that do not have names in [Table C-2 on page C-2](#) as option-*n*, where *n* is the option number. All fields are read/write. [Table C-1](#) describes the field values for the options.

**Table C-1** DHCP and BOOTP Fields

Name	Value
chaddr	blob (sequence of bytes)
ciaddr	IP address
file	string
flags	16-bit unsigned integer
giaddr	IP address
hlen	8-bit unsigned integer
hops	8-bit unsigned integer

**Table C-1 DHCP and BOOTP Fields (continued)**

Name	Value
htype	8-bit unsigned integer
op	8-bit unsigned integer
secs	16-bit unsigned integer
siaddr	IP address
sname	string
xid	32-bit unsigned integer
yiaddr	IP address

Table C-2 lists the DHCP and BOOTP options.

**Table C-2 DHCP and BOOTP Options**

Name (*=multivalued)	Number	Value
all-subnets-local	27	byte-valued boolean
arp-cache-timeout	35	int
boot-file	61	string
boot-size	13	16-bit unsigned integer
broadcast-address	28	IP address
cablelabs-client-configuration	122	blob (sequence of bytes)
		suboption:
ccc-primary-dhcp-server	1	IP address
ccc-secondary-dhcp-server	2	IP address
ccc-provisioning-server	3	blob (the first byte must be the type byte, with 0 for RFC 1035 encoding, and 1 for IP address encoding, for which the address must be in network order)
ccc-as-backoff-retry-blob	4	12-byte blob (3 unsigned 4-byte integers, which must be in network order); configures the Kerberos AS-REQ/AS-REP timeout, backoff, and retry mechanism
ccc-ap-backoff-retry-blob	5	12-byte blob (3 unsigned 4-byte integers, which must be in network order); configures the Kerberos AP-REQ/AP-REP timeout, backoff, and retry mechanism
ccc-kerberos-realm	6	variable-length blob (an RFC 1035 style name); a Kerberos realm name is required
ccc-use-tgt	7	1-byte unsigned integer boolean; indicates whether to use a Ticket Granting Ticket (TGT) when obtaining a service ticket for one of the application servers
ccc-provisioning-timer	8	1-byte unsigned integer; defines the maximum time allowed for the provisioning process to complete
ccc-ticket-control-mask	9	2-byte unsigned integer, in host order
ccc-kdc-addresses-blob	10	variable-length (multiple of 4) IP address, in network order

**Table C-2 DHCP and BOOTP Options (continued)**

<b>Name (*=multivalued)</b>	<b>Number</b>	<b>Value</b>
cisco-subnet-allocation	220	blob (structured)
cisco-vpn-id	221	blob (structured)
cookie-servers*	8	IP address
default-ip-ttl	23	8-bit unsigned int
default-tcp-ttl	37	8-bit unsigned int
dhcp-class-identifier	60	string
dhcp-client-identifier	61	blob (sequence of bytes)
dhcp-lease-time	51	int
dhcp-max-message-size	57	16-bit unsigned integer
dhcp-message	56	string
dhcp-message-type	53	blob (sequence of bytes)
dhcp-option-overload	52	blob (sequence of bytes)
dhcp-parameter-request-list*	55	8-bit unsigned integer
dhcp-parameter-request-list-blob	55	blob (sequence of bytes)
dhcp-rebinding-time	59	int
dhcp-renewal-time	58	int
dhcp-requested-address	50	IP address
dhcp-server-identifier	54	IP address
dhcp-user-class-id	77	string
domain-name	15	string
domain-name-servers*	6	IP address
extensions-path	18	string
finger-servers*	73	IP address
font-servers*	48	IP address
host-name	12	string
ieee802.3-encapsulation	36	8-bit unsigned integer
impress-servers*	10	IP address
interface-mtu	26	16-bit unsigned integer
ip-forwarding	19	byte-valued boolean
irc-servers*	74	IP address
log-servers*	7	IP address
lpr-servers*	9	IP address
mask-supplier	30	byte-valued boolean
max-dgram-reassembly	22	16-bit unsigned integer
merit-dump	14	string

Table C-2 DHCP and BOOTP Options (continued)

Name (*=multivalued)	Number	Value
mobile-ip-home-agents*	68	IP address
name-servers*	5	IP address
netbios-dd-servers*	45	IP address
netbios-name-servers*	44	IP address
netbios-node-type	46	blob (sequence of bytes)
netbios-scope	47	string
nis+-servers*	65	IP address
nis+domain	64	string
nis-domain	40	string
nis-servers*	41	IP address
nntp-servers*	71	IP address
non-local-source-routing	20	byte-valued boolean
ntp-servers*	42	IP address
path-mtu-aging-timeout	24	int
path-mtu-patheau-tables*	25	16-bit unsigned integer
perform-mask-discovery	29	byte-valued boolean
policy-filters*	21	IP address (there can be two policy filters, each one having its own IP address)
pop3-servers*	70	IP address
relay-agent-info	82	blob (sequence of bytes)
		suboption:
relay-agent-circuit-id	1	(deprecated in favor of <i>relay-agent-circuit-id-data</i> )
relay-agent-circuit-id-data	1	blob (does not require the suboption number as the first byte); accesses and manipulates the <i>relay-agent-circuit-id</i> data from a DHCP request or response
relay-agent-remote-id	2	(deprecated in favor of <i>relay-agent-remote-id-data</i> )
relay-agent-remote-id-data	2	blob (does not require the suboption number as the first byte); accesses and manipulates the <i>relay-agent-remote-id</i> data from a DHCP request or response
relay-agent-device-class	4	(deprecated in favor of <i>relay-agent-device-class-data</i> )
relay-agent-device-class-data	4	4-byte unsigned integer; represents the device class, or individual attributes of the cable modem (RFC 3256)
relay-agent-subnet-selection-data	5, 150	IP address
relay-agent-vpn-id-data	181, 151	string
relay-agent-server-id-override-data	182, 152	IP address
resource-location-servers*	11	IP address
root-path	17	string

**Table C-2 DHCP and BOOTP Options (continued)**

Name (*=multivalued)	Number	Value
router-discovery	31	byte-valued boolean
router-solicitation-address	32	IP address
routers*	3	IP address
smtp-servers*	69	IP address
static-routes*	33	IP address
streettalk-directory-assistance-servers*	76	IP address
streettalk-servers*	75	IP address
subnet-mask	1	IP address
subnet-selection	118	IP address
swap-server	16	IP address
tcp-keepalive-internal	38	int
tcp-keepalive-garbage	39	byte-valued boolean
tftp-server	66	string
time-offset	2	int
time-servers*	4	IP address
trailer-encapsulation	34	byte-valued boolean
vendor-encapsulated-options	43	blob (sequence of bytes)
vpn-id	185	blob (structured)
www-servers*	72	IP address
x-display-managers*	49	IP address

Table C-3 lists the decoded packet fields.

**Table C-3 Decoded Packet Field**

Data Item	Value	Description
dump-packet	int	When the value of dump-packet is set to 1, Network Registrar dumps the current decoded DHCP/BOOTP packet to the log file. An extension can put the value 1 into this data item at multiple points in its execution. This may be useful when debugging extensions.
mac-address	blob	MAC address that came in the client packet. The first byte is the hardware type, the second is the hardware length, and the remaining (up to 16) is the information from the chaddr read just after <b>post-packet-decode</b> . This is a useful aggregation of the <i>htype</i> , <i>hlen</i> , and <i>chaddr</i> fields of the DHCP packet. When read it is constructed from these fields; when written it is placed into these fields.

## Request Dictionary

Table C-4 lists the data items that you can set in the request dictionary at any time. The DHCP server reads them at various times. Unless indicated otherwise, all operations are read/write.

**Table C-4 Request Dictionary Specific Data Items**

Data Item	Value
<i>allow-bootp</i>	int
If set to 1, allows BOOTP for any scope for this request. Read during scope selection and while checking for lease acceptability.	
<i>allow-dhcp</i>	int
If set to a 1, allows DHCP for any scope for this request. Read during scope selection and while checking for lease acceptability.	
<i>allow-dynamic-bootp</i>	int
If set to a 1, allows dynamic BOOTP for any scope for this request. Read during scope selection and while checking for lease acceptability.	
<i>bootp-reply-options</i>	blob
Overrides any <i>bootp-reply-options</i> in any policy. Read when gathering data for the output packet.	
<i>client-class-name</i>	string
Name of the client-class used to complete the client information (if any).	
<i>client-class-policy</i>	string
Name of the policy that is associated with the client-class. If set, it must be with the name of a policy that was already configured in the server.	
<i>client-domain-name</i>	string
Domain name that the client wants to use. It may not exist, in which case the DHCP server uses the domain name specified in the scope. Read when queuing the request for DNS update just prior to the update of stable storage.	
<i>client-host-name</i>	string
Host name for the client in DNS; read when queuing in the request for a DNS update just before updating stable storage. Places the actual name in DNS when that operation completes.	
<i>client-id</i>	blob
Client identification that the server uses to track the client. May be the <i>client-id</i> sent with a request or internally generated from the MAC address. See <i>client-id-created-from-mac-address</i> .	
<i>client-id-created-from-mac-address</i>	int
If set to 1, the <i>client-id</i> must be created for internal use from the client-supplied MAC address and should not be used in reporting.	
<i>client-ipaddress</i>	IP address
IP address from which the client sent its packet. Note that it could be zero if the client does not yet have an IP address.	
<i>client-lookup-id</i>	blob
Client lookup ID calculated by the <i>client-lookup-id</i> expression of the client-class.	

**Table C-4 Request Dictionary Specific Data Items (continued)**

<b>Data Item</b>	<b>Value</b>
<i>client-mac-address</i>	blob
MAC address stored in the client object associated with the request dictionary. Has the same format (and was created from) the <i>mac-address</i> described in <a href="#">Table C-3 on page C-5</a> .	
<i>client-os-type</i>	int
Change the client entry of the request packet by setting this at the <b>pre-client-lookup</b> or <b>post-client-lookup</b> extension points. Can also be read at <b>check-lease-acceptable</b> , but cannot be set there. To set the value, you must first set the <i>os-type</i> in the <b>post-packet-decode</b> request dictionary.	
<i>client-policy</i>	string
Name of the policy that is associated with the client entry. If set, must be the name of a preconfigured policy in the DHCP server.	
<i>client-port</i>	int
Port from which the client sent its request.	
<i>client-requested-host-name</i>	string
Host name that the client requested be used for the DNS update. The DHCP server saves this information so that a change can be detected.	
<i>client-wants-nulls-in-strings</i>	int
Determines whether the DHCP server returns strings to the client terminated with a null. If set to 1, the server terminates strings with a null. If set to 0, it does not terminate strings with a null. Set before <b>post-packet-decode</b> and read when encoding the response packet after <b>pre-packet-encode</b> .	
<i>dhcp-reply-options</i>	blob
Overrides any DHCP reply options specified in any policy. Read when gathering data for the output packet.	
<i>import-packet</i>	int
Determines whether the server treats the packet as if it came from an import client. If set to 1, the server treats the client as an import client and performs all DNS operations on it before sending an ACK. Read when checking the server import mode (right after <b>post-packet-decode</b> ), getting ready for DNS processing, and when setting the reply address.	
<i>limitation-count</i>	int
Number of simultaneous users allowed with the same <i>limitation-id</i> .	
<i>limitation-id</i>	blob
Calculated by the <i>limitation-id</i> expression (if any) for the client-class in which this request falls.	
<i>limitation-id-null</i>	int
Set to 1(true) if the <i>limitation-id</i> is null, 0 (false) if another value.	
<i>log-client-criteria-processing</i>	int
If set to a 1, logs the criteria processing for the client for this request. Read when trying to acquire a new lease for a client that does not have one, and when checking for lease acceptability.	
<i>log-client-detail</i>	int
If set to a 1, logs the client-class processing for this request. Read at the end of client-class processing, after <b>post-client-lookup</b> .	

**Table C-4 Request Dictionary Specific Data Items (continued)**

<b>Data Item</b>	<b>Value</b>
<i>log-dns-update-detail</i>	int
If set to a 1, logs DNS update details for this request.	
<i>log-failover-detail</i>	int
If set to a 1, logs a more detailed level of failover activity, such as all failover state changes.	
<i>log-incoming-packet-detail</i>	int
If set to a 1, checks whether detailed incoming packet tracing occurred for this request, so that you do not need to put a separate trace on it. Read before packet decoding and the first extension point.	
<i>log-incoming-packets</i>	int
If set to a 1, logs the incoming packets for this request. Read after <b>post-decode-packet</b> .	
<i>log-ldap-create-detail</i>	int
If set to a 1, logs messages whenever the DHCP server initiates a lease state entry creation to, receives a response from, or retrieves a result or error message from an LDAP server.	
<i>log-ldap-query-detail</i>	int
If set to a 1, logs messages whenever the DHCP server initiates a query to, receives a response from, or retrieves a query result or an error message from an LDAP server.	
<i>log-ldap-update-detail</i>	int
If set to a 1, logs messages whenever the DHCP server initiates an update lease state to, receives a response from, or a retrieves a result or error message from an LDAP server.	
<i>log-leasequery</i>	int
If set to a 1, logs messages when leasequery packets are processed without internal errors and result in an ACK or a NAK.	
<i>log-incoming-packets</i>	int
If set to a 1, logs incoming packets for this request. Read after <b>post-decode-packet</b> .	
<i>log-missing-options</i>	int
If set to a 1, logs missing options (those a client requests but the DHCP server cannot return). Read while gathering data for the response.	
<i>log-outgoing-packet-detail</i>	int
If set to a 1, logs a detailed dump of the outgoing packet for this request. Read after <b>pre-packet-encode</b> and just before sending the packet to the DHCP client.	
<i>log-unknown-criteria</i>	int
If set to a 1, logs any unknown criteria specified in the client's inclusion or exclusion criteria for this request. Read when acquiring a new client lease or checking lease acceptability for an existing client.	
<i>namespace-description</i>	string (read-only)
Description for the namespace. See <i>namespace-name</i> for details.	
<i>namespace-id</i>	int (read-only)
Namespace identifier. See <i>namespace-name</i> for details.	

**Table C-4 Request Dictionary Specific Data Items (continued)**

Data Item	Value
<i>namespace-name</i>	string (read-only)
	Name of the namespace. The request dictionary does not have valid values for these items at <b>post-packet-decode</b> , but does at all other extension points, because the namespace has not yet been determined. This is so that a script can change the <i>vpn-id</i> option or suboption at <b>post-packet-decode</b> and thereby affect the namespace used for a lease.
<i>namespace-vrf-name</i>	string (read-only)
	Virtual routing and forwarding table identifier for the namespace. See <i>namespace-name</i> for details.
<i>namespace-vpn-id</i>	blob, typically 7 bytes (read-only)
	Virtual private network identifier for the namespace. See <i>namespace-name</i> for details.
<i>ping-clients</i>	int
	If set to a 1, performs a ping before offering a lease for this request. Read just before determining if a lease is acceptable for a client.
<i>reply-to-client-address</i>	int
	If set to 1, the server sends the response packet to the <i>client-ip-address</i> and the <i>client-port</i> instead of using the RFC-mandated algorithm.
<i>selection-criteria</i>	string
	Comma-separated string that contains the scope's selection criteria.
<i>selection-criteria-excluded</i>	string
	Comma-separated string that contains the scope's exclusion criteria.
<i>send-ack-first</i>	int
	If set to a 1, updates DNS after the ACK for DHCP requests. Read just before initiating the DNS operation.
<i>transaction-time</i>	int
	Time, in seconds since 1970, that the input packet was decoded.
<i>update-dns</i>	string
	Requests partial, full, or no dynamic DNS updates on a per-request packet basis. Input and output values are 1=update-all, 2=update-fwd-only, 3=update-rev-only, and 0=update-none.
<i>update-dns-for-bootp</i>	int
	If set to a 1, updates DNS for BOOTP requests for this request. Read just before initializing the DNS operation for BOOTP.
<i>verbose-logging</i>	int
	If set to a 1, logs verbose messages for this request. Read at various times during processing.

## Response Dictionary

Table C-5 lists the data items you can set in the response dictionary at any time. The DHCP server reads them at various times. Unless indicated otherwise, the operation is read/write.

**Table C-5 Response Dictionary Specific Data Items**

Data Item	Value
<i>auto-configure</i>	int
Network Registrar can ask and be notified if autoconfiguration should be disabled on the local subnet. Writing an extension script to return <i>yiaddr</i> =0.0.0.0 and setting this option (0xFB) to 0 prevents a Windows 2000 RC3 DHCP client from autoconfiguring. Allows clients to choose a link-local IP address so that they can communicate with other hosts on the same link.	
<i>client-domain-name</i>	string (read-only)
From the client information in the lease, the domain name that the client wants to use. It might not exist, in which case the DHCP server uses the domain name specified in the scope. Read when queuing the request for DNS update just prior to the update of stable storage.	
<i>client-expiration-time</i>	int
Absolute time value of the lease expiration time last given to the client.	
<i>client-host-name</i>	string
From the client information in the lease, the host name that the DHCP server puts into DNS. Read when queuing the request for a DNS update just before updating stable storage.	
<i>client-id</i>	blob
From the client information in the lease, the client identification that the server used to keep track of the client. This might be the <i>client-id</i> sent with a request or a <i>client-id</i> internally generated from the MAC address.	
<i>client-id-created-from-mac-address</i>	int (read-only)
From the client information in the lease. If set to 1, the <i>client-id</i> must be created from the MAC address and should not be used in reporting.	
<i>client-last-transaction-time</i>	int (read-only)
Time, in seconds, since 1970, that the DHCP server last heard from this client.	
<i>client-mac-address</i>	blob
From the client information in the lease, the MAC address stored in the client object associated with the request dictionary. Has the same format as (and was created from) the MAC address.	
<i>client-os-type</i>	int
Change the client entry of the request packet by setting this at the <b>pre-client-lookup</b> or <b>post-client-lookup</b> extension points. Can also be read at <b>check-lease-acceptable</b> , but cannot be set there. To set the value, you must first set the <i>os-type</i> in the <b>post-packet-decode</b> request dictionary.	
<i>client-limitation-id</i>	blob (read-only)
Limitation identifier of the client associated with the current lease.	
<i>client-requested-host-name</i>	string
From the client information in the lease, the host name that the client requested for the DNS update.	
<i>domain-name-changed</i>	int
If set to 1, the domain name in the current packet differs from the domain name used in the DNS update. Read after <b>check-lease-acceptable</b> and before <b>pre-packet-encode</b> .	
<i>host-name-changed</i>	int
If set to 1, the host name in the current packet differs from that used in the DNS update. Read after <b>check-lease-acceptable</b> and before <b>pre-packet-encode</b> .	

**Table C-5 Response Dictionary Specific Data Items (continued)**

<b>Data Item</b>	<b>Value</b>
<i>host-name-in-dns</i>	int
If set to 1, the host name is in DNS. Read after <b>check-lease-acceptable</b> and before <b>pre-packet-encode</b> . Written after the host name goes into DNS.	
<i>lease-deactivated</i>	int (read-only)
If set to 1, reports that the lease is de-activated.	
<i>lease-ipaddress</i>	IP address (read-only)
IP address of the lease that the DHCP server uses in processing.	
<i>lease-namespace-description</i>	string (read-only)
Description for the namespace stored with a response's lease.	
<i>lease-namespace-id</i>	int (read-only)
Identifier for the namespace stored with a response's lease.	
<i>lease-namespace-name</i>	string (read-only)
Name of the namespace stored with a response's lease.	
<i>lease-namespace-vrf-name</i>	string (read-only)
Virtual routing and forwarding table identifier for the namespace stored with a response's lease.	
<i>lease-namespace-vpn-id</i>	blob, typically 7 bytes (read-only)
Virtual private network (VPN) identifier for the namespace stored with a response's lease.	
<i>lease-relay-agent-circuit-id</i>	blob
Accesses and manipulates the relay agent circuit id data as stored with a response's lease. Requires the suboption number 1 as the first byte. Deprecated in favor of the <i>lease-relay-agent-circuit-id-data</i> item.	
<i>lease-relay-agent-circuit-id-data</i>	blob (use instead of deprecated <i>lease-relay-agent-circuit-id</i> )
Accesses and manipulates the <i>relay-agent-circuit-id-data</i> as stored with a response's lease. Relevant only if the <i>save-relay-agent-data</i> DHCP server attribute is enabled.	
<i>lease-relay-agent-remote-id</i>	blob
Accesses and manipulates the <i>relay-agent-remote-id</i> data as stored with a response's lease. Requires suboption number 2 as the first byte. Deprecated in favor of the <i>lease-relay-agent-remote-id-data</i> item.	
<i>lease-relay-agent-remote-id-data</i>	blob (use instead of <i>lease-relay-agent-remote-id</i> item)
Accesses and manipulates the <i>relay-agent-remote-id-data</i> as stored with a response's lease. Relevant only if the <i>save-relay-agent-data</i> DHCP server attribute is enabled.	
<i>lease-relay-agent-server-id-override-data</i>	IP address
Accesses and manipulates the <i>relay-agent-server-id-override-data</i> as stored with a response's lease. Relevant only if the <i>save-relay-agent-data</i> DHCP server attribute is enabled.	
<i>lease-relay-agent-subnet-selection-data</i>	IP address
Accesses and manipulates the <i>relay-agent-subnet-selection-data</i> as stored with a response's lease. Relevant only if the <i>save-relay-agent-data</i> DHCP server attribute is enabled.	
<i>lease-relay-agent-vpn-id-data</i>	blob

**Table C-5 Response Dictionary Specific Data Items (continued)**

Data Item	Value
	Accesses and manipulates the <i>relay-agent-vpn-id</i> data as stored with a response's lease. Relevant only if the <i>save-relay-agent-data</i> DHCP server attribute is enabled.
<i>lease-reserved</i>	int (read-only)
	If set to 1, reports if the lease is reserved.
<i>lease-start-time-of-state</i>	int (read-only)
	Time, in seconds since 1970, that this lease was first placed into its current state.
<i>lease-state</i>	string (read-only)
	State of the lease, which can be <i>available</i> , <i>offered</i> , <i>expired</i> , <i>leased</i> , or <i>unavailable</i> .
<i>mac-address</i>	blob
	If set to 1, the scope allows BOOTP. Written after a DNS operation completes.
<i>namespace-description</i>	string (read-only)
	Description for the namespace.
<i>namespace-id</i>	int (read-only)
	Namespace identifier.
<i>namespace-name</i>	string (read-only)
	Name of the namespace.
<i>namespace-vrf-name</i>	string (read-only)
	Virtual routing and forwarding table (VRF) identifier for the namespace.
<i>namespace-vpn-id</i>	blob, typically 7 bytes (read-only)
	Virtual private network (VPN) identifier for the namespace.
<i>ping-clients</i>	int
	If set to 1, performs a ping before offering a lease for this request. Read just before determining a client's lease acceptability.
<i>reply-ipaddress</i>	IP address
	IP address to use when replying to the DHCP client. Read just after <b>pre-packet-encode</b> . If you change its value in a <b>pre-packet-encode</b> , the IP address you place in it should be for a system that can respond to ARP queries (unless it is a broadcast address). Even if unicast is enabled and the broadcast flag is not set in the DHCP request, the local ARP cache is not set with a mapping from a new <i>reply-ipaddress</i> in the <b>pre-packet-encode</b> to the MAC address in the DHCP request.
<i>reply-port</i>	int
	Port to use when replying to the DHCP client. Read just after <b>pre-packet encode</b> .
<i>reverse-name-in-dns</i>	int
	If equal to 1, the reverse name is in DNS. Read before initializing a DNS operation.
<i>scope-allow-bootp</i>	int
	If set to 1, the scope allows BOOTP. Written after a DNS operation completes.
<i>scope-allow-dhcp</i>	int (read-only)
	If set to 1, the scope allows DHCP.

**Table C-5 Response Dictionary Specific Data Items (continued)**

<b>Data Item</b>	<b>Value</b>
<i>scope-allow-dynamic-bootp</i>	int (read-only) If set to 1, the scope allows dynamic BOOTP.
<i>scope-available-leases</i>	int (read-only) Number of available leases on the current scope.
<i>scope-deactivated</i>	int (read-only) If set to 1, the scope is de-activated.
<i>scope-dns-forward-server-address</i>	IP address (read-only) DNS server to use for the DNS forward address.
<i>scope-dns-forward-zone-name</i>	string (read-only) Forward zone name configured in the scope.
<i>scope-dns-number-of-host-bytes</i>	int (read-only) Number of host bytes used by the DHCP server code that handles DNS updates.
<i>scope-dns-reverse-server-address</i>	IP address (read-only) DNS server to use for the DNS reverse address.
<i>scope-dns-reverse-zone-name</i>	string (read-only) Reverse zone name configured in the scope.
<i>scope-name</i>	string (read-only) Name of the scope that contains the lease that the DHCP server is processing.
<i>scope-network-number</i>	IP address (read-only) Network number of the scope that contains the lease the DHCP server is processing.
<i>scope-ping-clients</i>	boolean (read-only) If set to 1, the scope associated with the current lease was configured to support a ping operation prior to offering a lease.
<i>scope-primary-network-number</i>	IP address (read-only) Network number of this scope's primary scope.
<i>scope-primary-subnet-mask</i>	IP address (read-only) Subnet mask of this scope's primary scope.
<i>scope-renew-only</i>	int (read-only) If set to 1, the scope is renew-only.
<i>scope-renew-only-expire-time</i>	int (read-only) Absolute time, in seconds since January 1, 1970, at which a renew-only scope should cease to be renew-only.
<i>scope-selection-tags</i>	string (read-only) Comma-separated string that contains the scope's selection criteria.
<i>scope-send-ack-first</i>	int (read-only) If set to 1, the scope sends an ACK before performing the rest of the processing.
<i>scope-subnet-mask</i>	IP address (read-only)

**Table C-5 Response Dictionary Specific Data Items (continued)**

Data Item	Value
	Subnet mask of the scope that contains the lease the DHCP server is processing.
<i>scope-update-dns</i>	string (read-only)
	DNS updates for forward or reverse zones. Output values are 1=update-all, 2=update-fwd-only, 3=update-rev-only, and 0=update-none. Introduced in Network Registrar 6.1.1.
<i>scope-update-dns-enabled</i>	boolean (read-only)
	If set to 1, the scope has update DNS enabled for forward and reverse zones. Deprecated in favor of <i>scope-update-dns</i> .
<i>scope-update-dns-for-bootp</i>	int (read-only)
	If set to 1, the scope has update DNS enabled for BOOTP.
<i>transaction-time</i>	int (read-only)
	Time, in seconds since 1970, that the request was decoded.

## Extension Dictionary API

This section contains the dictionary method calls to use when accessing dictionaries from Tcl extensions and shared libraries.

### Tcl Attribute Dictionary API

In an attribute dictionary, the keys are constrained to be the names of attributes as defined in the Network Registrar DHCP server configuration. The values are the string representation of the legal values for that particular attribute. For example, IP addresses are specified by the dotted-decimal string representation of the address, and enumerated values are specified by the name of the enumeration. This means that numbers are specified by the string representation of the number.

Attribute dictionaries are unusual in that they can contain more than one instance of a key. These instances are ordered, with the first instance at index zero. Some of the attribute dictionary methods allow an index to indicate a particular instance or position in the list of instances to be referenced.

For the Tcl API routine signature and command syntax, see the [“Tcl Attribute Dictionary API” section on page C-14](#).

### Tcl Request and Response Dictionary Methods

Attribute dictionaries use commands with which you can change and access the values in the dictionaries. [Table C-6](#) lists the commands to use with the request and response dictionaries. In this case, you can define the *dict* variable as **request** or **response**.

See the *install-path/examples/dhcp/tcl/tclextension.tcl* file for examples.

**Table C-6** *Tcl Request and Response Dictionary Methods*

Method	Syntax
<b>get</b>	<code>\$dict get attribute [index [bMore]]</code>
	Returns the value of the attribute from the dictionary, represented as a string. If the dictionary does not contain the attribute, the empty string is returned instead. If you include the index value, this returns the <i>indexth</i> instance of the attribute. Some attributes can appear more than once in the request or response packet. The index selects which instance to return.  If you include the <i>bMore</i> , the <b>get</b> method sets <i>bMore</i> to TRUE if there are more attributes after the one returned, otherwise to FALSE. Use this to determine whether to make another call to <b>get</b> to retrieve other instances of the attribute.
<b>log</b>	<code>\$dict log level message ...</code>
	Outputs a message into the DHCP server's logging system. The level should be LOG_ERROR, LOG_WARNING, or LOG_INFO. The remaining arguments are concatenated and sent to the logging system at the specified level.
<b>remove</b>	<code>\$dict remove attribute [index]</code>
	Removes the attribute from the dictionary. If you omit the index or set it to the special value REMOVE_ALL, this removes any existing instances of the attribute. If you include the index as a number, this removes the instance of the attribute at the position indicated. This method always returns 1, even if the dictionary does not contain that attribute at that index.
<b>put</b>	<code>\$dict put attribute value [index]</code>
	Associates a value with the attribute in the dictionary. If you omit the index or set it to the special value REPLACE, this replaces any existing instances of the attribute with the single value. If you include the index value as the special value APPEND, this appends a new instance of the attribute to the end of the list of instances of the attribute. If you include the index value as a number, this inserts a new instance of the attribute at the position indicated. If you set the index value to the special value AUGMENT, this puts the attribute only if there is not one already.
<b>trace</b>	<code>\$dict trace level message ...</code>
	Returns a message in the DHCP server's packet tracing system. At level 0, no tracing occurs. At level 1, it traces only that the server received the packet and sent a reply. At level 4, it traces everything. The remaining arguments are concatenated and sent to the tracing system at the specified level. The default tracing is set using the DHCP server <i>extension-trace-level</i> attribute.

## Tcl Environment Dictionary Methods

**Table C-7** describes the commands to use with the environment dictionary. In this case, you can define the *dict* variable as **environ**, as in the following procedure example:

```
proc tclhelloworld2 { request response environ } {
    $environ put trace-level 4
    $environ log LOG_INFO "Environment hello world"
}
```

**Table C-7** *Tcl Environment Dictionary Methods*

Method	Syntax
<b>clear</b>	<code>\$dict clear</code>
	Removes all entries from the dictionary.

**Table C-7 Tcl Environment Dictionary Methods (continued)**

Method	Syntax
<b>containsKey</b>	<i>\$dict containsKey key</i>
	Returns 1 if the dictionary contains the key, otherwise 0.
<b>firstKey</b>	<i>\$dict firstKey</i>
	Returns the name of the first key in the dictionary. Note that the keys are not stored sorted by name. If a key does not exist, returns the empty string.
<b>get</b>	<i>\$dict get key</i>
	Returns the value of the key from the dictionary. If a key does not exist, returns the empty string.
<b>isEmpty</b>	<i>\$dict isEmpty</i>
	Returns 1 if the dictionary has no entries, otherwise 0.
<b>log</b>	<i>\$dict log level message ...</i>
	Returns a message in the DHCP server's logging system. The <i>level</i> should be one of LOG_ERROR, LOG_WARNING, or LOG_INFO. The remaining arguments are concatenated and sent to the logging system at the specified level.
<b>nextKey</b>	<i>\$dict nextKey</i>
	Returns the name of the next key in the dictionary that follows the key returned in the last call to <b>firstKey</b> or <b>nextKey</b> . If a key does not exist, returns the empty string.
<b>put</b>	<i>\$dict put key value</i>
	Associates a value with the key, replacing an existing instance of the key with the new value.
<b>remove</b>	<i>\$dict remove key</i>
	Removes the key from the dictionary. Always returns 1, even if the dictionary did not contain the key.
<b>size</b>	<i>\$dict size</i>
	Returns the number of entries in the dictionary.
<b>trace</b>	<i>\$dict trace level message ...</i>
	Returns a message in the DHCP server's packet tracing system. At level 0, no tracing occurs. At level 1, it traces only that the server received the packet and sent a reply. At level 4, it traces everything. The remaining arguments are concatenated and sent to the tracing system at the specified level. The default tracing is set using the DHCP server <i>extension-trace-level</i> attribute.

## DEX Attribute Dictionary API

When writing DEX extensions for C/C++, you can specify keys as the attribute name's string representation or by type (a byte sequence defining the attribute). This means that some of these access methods have four different variations that are the combinations of string or type for the key or value.

A basic DEX extension example might be:

```
int DEXAPI dexhelloworld( int iExtensionPoint,
    dex_AttributeDictionary_t* pRequest,
    dex_AttributeDictionary_t* pResponse,
    dex_EnvironmentDictionary_t* pEnviron )
{
    pEnviron->log( pEnviron, DEX_LOG_INFO, "hello world" );
    return DEX_OK;
}
```

See the `install-path/examples/dhcp/dex/dextension.c` file or other files in that directory for examples.

## DEX Request and Response Dictionary Methods

DEX attribute dictionaries use active commands, called methods, with which you can change and access values. Table C-8 lists the methods to use with the request and response dictionaries. In this case, you can define the `pDict` variable as **pRequest** or **pResponse**, as in:

```
pRequest->get( pRequest, "host-name", 0, 0 );
```

**Table C-8 DEX Request and Response Dictionary Methods**

Method	Syntax
<b>allocateMemory</b>	<b>void* pDict-&gt;allocateMemory( dex_AttributeDictionary_t* pDict, unsigned int iSize )</b>
	Allocates memory in extensions that persists only for the lifetime of this request.
<b>get</b>	<b>const char* pDict-&gt;get( dex_AttributeDictionary_t* pDict, const char* pszAttribute, int iIndex, abool_t* pbMore )</b>
	Returns the value of the <i>iIndexed</i> instance of the attribute from the dictionary, represented as a string. If the dictionary does not contain the attribute (or that many instances of it), the empty string is returned instead. If <i>pbMore</i> is nonzero, the <b>get</b> method sets <i>pbMore</i> to TRUE if there are more instances of the attribute after the one returned, otherwise to FALSE. Use this to determine whether to make another call to <b>get</b> to retrieve other instances of the attribute.
<b>getBytes</b>	<b>const abytes_t* pDict-&gt;getBytes( dex_AttributeDictionary_t* pDict, const char* pszAttribute, int iIndex, abool_t* pbMore )</b>
	Returns the value of the <i>iIndexed</i> instance of the attribute from the dictionary, as a sequence of bytes. If the dictionary does not contain the attribute (or that many instances of it), returns 0 instead. If <i>pbMore</i> is nonzero, the <b>getBytes</b> method sets it to TRUE if there are more instances of the attribute after the one returned, otherwise to FALSE. Use this to determine whether to make another call to <b>getBytes</b> to retrieve other instances of the attribute.
<b>getByType</b>	<b>const char* pDict-&gt;getByType( dex_AttributeDictionary_t* pDict, const abytes_t* pszAttribute, int iIndex, abool_t* pbMore )</b>
	Returns the value of the <i>iIndexed</i> instance of the attribute from the dictionary, represented as a string. If the dictionary does not contain the attribute (or that many instances of it), returns the empty string instead. If <i>pbMore</i> is nonzero, the <b>getByType</b> method sets <i>pbMore</i> to TRUE if there are more instances of the attribute after the one returned, otherwise to FALSE. Use this to determine whether to make another call to <b>getByType</b> to retrieve other instances.
<b>getBytesByType</b>	<b>const abytes_t* pDict-&gt;getBytesByType( dex_AttributeDictionary_t* pDict, const abytes_t* pszAttribute, int iIndex, abool_t* pbMore )</b>
	Returns the value of the <i>iIndexed</i> instance of the attribute from the dictionary, as a sequence of bytes. If the dictionary does not contain the attribute (or that many instances of it), 0 is returned instead. If <i>pbMore</i> is nonzero, sets the variable pointed to TRUE if there are more instances of the attribute after the one returned, otherwise to FALSE. Use this to determine whether to make another call to <b>get</b> to retrieve other instances of the attribute.

Table C-8 DEX Request and Response Dictionary Methods (continued)

Method	Syntax
<b>getType</b>	<b>const abytes_t* pDict-&gt;getType( dex_AttributeDictionary_t* pDict, const char* pszAttribute )</b>
	Returns a pointer to the byte sequence defining the attribute, if the attribute name matches a configured attribute, otherwise 0.
<b>log</b>	<b>abool_t pDict-&gt;log( dex_AttributeDictionary_t* pDict, int eLevel, const char* pszFormat, ... )</b>
	Returns a message in the DHCP server's logging system. The <i>eLevel</i> should be one of DEX_LOG_ERROR, DEX_LOG_WARNING, or DEX_LOG_INFO. The <i>pszFormat</i> is treated as a printf style format string, and it, along with the remaining arguments, are formatted and sent to the logging system at the specified level.
<b>put</b>	<b>abool_t pDict-&gt;put( dex_AttributeDictionary_t* pDict, const char* pszAttribute, const char* pszValue, int iIndex )</b>
	Converts <i>pszValue</i> to a sequence of bytes, according to the definition of <i>pszAttribute</i> in the server configuration. Associates that sequence of bytes with the attribute in the dictionary. If <i>iIndex</i> is the special value DEX_REPLACE, replaces any existing instances of the attribute with a single value. If the special value DEX_APPEND, it appends a new instance of the attribute to its list. If the special value DEX_AUGMENT, puts the attribute only if there is not one already. Otherwise, inserts a new instance at the position indicated. Returns TRUE unless the attribute name does not match any configured attributes or the value could not be converted to a legal value.
<b>putByType</b>	<b>abool_t pDict-&gt;putByType( dex_AttributeDictionary_t* pDict, const abytes_t* pszAttribute, const char* pszValue, int iIndex )</b>
	Converts <i>pszValue</i> to a sequence of bytes, according to the definition of <i>pszAttribute</i> in the server configuration. Associates that sequence of bytes with the attribute in the dictionary. If <i>iIndex</i> is the special value DEX_REPLACE, replaces any existing instances of the attribute with a single new value. If the special value DEX_APPEND, appends a new instance of the attribute to its list. If the special value DEX_AUGMENT, puts the attribute only if there is not one already. Otherwise, inserts a new instance at the position indicated.
<b>putBytes</b>	<b>abool_t pDict-&gt;putBytes( dex_AttributeDictionary_t* pDict, const char* pszAttribute, const abytes_t* pszValue, int iIndex )</b>
	Associates <i>pszValue</i> with the <i>pszAttribute</i> in the dictionary. If <i>iIndex</i> is the special value DEX_REPLACE, replaces any existing instances of the attribute with a single new value. If the special value DEX_APPEND, appends a new instance of the attribute to its list. If the special value DEX_AUGMENT, puts the attribute only if there is not one already. Otherwise, inserts a new instance at the position indicated. Returns TRUE unless the attribute name does not match a configured one.
<b>putBytesByType</b>	<b>abool_t pDict-&gt;putBytesByType( dex_AttributeDictionary_t* pDict, const abytes_t* pszAttribute, const abytes_t* pszValue, int iIndex )</b>
	Associates <i>pszValue</i> with the <i>pszAttribute</i> in the dictionary. If <i>iIndex</i> is the special value DEX_REPLACE, replaces any existing instances of the attribute with the new value. If the special value DEX_APPEND, appends a new instance of the attribute to its list. If the special value DEX_AUGMENT, puts the attribute only if there is not one already. Otherwise, inserts a new instance of the attribute at the position indicated.

**Table C-8 DEX Request and Response Dictionary Methods (continued)**

Method	Syntax
<b>remove</b>	<b>abool_t</b> <i>pDict</i> -> <b>remove</b> ( <b>dex_AttributeDictionary_t*</b> <i>pDict</i> , <b>const char*</b> <i>pszAttribute</i> , <b>int</b> <i>iIndex</i> )
	Removes the attribute from the dictionary. If <i>iIndex</i> is the special value DEX_REMOVE_ALL, removes any existing instances of the attribute. Otherwise, removes the instance at the position indicated. Returns TRUE, even if the dictionary did not contain that attribute at that index, unless the attribute name does not match any configured one.
<b>removeByType</b>	<b>abool_t</b> <i>pDict</i> -> <b>removeByType</b> ( <b>dex_AttributeDictionary_t*</b> <i>pDict</i> , <b>const abytes_t*</b> <i>pszAttribute</i> , <b>int</b> <i>iIndex</i> )
	Removes the attribute from the dictionary. If <i>iIndex</i> is the value DEX_REMOVE_ALL, removes any existing instances of the attribute. Otherwise, removes the instance at the position indicated. Always returns TRUE, even if the dictionary does not contain that attribute at that index.
<b>trace</b>	<b>abool_t</b> <i>pDict</i> -> <b>trace</b> ( <b>dex_AttributeDictionary_t*</b> <i>pDict</i> , <b>int</b> <i>iLevel</i> , <b>const char*</b> <i>pszFormat</i> , ... )
	Returns a message in the DHCP server's packet tracing system. At level 0, no tracing occurs. At level 1, it traces only that the server received the packet and sent a reply. At level 4, it traces everything. The remaining arguments are concatenated and sent to the tracing system at the specified level. The default tracing is set using the DHCP server <i>extension-trace-level</i> attribute.

## DEX Environment Dictionary Methods

The environment dictionary uses active commands, called methods, with which you can change and access the dictionary values. Table C-9 lists the methods to use with the environment dictionary. In this case, you can define the *pDict* variable as **pEnviron**, as in:

```
pEnviron->log( pEnviron, DEX_LOG_INFO, "Environment hello world" );
```

**Table C-9 DEX Environment Dictionary Methods**

Method	Syntax
<b>allocateMemory</b>	<b>void*</b> <i>pDict</i> -> <b>allocateMemory</b> ( <b>dex_EnvironmentDictionary_t*</b> <i>pDict</i> , <b>unsigned int</b> <i>iSize</i> )
	Allocates memory for extensions that persists only for the lifetime of this request.
<b>clear</b>	<b>void</b> <i>pDict</i> -> <b>clear</b> ( <b>dex_EnvironmentDictionary_t*</b> <i>pDict</i> )
	Removes all entries from the dictionary.
<b>containsKey</b>	<b>abool_t</b> <i>pDict</i> -> <b>containsKey</b> ( <b>dex_EnvironmentDictionary_t*</b> <i>pDict</i> , <b>const char*</b> <i>pszKey</i> )
	Returns TRUE if the dictionary contains the key, otherwise FALSE.
<b>firstKey</b>	<b>const char*</b> <i>pDict</i> -> <b>firstKey</b> ( <b>dex_EnvironmentDictionary_t*</b> <i>pDict</i> )
	Returns the name of the first key in the dictionary. Note that the keys are not stored sorted by name. If a key does not exist, returns the empty string.
<b>get</b>	<b>const char*</b> <i>pDict</i> -> <b>get</b> ( <b>dex_EnvironmentDictionary_t*</b> <i>pDict</i> , <b>const char*</b> <i>pszKey</i> )
	Returns the value of the key from the dictionary. If a key does not exist, returns the empty string.

Table C-9 DEX Environment Dictionary Methods (continued)

Method	Syntax
<b>isEmpty</b>	<b>abool_t</b> <i>pDict</i> -> <b>isEmpty</b> ( <b>dex_EnvironmentDictionary_t*</b> <i>pDict</i> )
	Returns TRUE if the dictionary has 0 entries, otherwise FALSE.
<b>log</b>	<b>abool_t</b> <i>pDict</i> -> <b>log</b> ( <b>dex_EnvironmentDictionary_t*</b> <i>pDict</i> , <b>int</b> <i>eLevel</i> , <b>const char*</b> <i>pszFormat</i> , ... )
	Returns a message in the DHCP server's logging system. The <i>eLevel</i> should be one of LOG_ERROR, LOG_WARNING, or LOG_INFO. The <i>pszFormat</i> is treated as a printf style format string, and it, along with the remaining arguments, are formatted and sent to the logging system at the specified level.
<b>nextKey</b>	<b>const char*</b> <i>pDict</i> -> <b>nextKey</b> ( <b>dex_EnvironmentDictionary_t*</b> <i>pDict</i> )
	Returns the name of the next key in the dictionary that follows the key returned in the last call to <b>firstKey</b> or <b>nextKey</b> . If a key does not exist, returns the empty string.
<b>put</b>	<b>abool_t</b> <i>pDict</i> -> <b>put</b> ( <b>dex_EnvironmentDictionary_t*</b> <i>pDict</i> , <b>const char*</b> <i>pszKey</i> , <b>const char*</b> <i>pszValue</i> )
	Associates a value with the key, replacing an existing instance of the key with the new value.
<b>remove</b>	<b>abool_t</b> <i>pDict</i> -> <b>remove</b> ( <b>dex_EnvironmentDictionary_t*</b> <i>pDict</i> , <b>const char*</b> <i>pszKey</i> )
	Removes the key and the associated value from the dictionary. Always returns TRUE, even if the dictionary did not contain the key.
<b>size</b>	<b>int</b> <i>pDict</i> -> <b>size</b> ( <b>dex_EnvironmentDictionary_t*</b> <i>pDict</i> )
	Returns the number of entries in the dictionary.
<b>trace</b>	<b>abool_t</b> <i>pDict</i> -> <b>trace</b> ( <b>dex_EnvironmentDictionary_t*</b> <i>pDict</i> , <b>int</b> <i>iLevel</i> , <b>const char*</b> <i>pszFormat</i> , ... )
	Returns a message in the DHCP server's packet tracing system. At level 0, no tracing occurs. At level 1, it traces only that the server received the packet and sent a reply. At level 4, it traces everything. The remaining arguments are concatenated and sent to the tracing system at the specified level. The default tracing is set using the DHCP server <i>extension-trace-level</i> attribute.