DHCP 拡張ディクショナリ

この付録では、DHCP 拡張ディクショナリエントリと、拡張ディクショナリへのアプリケーション プログラム インターフェイス (API) について説明します。このクラスは、要求ディクショナリと応答ディクショナリで使用できるデータ項目、および Tcl 拡張機能および共有ライブラリから辞書にアクセスするときに使用する呼び出しについて説明します。

この付録の構成は、次のとおりです。

拡張ディクショナリ エントリ

ディクショナリは、キーと値のペアを含むデータ構造です。ディクショナリには、要求ディクショナリと応答ディクショナリで使用する属性ディクショナリと環境ディクショナリの2種類があります。このセクションでは、要求ディクショナリと応答ディクショナリについて説明します。環境辞書のエントリについては、TCL 環境ディクショナリ メソッド （49 ページ）で説明します。

復号化された DHCP パケット データ項目

デコードされた DHCPv4 パケット データ項目は、DHCP パケットの情報を表し、要求ディクショナリと応答ディクショナリの両方で使用できます。これらのディクショナリは、デコードされた要求とデコードされた応答よりもかなり多くの内部サーバーデータ構造にアクセスできます。

アスタリスク ()*が付いたすべてのオプションは複数であり、各オプションに複数の値が関連付けられている可能性があります。DHCP/BOOTP パケットでは、これらのデータ項目はすべて同じオプションに表示されます。ただし、拡張インターフェイスでは、インデックスを使用してこれらの複数のデータ項目にアクセスできます。

DHCP 拡張ディクショナリ

名前を持たないオプションには、オプション –nを指定します。 表 3 : DHCPv4 およびブート・オプション （3 ページ）すべてのフィールドは読み取り/書き込み可能です。表 1 : DHCPv4 およびブート・フィールド （2 ページ） DHCPv4 パケットのフィールド値を記述します。表 2 : DHCPv6 フィールド （2 ページ）は DHCPv6 メッセージのフィールド値を記述しています。

表 *1 : DHCPv4* およびブート・フィールド

名前	値
chaddr	blob (バイトのシーケンス)
ciaddr	IP アドレス
file	文字列
Flags	16 ビットの符号なし整数
giaddr	IP アドレス
hlen	8 ビットの符号なし整数
hops	8 ビットの符号なし整数
htype	8 ビットの符号なし整数
op	8 ビットの符号なし整数
secs	16 ビットの符号なし整数
siaddr	IP アドレス
sname	文字列
xid	32 ビットの符号なし整数
yiaddr	IP アドレス

表 *2 : DHCPv6* フィールド

名前	値
hop-count	8 ビットの符号なし整数
link-address	[IPv6 アドレス（IPv6 address）]
msg-type	8 ビットの符号なし整数
peer-address	[IPv6 アドレス（IPv6 address）]
xid	32 ビットの符号なし整数

次の表に、DHCPv4 の DHCP オプションと BOOTP オプションを示します。

表 *3 : DHCPv4* およびブート・オプション

名前（* = 複数値）	ケース	値
6rd	212	binary
access-domain	213	DNS name
すべてのサブネットローカル	27	バイト値ブール型
andsf-v4	142	IP アドレス
arp-cache-timeout	35	符号なし時間
認証	90	blob(バイトのシーケンス)。5フィールド
auto-configure	116	8 ビットの符号なし整数
ベースタイム	152	date
bcmcs サーバー-a*	89	IP アドレス
bcmcs サーバー-d*	88	DNS name
boot-file	67	文字列
ブートサイズ	13	16 ビットの符号なし整数
broadcast-address	36	IP アドレス
ケーブルラボ-125(v-i-ベンダー情報 ID: 4491)	125 サブオプション：	binary
oro	1	オプション要求、8 ビット符号なし整数 (8 ビット符号なし整数)
tftp-servers	2	TFTP サーバーの IP アドレス
eルーターコンテナ	3	Erouter コンテナ オプション (バイナリ;TLV エンコード オプション)
パケットケーブルミブ・エンブ	4	MIB 環境インジケーター (8 ビット列挙)
モデム機能	5	モデム機能エンコーディング(バイナリ;TLV5 エンコードデータ)
acs-server	6	ACS サーバー サブオプション(バイナリ)

DHCP 拡張ディクショナリ

名前（* = 複数値）	ケース	値
radius-server	7	RADIUS サーバー サブオプション (バイナリ)
dhcpv6-servers	123	DHCPv6 サーバーサブオプション (バイナリー)
IP-プレフ	124	IPv4 または IPv6 の基本設定 (8 ビット列挙)
ケーブルラボ-クライアント-コンフィギュレーション	122 サブオプション：	BLOB（バイト シーケンス）
primary-dhcp- server	1	IP アドレス（IP Address）
secondary-dhcp- server	2	IP アドレス
provisioning- server	3	BLOB (最初のバイトはタイプバイトで、RFC 1035 エンコーディングでは 0、IP アドレスエンコーディングの場合は1で、アドレスはネットワーク順にする必要があります)
バックオフ再試行 - BLOB	4	12 バイトの BLOB (3 つの符号なし 4 バイト整数、ネットワーク順にする必要があります)。Kerberos AS-REQ/AS-REP タイムアウト、バックオフ、および再試行メカニズムを設定します
ap-backoff-再試行- BLOB	5	12 バイトの BLOB (3 つの符号なし 4 バイト整数、ネットワーク順にする必要があります)。Kerberos AP-REQ/AP-REP タイムアウト、バックオフ、および再試行メカニズムを設定します。
kerberos-realm	6	可変長ブロブ(RFC 1035スタイル名)。Kerberos 領域名が必要です
使用-tgt	7	1 バイトの符号なし整数ブール値。アプリケーション サーバーの 1 つのサービス チケットを取得するときに、チケット保証チケット (TGT) を使用するかどうかを示します

DHCP 拡張ディクショナリ

名前（* = 複数値）	ケース	値
provisioning-timer	8	1 バイトの符号なし整数。プロビジョニングプロセスの完了に必要な最大時間を定義します。
チケットコントロール-マスク	9	ホスト順に 2 バイトの符号なし整数
kdc アドレス - ブロブ	10	可変長 (4 の倍数) IP アドレス (ネットワーク順)
captive-portal	114	文字列
captive-portal-old	160	文字列
capwap-ac-v4*	138	IP アドレス
シスコ自動設定	251	境界バイト
シスコ クライアント-最終トランザクション時間	163	32 ビットの符号なし整数
シスコ クライアント要求ホスト名	162	文字列
シスコ-VPN ID	221	ブロブ (構造化)
クラスレス静的ルート	121	BLOB（構造化）
client-fqdn	81	blob (バイトのシーケンス)。4 フィールド: フラグ、rcode-1、rcode-2、およびドメイン名
クッキーサーバー*	8	IP アドレス
data-source	157	8 ビットの符号なし整数
デフォルト-ip-ttl	23	8 ビットの符号なし整数
デフォルト-tcp-ttl	37	8 ビットの符号なし整数
dhcp4o6-s46-saddr	109	[IPv6 アドレス（IPv6 address）]
dhcp-class-identifier	60	文字列
dhcp-client-identifier	61	BLOB（バイト シーケンス）
dhcp リース時間	51	符号なし時間
メッセージサイズ	57	16 ビットの符号なし整数

DHCP 拡張ディクショナリ

名前（* = 複数値）	ケース	値
dhcp メッセージ	72	文字列
dhcp-message-type	53	8 ビットの符号なし整数
dhcp オプション過負荷	52	8 ビットの符号なし整数
dhcp-parameter-request-list*	55	8 ビットの符号なし整数
dhcp パラメーター要求 - リスト BLOB*	55	BLOB（バイト シーケンス）
dhcp 再バインド時間	59	符号なし時間
dhcp 更新時間	58	符号なし時間
dhcp-requested-address	50	IP アドレス
DHCP サーバー識別子	54	IP アドレス
状態	156	8 ビットの符号なし整数
dhcp-user-class-id	77	カウントされた len バイト配列のセット。2 つのフィールド: サイズとユーザー データ
domain-name	15	文字列
domain-name-servers*	6	IP アドレス
ドメイン検索	119	BLOB（バイト シーケンス）
dot-address*	148	IP アドレス
dots-ri	147	DNS name
終了	255	長さなし
拡張機能パス	18	文字列
指サーバー*	73	IP アドレス
フォントサーバー*	48	IP アドレス
力を更新可能*	145	8 ビットの符号なし整数
ジオコンフェ	123	BLOB（バイト シーケンス）
ジオコンフィシビック	99	BLOB（バイト シーケンス）
ジオロック	144	binary

名前（* = 複数値）	ケース	値
host-name	12	文字列
ieee802.3-encapsulation	36	バイト値ブール値
印象づけるサーバー*	10	IP アドレス
interface-mtu	26	16 ビットの符号なし整数
ip-forwarding	19	バイト値ブール型
ipv6-only-preferred	108	32 ビットの符号なし整数
ircサーバー*	74	IP アドレス
iSNS	83	blob(バイトのシーケンス)。7フィールド
ldap-url	95	文字列
log-servers*	7	IP アドレス
失われたサーバー	137	DNS 名 (RFC 5223 を参照)
lpr サーバー*	9	IP アドレス
lq 関連付け IP*	92	IP アドレス
lq クライアント最終トランザクション時間	91	符号なし時間
マスクサプライヤー	30	バイト値ブール型
マックス・ドグラム再構成	22	16 ビットの符号なし整数
mcns-security-server	128	IP アドレス
メリットダンプ	14	文字列
モバイルip-ホームエージェント*	68	IP アドレス
モスアドレス	139 サブオプション：	バイナリ;3 サブオプション
は	1	IP アドレス（IP Address）
cs	2	IP アドレス
es	3	IP アドレス

DHCP 拡張ディクショナリ

名前（* = 複数値）	ケース	値
モスト fqdn	140 サブオプション：	バイナリ;3 サブオプション
は	1	DNS name
cs	2	DNS name
es	3	DNS name
泥のURL	161	文字列
name-servers*	5	IP アドレス
ネームサービス検索*	117	16 ビットの符号なし整数
nds コンテキスト	87	文字列
nds-サーバー*	85	IP アドレス
nds ツリー	86	文字列
ネットビオス-ddサーバー*	45	IP アドレス
netbios-name-servers*	44	IP アドレス
netbios-node-type	46	8 ビットの符号なし整数
ネットビオススコープ	47	文字列
ネットインフォ親サーバーアドイン	112	IP アドレス
ネットインフォ親サーバータグ	113	文字列
ネットウェアイップドメイン	62	文字列
ネットウェア情報	63	BLOB（バイト シーケンス）
nis+ドメイン	64	文字列
nis+サーバー*	65	IP アドレス
nis-ドメイン	40	文字列
nis-サーバー*	41	IP アドレス
nntp サーバー*	71	IP アドレス

名前（* = 複数値）	ケース	値
非ローカル ソース ルーティング	20	バイト値ブール型
ntp-servers*	54	IP アドレス
パッド	[0]	長さなし
パナエージェント	136	IP アドレス (RFC 5192 を参照)
パス-mtu エージング タイムアウト	24	符号なし時間
パス-mtu-プラトーテーブル*	25	16 ビットの符号なし整数
マスク検出の実行	29	バイト値ブール型
ポリシー フィルター*	21	IP アドレス (2 つのポリシー フィルタがあり、それぞれに独自の IP アドレスを持つことができます)
pop3-servers*	70	IP アドレス
ポシックスタイムゾーン	100	文字列 (RFC 4833 を参照)
pxe クライアント アーチ	93	16 ビットの符号なし整数
pxe クライアント-マシン ID	97	blob (バイトのシーケンス)。2 フィールド: タイプフラグと uuid
pxe クライアント ネットワーク ID	94	blob (バイトのシーケンス)。2 フィールド: タイプフラグとバージョン
pxelinux-コンフィグファイル	209	文字列
pxelinux パス接頭辞	210	文字列
pxelinux-リブート時間	211	符号なし時間
クエリ終了時刻	155	date
クエリ開始時刻	154	date
rapid-commit	80	ヌル長さ
選択	146	バイナリ;4つのフィールド:予約済み、プライマリ再帰的な名前サーバー、セカンダリ再帰的な名前サーバー、およびドメインとネットワーク

DHCP 拡張ディクショナリ

名前（* = 複数値）	ケース	値
リレーエージェント情報サブオプション:	82 サブオプション：	BLOB（バイト シーケンス）
relay-agent-circuit-id- data	1	BLOB（バイト シーケンス）
relay-agent-remote-id- data	2	BLOB（バイト シーケンス）
リレーエージェント-デバイス-クラスデータ	4	4 バイト符号なし整数
リレーエージェント-サブネット- 選択データ	5	IP アドレス
subscriber-id	6	ネットワーク クライアントまたはサブスクライバを識別する文字列
radius-attributes	7	サポートされる属性は、ユーザー、クラス、およびフレームプールです。
認証	8	binary
v-i-vendor-opts	9	ベンダー オプション
シスコサブネット選択	150	IP アドレス
シスコ-VPN ID	151	binary
シスコ サーバー ID オーバーライド	152	IP アドレス
（注） サブオプション・データの前に 2 バイト (サブオプション・コードとデータ長) を戻したリレー・エージェント回線 ID、リレー・エージェント・リモート ID、およびリレー・エージェント・デバイス・クラス・サブオプションは非推奨ですが、まだ使用可能です。		
リソースロケーションサーバー*	11	IP アドレス
root-path	17	文字列
ルーター発見	31	バイト値ブール型
ルーター勧誘アドレス	32	IP アドレス
routers*	3	IP アドレス
sip-servers	120	blob (バイトのシーケンス)。2 フィールド: フラグと sip サーバー リスト

名前（* = 複数値）	ケース	値
一ロウアcsドメイン	141	DNS name
slp-ディレクトリエージェント*	78	blob (バイトのシーケンス)。2 フィールド: 必須およびエージェント IP リスト
slp サービス スコープ*	79	blob (バイトのシーケンス)。2 フィールド: 必須および slp スコープリスト
smtp-servers*	69	IP アドレス
start-time-of-state	153	符号なし時間
static-routes*	33	IP アドレス
status-code	151	バイナリ;2 つのフィールド: ステータス コードとステータス メッセージ
ストリートトークディレクトリ - アシスタンスサーバー*	76	IP アドレス
ストリートトークサーバー*	75	IP アドレス
サブネット-アロク	220	blob（バイトのシーケンス）。5 つのフィールド : フラグ、サブネット要求、サブネット情報、サブネット名、サブネット推奨リース時間
subnet-mask	1	IP アドレス（IP Address）
subnet-selection	118	IP アドレス
スワップサーバー	16	IP アドレス
リダイレクト	143	カウントされた len バイト配列のセット。2 つのフィールド: サイズと url
tcp-キープアライブゴミ	39	バイト値ブール型
TCP キープアライブ内部	38	符号なし時間
tftp-server	66	文字列
tftp-server-address*	150	IP アドレス
time-offset	2	署名された時間
time-servers*	4	IP アドレス

名前（* = 複数値）	ケース	値
トレーラーカプセル化	34	バイト値ブール型
tzdb タイムゾーン	101	文字列（RFC 4833 を参照）
ユーザー認証	98	文字列
v4-pcp-server*	158	binary
v4-portparams	159	バイナリ;3 つのフィールド: オフセット、psid-len、および psid
v-i-ベンダークラス	124	BLOB（バイト シーケンス）
v-i-ベンダー情報	125	BLOB（バイト シーケンス）
vendor-encapsulated-options	43	BLOB（バイト シーケンス）
vpn-id	185	ブロブ(構造化)。2 フィールド: フラグと vpn-id
www サーバー*	72	IP アドレス
xディスプレイマネージャ*	49	IP アドレス

次の表に、DHCPv6 オプションを示します。

（注） これらのオプションへのアクセスは、 、 putOptiongetOption、およびremoveOptionメソッドを使用してのみ使用できます。

表 *4 : DHCPv6* のオプション

名前（* = 複数値）	ケース	値
4rd	97	コンテナ (オプション)
4rd-map-rule	98	バイナリ;6 つのフィールド: prefix4-len、プレフィックス6レン、ea-len、フラグ、ルール ipv4 プレフィックス、およびルール ipv6 プレフィックス
4rd-non-map-rule	99	バイナリ;3 つのフィールド: フラグ、トラフィック クラス、およびドメイン pmtu
access-domain	57	DNS name
アドルセル	84	バイナリ;1 フィールド: 予約済み AP

名前（* = 複数値）	ケース	値
アドルセルテーブル	85	バイナリ;3 つのフィールド: ラベル、優先順位、およびプレフィックス
aftr-name	64	DNS name
アニ・アプ・ブシド	108	BLOB（バイト シーケンス）
アニ・アプ・ネーム	107	文字列
アニアット	105	バイナリ;2 フィールド: 予約済みおよび att
ani-ネットワーク名	106	文字列
ani演算子 ID	109	BLOB（バイト シーケンス）
アニ演算子レルム	110	文字列
auth	11	バイナリ;5つの分野:プロトコル、アルゴリズム、リプレイ検出方式、リプレイ検出、認証情報
bcmcs-サーバー-a*	34	[IPv6 アドレス（IPv6 address）]
bcmcs-サーバー-d*	33	DNS name
ブートファイルパラム	60	カウント型。2 つのフィールド: typecnt サイズとパラメーター
ブートファイル-URL	59	文字列
cablelabs-17 (vendor-opts ID: 4491)	17 サブオプション：	vendor-opts; 27 サブオプション
oro	1	16 ビットの符号なし整数
device-type	2	string
埋め込みコンポーネント-リスト	3	文字列
device-serial-number	4	文字列
hardware-version-number	5	文字列
ソフトウェアバージョン番号	6	string
ブート・ロム・バージョン	7	文字列

DHCP 拡張ディクショナリ

名前（* = 複数値）	ケース	値
vendor-oui	8	文字列
モデル番号	9	文字列
vendor-name	10	文字列
ecm-cfg-カプセル化	15	文字列
tftp-servers	32	[IPv6 アドレス（IPv6 address）]
config-file-name	33	文字列
syslog-servers	34	[IPv6 アドレス（IPv6 address）]
モデム機能	35	binary
device-id	36	binary
rfc868-servers	37	[IPv6 アドレス（IPv6 address）]
time-offset	38	符号付き時間
IP-プレフ	39	8 ビットの符号なし整数
acs-server	40 サブオプション：	バイナリ;2 サブオプション
flag	[0]	8 ビットの符号なし整数
サーバー	[0]	
radius-server	41 サブオプション：	バイナリ;2 サブオプション
flag	[0]	8 ビットの符号なし整数
サーバー	[0]	
セル ID	54	[IPv6 アドレス（IPv6 address）]
チャッピングコア	61	[IPv6 アドレス（IPv6 address）]
cmts機能	1025	binary
cm-mac-address	1026	binary
eルーターコンテナ	1027	binary

名前（* = 複数値）	ケース	値
ケーブルラボ-クライアント構成	2170 サブオプション：	バイナリ;2 サブオプション (各種データ・タイプ)
primary-dhcp-server	1	IP アドレス（IP Address）
secondary-dhcp-server	2	IP アドレス
cablelabs-client-configuration-v6	2171 サブオプション：	バイナリ;9 サブオプション (各種データ・タイプ)
プライマリ dhcpv6-server- セレクタ ID	1	binary
セカンダリ dhcpv6-server- セレクタ ID	2	binary
provisioning-server	3	binary
バックオフ再試行	4	binary
ap-backoff 再試行	5	binary
kerberos-realm	6	DNS name
使用-tgt	7	符号なし 8 ビット
provisioning-timer	8	符号なし 8 ビット
チケットコントロールマスク	9	符号なし 16 ビット
ケーブルラボ-相関 ID	2172	符号なし 32 ビット
captive-portal	103	文字列
capwap-ac-v6*	52	[IPv6 アドレス（IPv6 address）]
クライアントアーチタイプ*	61	符号なし 16 ビット
client-data	45	（オプションの）コンテナ
client-fqdn	39	バイナリ;2 フィールド: フラグとドメイン名
client-identifier	1	BLOB（バイト シーケンス）
クライアント リンクレイヤー アドレス	79	バイナリ;2 つのフィールド: リンク層タイプとリンク層アドレス

DHCP 拡張ディクショナリ

名前（* = 複数値）	ケース	値
clt-time	46	符号なし時間 (RFC 5007 を参照)
dhcp4-o-dhcp6-server	88	[IPv6 アドレス（IPv6 address）]
dhcpv4-msg	87	BLOB（バイト シーケンス）
dns-servers*	23	[IPv6 アドレス（IPv6 address）]
domain-list	24	DNS name
dot-address*	142	[IPv6 アドレス（IPv6 address）]
dots-ri	141	DNS name
elapsed-time	8	符号なし 16 ビット
エロ	43	符号なし 16 ビット (RFC 4994 を参照)
erp-ローカル・ドメイン名	65	DNS name
ジオコンフィシビック	36	binary
ジオロック	63	BLOB（バイト シーケンス）
ia-na	3	バイナリ;3つの分野:アイド、t1、およびt2
ia-pd	25	バイナリ、3個のフィールド（iaid、t1、t2）
ia-ta	4	バイナリ;1フィールド:アイド
イアアドル	5	バイナリ;3 つのフィールド: 住所、優先存続時間、および有効な有効期間
イアプレフィックス	26	バイナリ;4つのフィールド:優先存続時間、有効期間、プレフィックス長、およびプレフィックス
インフ-マックス-rt	83	符号なし時間
情報更新時間	32	符号なし時間
interface-id	18	BLOB（バイト シーケンス）
ipv6-adddress-andsf*	143	[IPv6 アドレス（IPv6 address）]
krb-デフォルト領域名	77	文字列
クルブ-クdc	78	バイナリ;5 つのフィールド: 優先順位、重み、トランスポート・タイプ、kdc-ipv6-アドレス、およびレルム名

名前（* = 複数値）	ケース	値
krb プリンシパル名	75	バイナリ;2 つのフィールド: 名前型と名前文字列
krb-レルム名	76	文字列
link-address	80	[IPv6 アドレス（IPv6 address）]
失われたサーバー	51	DNS 名（RFC 5223 を参照）
lqベースタイム	100	符号なし 32 ビット
lq クライアントリンク*	48	IPv6 アドレス (RFC 5007 を参照)
lq終了時刻	102	符号なし 32 ビット
lq クエリ	44	バイナリ構造 (RFC 5007 を参照)
lqリレーデータ	47	バイナリ (DHCPv6 メッセージ) (RFC 5007 を参照)
lq-開始時間	101	符号なし 32 ビット
mip6-haa	72	[IPv6 アドレス（IPv6 address）]
mip6-haf	73	DNS name
mip6-hnidf	49	DNS name
mip6-hnp	71	バイナリ;2 つのフィールド: プレフィックス長とプレフィックス
mip6-idinf	69	（オプションの）コンテナ
mip6-udinf	70	（オプションの）コンテナ
mip6-vdinf	50	（オプションの）コンテナ
モスアドレス	54 サブオプション：	バイナリ;3 サブオプション
は	1	[IPv6 アドレス（IPv6 address）]
cs	2	[IPv6 アドレス（IPv6 address）]
es	3	[IPv6 アドレス（IPv6 address）]

名前（* = 複数値）	ケース	値
モスト fqdn	55 サブオプション：	バイナリ;3 サブオプション
は	1	DNS name
cs	2	DNS name
es	3	DNS name
mpl パラメータ	104	BLOB（バイト シーケンス）
泥のURL	112	文字列
新しいポシックスタイムゾーン	41	文字列 (RFC 4833)
新しい tzdb タイムゾーン	54	文字列（RFC 4833）
nii	62	バイナリ;3 つのフィールド: タイプ、メジャー、マイナー
nis-domain-name*	29	DNS name
nis-サーバー*	27	IP アドレス
nisp-domain-name*	30	DNS name
nispサーバー*	36	IP アドレス
ntp-server	72	バイナリ;3 サブオプション (各種データ・タイプ)
oro	6	符号なし 16 ビット
パナエージェント*	40	IPv6 アドレス (RFC 5192 を参照)
pd除外	67	バイナリ;2 フィールド: プレフィックス長とサブネット ID
環境設定	7	符号なし 8 ビット
prefix64	113	バイナリ;3 つのフィールド: ASM-m プレフィックス64、SSM-m プレフィックス64、および u プレフィックス 64
radius	81	BLOB（バイト シーケンス）
rapid-commit	14	ゼロサイズ

名前（* = 複数値）	ケース	値
選択	74	バイナリ;3 つのフィールド: 再帰名サーバー、予約および prf、およびドメインとネットワーク
再設定-受け入れる	20	0 サイズ
再設定-メッセージ	19	符号なし 8 ビット
リレー エージェント サブスクライバー ID	38	binary
リレー ID	53	BLOB（バイト シーケンス）
リレーメッセージ	9	binary
リレーポート	135	符号なし 16 ビット
remote-id	37	バイナリ;2 フィールド: エンタープライズ ID とリモート ID
ルスー	66	（オプションの）コンテナ
s46-br	90	[IPv6 アドレス（IPv6 address）]
s46-cont-lw	96	（オプションの）コンテナ
s46-cont-mape	94	（オプションの）コンテナ
s46-cont-mapt	95	（オプションの）コンテナ
s46-dmr	91	IPv6 可変長接頭部
s46-portparams	93	バイナリ、3 個のフィールド（offset、psid-len、psid）
s46-priority*	111	符号なし 16 ビット
s46-rule	89	バイナリ;5 つのフィールド: フラグ、ea レン、プレフィックス4-len、ipv4-prefix、およびプレフィックス6
s46-v4v6bind	92	バイナリ;2 つのフィールド: ipv4 アドレスおよびバインド ipv6 接頭部
server-identifier	2	BLOB（バイト シーケンス）
サーバー ユニキャスト	12	[IPv6 アドレス（IPv6 address）]
サイプサーバーアドレス	22	[IPv6 アドレス（IPv6 address）]

DHCP 拡張ディクショナリ

名前（* = 複数値）	ケース	値
サイプサーバー名	21	DNS name
一ロウアcsドメイン	58	DNS name
サーバー*	31	[IPv6 アドレス（IPv6 address）]
ゾルマックス-rt	82	符号なし時間
status-code	13	バイナリ、2 個のフィールド（status-code、status-message）
リダイレクト	136	カウント型。2 つのフィールド：typecnt サイズと url
ユーザー クラス	15	カウント型。2 つのフィールド: typecnt サイズとユーザーデータ
v6-pcp-server*	86	[IPv6 アドレス（IPv6 address）]
vendor-class	16	vendor-class
vendor-opts	17	ベンダーオプト(cablelabs-17も参照)
vpn-id	68	バイナリ;2 フィールド: フラグと vpn-id

（注） 複数のインスタンス オプションもあります(つまり、1 つのオプションで複数の値だけでなく、複数のインスタンスを設定することもできます)。複数のインスタンスを持つことができるオプションは次のとおりです。

- ia-na
- ia-pd
- ia-ta
- イアアドル
- イアプレフィックス
- 選択
- s46-br
- s46-cont-mape
- v6-pcp-server

要求ディクショナリ

次の表は、要求ディクショナリでいつでも設定できるデータ項目を示しています。DHCPサーバーは、さまざまな時間にこれらのファイルを読み取ります。特に指定されていない限り、すべての操作は読み取り/書き込み可能です。

表 *5:* 要求ディクショナリ固有のデータ項目

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
アクティブリースクエリコントロール	int（v4、v6）
リースの送信を制御します(特定の状態の変更に対してのみなど)。値は、0(未指定(サーバーが通知を送信するかどうかを決定する)、1 - 送信(サーバーが通知を送信する)、および2は送信しない(サーバーは通知を送信しません) です。アクティブリースクエリコントロールは0(指定なし) として初期化されます。	
許可ブート	int（v4）
1に設定すると、この要求に対して任意のスコープに対してBOOTPを許可します。スコープ選択中およびリースの受容性の確認中に読み取り。	
許可する- dhcp	int（v4）
1 に設定すると、この要求のスコープに対して DHCP を許可します。リースの受け入れ確認中およびスコープの選択中に読み取ります。	
許可動的ブート	int（v4）
1に設定すると、この要求の任意のスコープに対して動的BOOTPを使用できます。リースの受け入れ確認中およびスコープの選択中に読み取ります。	
ブート応答オプション	ブロブ (v4)
任意のポリシー内のv4-bootp-replyオプションをオーバーライドします。出力パケットのデータを収集する際に読み取り。(IPv6 ブート応答オプションはありません。	
client-class-name	文字列 (v4、v6)
クライアント情報を完成させるために使用するクライアント クラスの名前 (存在する場合)。読み取り専用です。	
クライアント クラス ポリシー	string（v4、v6）
クライアントクラスに関連付けられているポリシーの名前。設定する場合は、サーバーで既に構成されているポリシーの名前を持つ必要があります。	
client-domain-name	string（v4、v6）

DHCP 拡張ディクショナリ

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
クライアントが使用するドメイン名。存在しない場合、DHCP サーバーはスコープで指定されたドメイン名を使用します。安定した記憶域の更新の直前にDNS更新の要求をキューに入れるときに読み取り。DHCPv6 の場合、クライアント FQDN 値を上書きし、DNS 更新に使用します。	
client-host-name	string（v4、v6）
DNS 内のクライアントのホスト名。安定したストレージを更新する直前に DNS 更新の要求をキューに入れる場合に読み取りが行われます。操作が完了すると、実際の名前がDNSに配置されます。DHCPv6 の場合、client-fqdn 値を上書きし、DNS 更新に使用します。	
client-id	ブロブ (v4, v6)
サーバーがクライアントを追跡するために使用するクライアント ID。クライアント ID は、要求と共に送信されるか、または MAC アドレスから内部的に生成されます。client-id-created-from-mac-address を参照してください。DHCPv6 の場合、クライアント識別子オプション値 (クライアントの DUID)。	
client-id-created-from-mac-address	int（v4）
1 に設定した場合、クライアント IDはクライアント提供の MAC アドレスから内部使用するために作成する必要があり、レポートで使用しないでください。	
クライアント ip アドレス	IP アドレス（v4）
クライアントがパケットを送信した IP アドレス。クライアントにまだ IP アドレスがない場合は、0 になる可能性があることに注意してください。	
クライアント制限 ID	blob（v4、v6）
クライアントの制限 ID。	
クライアントルックアップ ID	blob（v4、v6）
クライアントクラスのクライアントルックアップID式によって計算されたクライアントルックアップ ID。	
client-mac-address	blob（v4）
要求ディクショナリに関連付けられたクライアント オブジェクトに格納されている MAC アドレス。同じ形式 (およびから作成された) mac アドレスを持っています。	
クライアント-osタイプ	int（v4）
またはpre-client-lookup拡張ポイントでこれを設定して、要求パケットのクライアント エントリpost-client-lookupを変更します。で読むことcheck-lease-acceptableもできますが、そこで設定することはできません。値を設定するには、最初に要求ディクショナリで os-typepost-packet-decodeを設定する必要があります。	

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
client-packet	blob (v4、v6、読み取り専用)
受信パケットのクライアント部分。DHCPv4 の場合、これは完全なパケットです。DHCPv6 の場合、これはクライアント・メッセージです。(パケット全体を取得するには、パケットを参照してください。	
クライアント ポリシー	string（v4、v6）
クライアント エントリに関連付けられているポリシーの名前。設定する場合は、DHCP サーバーで構成済みのポリシーの名前を指定する必要があります。	
client-port	int (v4、v6)
クライアントが要求を送信したポート。	
クライアントが要求したホスト名	文字列 (v4)
クライアントが要求したホスト名が DNS 更新に使用されます。DHCP サーバーはこの情報を保存して、変更を検出できるようにします。	
クライアント ユニキャスト	ブール値 (v6、読み取り専用)
受信パケットがクライアントによってサーバーにユニキャストされた場合は true。	
クライアントが望む null-in-string	int（v4）
DHCP サーバーが、null で終了した文字列をクライアントに返すかどうかを判断します。1 に設定すると、サーバーは null で文字列を終了します。0 に設定した場合、null で文字列が終了することはありません。応答パケットpost-packet-decodeをエンコードする際に前に設定pre-packet-encodeし、 を読み取ります。	
派生 VPN-ID	int (v4、v6、読み取り専用)
VPN 識別子。詳細については、vpn-nameを参照してください。	
destination-ipaddress	IP アドレス (v6、読み取り専用)
パケットの宛先 IPv6 アドレス。	
dhcp 応答オプション	blob（v4、v6）
ポリシーで指定されたv4応答オプションまたはv6応答オプションをオーバーライドします。出力パケットのデータを収集する際に読み取り。	
ダンプパケット	int (v4、v6、書き込み専用)

DHCP 拡張ディクショナリ

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
1 に設定すると、Cisco Prime Network レジストラーは、現在デコードされた DHCP/BOOTP パケットをログファイルにダンプします。拡張機能は、その実行の複数のポイントでこのデータ項目に値 1 を配置できます。これは、拡張機能をデバッグするときに役立つ場合があります。	
フェールオーバー ロール	int (v4、v6、読み取り専用)
フェールオーバー サーバーの役割を決定します。フェールオーバー サーバーの役割は、次の 3 つの値のいずれかになります。 • • なし:フェールオーバーが構成されていません。 • メイン/バックアップ:フェールオーバーが構成され、フェールオーバー サーバーの役割が構成されます。	
failover-state	int（v4、v6、読み取り専用）
フェールオーバー サーバーの状態を決定します。フェールオーバーの状態は、正常、パートナーダウン、通信中断、回復、競合の可能性、回復完了、起動、シャットダウン、または一時停止状態にすることができます。フェールオーバーが設定されていない場合、この値は None になります。	
インポート パケット	int（v4）
サーバーがパケットをインポート クライアントから送信されたパケットとして扱うかどうかを決定します。1 に設定すると、サーバーはクライアントをインポート クライアントとして扱い、ACK を送信する前にクライアントに対するすべての DNS 操作を実行します。サーバーのインポート モードを確認するときに (post-packet-decode直後)、DNS 処理の準備、および応答アドレスの設定時に読み取ります。	
制限カウント	int（v4）
同じ制限 IDで許可される同時ユーザー数。	
limitation-id	blob（v4）
この要求が存在するクライアント クラスの制限 id式 (存在する場合) によって計算されます。	
制限 id-null	int（v4、v6）
制限 idが null の場合は 1(TRUE) に設定し、別の値の場合は 0 (FALSE) に設定します。	
ログクライアント基準処理	int（v4、v6）
1 に設定すると、この要求に対するクライアントの基準処理がログに記録されます。クライアントがリースを持たないクライアントの新しいリースを取得しようとする場合、およびリースの受け入れ可能性を確認する場合に、読み取り。	
ログクライアントの詳細	int（v4、v6）

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
1 に設定すると、この要求に対するクライアント クラスの処理がログに記録されます。クライアント クラス処理の最後、post-client-lookup後のを読み取ります。	
ログ-dns-更新-詳細	int（v4、v6)
1 に設定すると、この要求の DNS 更新の詳細がログに記録されます。	
ログドロップブートパケット	int（v4）
1 に設定すると、ログはこの要求に対して BOOTP パケットをドロップします。	
ログドロップされた dhcp-パケット	int（v4、v6)
1 に設定すると、ログはこの要求に対して DHCP パケットを廃棄します。	
ログ ドロップ待機パケット	int（v4、v6)
1 に設定すると、ログはこの要求の待機パケットをドロップします。	
ログ フェールオーバーの詳細	int（v4）
1 に設定すると、すべてのフェールオーバー状態の変更など、より詳細なレベルのフェールオーバー アクティビティがログに記録されます。	
ログ着信パケットの詳細	int（v4、v6)
1に設定すると、この要求に対して詳細な着信パケット トレースが発生したかどうかをチェックし、個別のトレースを配置する必要がないようにします。パケットデコード前と最初の拡張ポイントを読み取ります。	
ログ着信パケット	int（v4、v6)
1に設定すると、この要求の着信パケットがログに記録されます。の後post-decode-packetで読んでください。	
ログ ldap 作成の詳細	int（v4）
1 に設定すると、DHCP サーバーがリース状態エントリの作成を開始するたびにメッセージがログに記録され、LDAP サーバーからの応答を受信したり、結果またはエラー メッセージを取得したりします。	
ログ ldap クエリの詳細	int（v4、v6)
1 に設定すると、DHCP サーバーが LDAP サーバーに対する照会を開始したり、LDAP サーバーから応答を受け取ったり、照会結果またはエラー・メッセージを取得したりするたびに、メッセージがログに記録されます。	
ログ ldap 更新の詳細	int（v4）

DHCP 拡張ディクショナリ

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
1 に設定すると、DHCP サーバーが更新リース状態を開始するたびにメッセージがログに記録されるか、からの応答を受信するか、LDAP サーバーから結果またはエラー・メッセージを取得します。	
ログリースクエリ	int（v4、v6)
1 に設定すると、leasequery パケットが内部エラーなしで処理され、ACK または NAK が生成されたときにメッセージがログに記録されます。	
ログ欠落オプション	int（v4、v6)
1 に設定すると、不足しているオプション (クライアントが要求するが DHCP サーバーから返せない) をログに記録します。応答のデータを収集しながら読み取ります。	
ログ発信パケットの詳細	int（v4、v6)
1 に設定すると、この要求の発信パケットの詳細なダンプが記録されます。pre-packet-encode パケットを DHCP クライアントに送信する直前に読み取ります。	
ログ成功メッセージ	int（v4、v6)
1 に設定すると、成功メッセージがログに記録されます。	
ログ不明基準	int（v4、v6)
1 に設定すると、この要求のクライアント包含基準または除外条件で指定された不明な条件がログに記録されます。新しいクライアントリースを取得するとき、または既存のクライアントのリース受諾性を確認するときに読み取り。	
log-v6-lease-detail	int（v6）
1 に設定すると、DHCPv6 リースアクティビティに関する個々のメッセージがログに記録されます。	
mac-address	blob（v4)
クライアント パケットに入っている MAC アドレス。最初のバイトはハードウェアの種類、2 番目のバイトはハードウェアの長さ、残りのバイトはの直後post-packet-decodeに読み取られたchaddrからの情報です。これは、DHCP パケットのhtype、hlen、およびchaddrフィールドの便利な集約です。読み取り時には、これらのフィールドから構築されます。書き込まれると、これらのフィールドに配置されます。	
最大クライアントルックアップ	整数 (v4、v6)
許容するクライアント データベース ルックアップの最大数。通常は 2 などの小さな整数です。プリセット値は 1 です。	
override-client-id	blob（v4、v6）

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
現在のクライアント ID 値に使用される BLOB。着信パケットのクライアント ID を置き換えます (両方の値はリース状態データベースに保持されます)。	
オーバーライドクライアント ID データ型	文字列 (v4、v6、読み取り専用)
オーバーライドクライアント IDのデータ型 (文字列の場合は "nstr" 、BLOB の場合は "blob") を返します。	
オーバーライドクライアント ID 文字列	string（v4、v6）
受信パケットのクライアント ID を置き換える文字列形式の現在のクライアント ID 値 (両方の値はリース状態データベースに保持されます)。getの場合、オーバーライド クライアント IDが文字列でない場合、バイナリ データは BLOB データとして書式設定され、"string" として返されます。	
パケット	blob（v4、v6）
受信パケット。DHCPv4 の場合、これはクライアントパケットと同じです。DHCPv6 の場合、リレーが行われた場合はフルパケット、リレーされない場合はクライアントパケットと同じパケットになります。これは、パケットデコード前の拡張ポイントからのみ書き込む必要があります。その後、サーバーはクライアントから受信したパケットではなく、この新しいパケットをデコードします。	
ping-clients	int（v4）
1 に設定すると、この要求のリースを提供する前に ping を実行します。リースがクライアントに対して許容されるかどうかを判断する前に、直前に読んでください。	
relay-agent-circuit-id	blob（v4、v6）
オプション 82 の回線 ID サブオプションの内容。	
リレーエージェント-サーキット ID データ	blob（v4、v6）
オプション 82 の circuit-id サブオプションのデータ部分のみの内容。	
リレーエージェント-デバイス-クラスデータ	blob（v4、v6）
オプション 82 の装置クラス・サブオプションの内容。	
リレー エージェントの半径属性	blob（v4)
オプション 82 の半径サブオプションの内容。	
リレーエージェント半径クラス	string（v4）
オプション 82 の radius サブオプションのカプセル化された class 属性。	
リレーエージェント半径プール名	string（v4）

DHCP 拡張ディクショナリ

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
オプション 82 の radius サブオプションのカプセル化された framed-pool 属性。	
リレーエージェント半径-ユーザー	string（v4）
オプション 82 の radius サブオプションのカプセル化された user 属性。	
relay-agent-remote-id	blob（v4、v6）
オプション 82 のリモート ID サブオプションの内容。	
relay-agent-remote-id	blob（v4、v6）
オプション 82 の remote-id サブオプションのデータ部分のみの内容。	
リレー エージェント サーバー ID オーバーライド データ	IPv6 アドレス (v4、v6)
オプション 82 のサーバー ID サブオプションの内容。IANA サブオプション 182 がパケット内にある場合、その値が表示されます。それ以外の場合は、Cisco サブオプション 152 の値が表示されます。	
relay-agent-subscriber-id	string（v4）
オプション 82 のサブスクライバー ID サブオプションの内容。	
リレーカウント	int (v6、読み取り専用)
DHCPv6 リレー ホップの数。	
返信オプション	blob
任意のポリシーで指定された DHCPv4 応答オプションをオーバーライドします。出力パケットのデータを収集するときに読み取り。	
クライアントアドレスへの返信	int（v4、v6）
v4 の場合、1 に設定すると、サーバーは応答パケットをクライアント ip アドレスとクライアントポートに送信します。v6 の場合、1 に設定すると、サーバーは応答パケットを送信側 (クライアントまたはリレー エージェント) のアドレスとポートに返送します。0 の場合、サーバーは RFC の強制アルゴリズムを使用して応答を送信します。	
予約アドレス	IP アドレス (v4、読み取り/書き込み)
クライアント用に予約されているアドレスのリスト。使用可能なスコープに一致する最初の使用可能なアドレス (予約への制限が有効になっている必要があります) がクライアントに割り当てられます。	
reserved-ip6addresses	IP アドレス (v6、読み取り/書き込み)

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
クライアント用に予約されているアドレスのリスト。使用可能なプレフィックスに一致する使用可能なすべてのアドレス (予約に制限が有効になっている必要があります) がクライアントに割り当てられます。	
予約済みプレフィックス	IP アドレス（v6、読み取り/書き込み）
クライアント用に予約されているプレフィックスのリスト。使用可能なプレフィックスに一致するすべての使用可能なプレフィックス (予約制限が有効になっている必要があります) がクライアントに割り当てられます。	
selection-criteria	string（v4、v6）
範囲選択基準を含むコンマ区切りの文字列。	
選択基準 -除外	string（v4、v6）
範囲の除外条件を含むコンマ区切りの文字列。	
送信-ack-ファースト	int（v4、v6)
サーバーはこのデータ項目を無視します。ただし、下位互換性のために、読み込みまたは設定は可能ですが、無視されます。デフォルト値は、0（false）です。	
source-ipaddress	IPv6 アドレス (v6、読み取り専用)
パケットの IPv6 送信元アドレス。	
トレース ID	string（v4、v6、読み取り専用）
パケットをトレースするためにシステムが使用する ID。	
トランザクション時間	int（v4、v6)
入力パケットがデコードされた時間 (1970 年以降の秒数)。	
更新-dns	string（v4、v6）
要求パケットごとに部分的、完全、または動的 DNS 更新を要求しません。入力値と出力値は、1=すべて更新、2=更新-fwdのみ、3=更新-rev-only、および 0=更新なしです。	
update-dns-for-bootp	int（v4）
1 に設定すると、この要求に対する BOOTP 要求の DNS が更新されます。BOOTP の DNS 操作を初期化する直前に読み取ります。	
詳細ログ	int（v4、v6)
1 に設定すると、この要求の詳細メッセージがログに記録されます。処理中のさまざまな時間に読み取り。	

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
VPN 説明	string（v4、v6、読み取り専用）
VPN の説明。詳細については、「vpn-name」を参照してください。	
vpn-name	string（v4、v6、読み取り専用）
VPN の名前。要求ディクショナリには、post-packet-decodeこれらの項目の有効な値が 含まれていませんが、VPNがまだ決定されていないため、他のすべての拡張ポイントで有効な値が設定されています。これは、スクリプトがderived-vpn-idオプションまたはサブオプションを変更post-packet-decodeし、リースに使用される VPN に影響を与えるためです。	
VPN-VPN-ID	blob、通常は7バイト (v4、v6、読み取り専用)
仮想プライベート ネットワーク識別子。詳細については、「vpn-name」を参照してください。	
vpn-vrf-name	string（v4、v6、読み取り専用）
VPN の仮想ルーティングおよび転送テーブル識別子。詳細については、「vpn-name」を参照してください。	

応答ディクショナリ

次の表は、応答辞書でいつでも設定できるデータ項目を示しています。DHCPサーバーは、さまざまな時間にそれらを読み取ります。特に指定されていない限り、操作は読み取り/書き込み可能です。

表 *6:* 応答ディクショナリ固有のデータ項目

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
アクティブリースクエリコントロール	int（v4、v6）
リースの送信を制御します(特定の状態の変更に対してのみなど)。値は、0(未指定(サーバーが通知を送信するかどうかを決定する)、1 - 送信 (サーバーが通知を送信する)、および2は送信しない (サーバーは通知を送信しません) です。アクティブリースクエリコントロールは 0 (指定なし) として初期化されます。	
クライアント・アクティブ・リース数	int（v6、読み取り専用）
DHCPv6 クライアントのアクティブなリース数。	
クライアント作成時間	int（v4、v6、読み取り専用）
クライアントの作成時刻。	
client-domain-name	文字列 (v4、読み取り専用)

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
リース内のクライアント情報から、クライアントが使用するドメイン名。DHCP サーバーがスコープで指定されたドメイン名を使用する場合、このドメインが存在しない可能性があります。安定した記憶域の更新直前に DNS 更新の要求をキューに入れる時に読み取ります。	
クライアントの有効期限	int（v4、v6、読み取り専用）
このサーバーによってクライアントに与えられた最大のリース有効期限 (1970 年以降の秒単位)。	
client-host-name	string（v4、読み取り専用）
リース内のクライアント情報から、DHCP サーバーが DNS に入れるホスト名。安定したストレージを更新する直前に DNS 更新の要求をキューに入れたときに読み取り。	
client-id	blob（v4、v6、読み取り専用）
リース内のクライアント情報から、サーバーがクライアントを追跡するために使用したクライアント ID。これは、要求と共に送信されたクライアント ID であるか、または MAC アドレスから内部的に生成されたクライアント ID である可能性があります。DHCPv6 の場合、クライアント識別子オプション値 (クライアントの DUID)。	
client-id-created-from-mac-address	int (v4、読み取り専用)
リース内のクライアント情報から。1 に設定した場合、クライアント IDは MAC アドレスから作成する必要があり、レポートでは使用しないでください。	
client-last-transaction-time	int（v4、v6、読み取り専用）
DHCP サーバーがこのクライアントから最後に聞き取った時間 (秒単位)。	
クライアント制限 ID	blob (v4、読み取り専用)
現在のリースに関連付けられているクライアントの制限 ID。	
client-mac-address	blob（v4、読み取り専用）
リース内のクライアント情報から、要求ディクショナリに関連付けられたクライアントオブジェクトに格納されている MAC アドレス。mac-addressと同じ形式 (および作成元) を持つ。	
クライアント-osタイプ	int（v4）
またはpre-client-lookup拡張ポイントでこれを設定して、要求パケットのクライアントエントリpost-client-lookupを変更します。で読むことcheck-lease-acceptableもできますが、そこで設定することはできません。値を設定するには、最初に要求ディクショナリで os-typepost-packet-decodeを設定する必要があります。	
クライアントオーバーライドクライアント ID	blob（v4、v6、読み取り専用）

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
現在のクライアント ID 値に使用される BLOB。着信パケットのクライアント ID を置き換えます (両方の値はリース状態データベースに保持されます)。	
クライアントオーバーライド-クライアント ID-データ型	string（v4、v6、読み取り専用）
文字列の場合は nstr またはBLOBの場合、クライアントオーバーライド クライアント IDのデータ型を返します。	
クライアントオーバーライド-クライアント ID 文字列	string（v4、v6、読み取り専用）
着信パケットからのすべての client-id を置き換える文字列形式の現在の client-id 値（両方の値がリース ステータス データベースに保持されていても）。getの場合、クライアントオーバーライド クライアント IDが文字列でない場合、バイナリ データは BLOB データとして書式設定され、"string" として返されます。	
client-packet	blob (v4、v6、読み取り専用)
応答パケットのクライアント部分。DHCPv4 の場合、これは完全なパケットです。DHCPv6 の場合、これはクライアント・メッセージです。(パケット全体を取得するには、パケットを参照してください。パケットエンコード後の拡張ポイントからのみ使用できます。	
クライアント再構成キー	文字列 (v6)
DHCPv6 リースのクライアント再構成キー属性値を返します。	
クライアント再構成キー生成時間	string（v6）
DHCPv6 リースのクライアント再構成-鍵生成時間属性値を戻します。	
クライアントリレーアドレス	IPv6 アドレス（v6、読み取り専用）
(最後の) リレーの送信元 IPv6 アドレス。	
クライアントリレーメッセージ	文字列 (v6、読み取り専用)
最後に中継された DHCPv6 メッセージ (クライアント メッセージを除く)。	
クライアントが要求したホスト名	文字列 (v4)
リース内のクライアント情報から、クライアントが DNS 更新のために要求したホスト名。	
クライアント ユーザー定義データ	string（v4、v6、読み取り専用）
ユーザー定義データ環境ディクショナリ データ項目から派生した、クライアントに以前または現在関連付けられている値を返します。つまりcheck-lease-acceptable、拡張ポイントで要求された場合は、lease-state-change以前に関連付けられた値を返します。またはpre-packet-encode拡張ポイントで要求された場合は、post-send-packet現在の値を返します。	

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
クライアント・ベンダー・クラス	string（v4、v6）
DHCPv4 または DHCPv6 リースのクライアント・ベンダー・クラス属性値を戻します。	
クライアントベンダー情報	string（v4、v6）
DHCPv4 または DHCPv6 リースのクライアント・ベンダー情報属性値を戻します。	
クライアント書き込みシーケンス	int（v6、読み取り専用）
クライアント IPv6 要求の書き込みシーケンス。	
クライアント書き込み時刻	int（v6、読み取り専用）
クライアント IPv6 書き込み要求の時刻。	
派生 VPN-ID	int (v4、v6、読み取り専用)
VPN 識別子。	
ドメイン名変更	int（v4）
1 に設定すると、現在のパケットのドメイン名は、DNS 更新で使用されるドメイン名とは異なります。前check-lease-acceptableと前にpre-packet-encode読んでください。	
ダンプパケット	int (v4、v6、書き込み専用)
1 に設定すると、Cisco Prime Network レジストラーは、現在デコードされた DHCP/BOOTP パケットをログ ファイルにダンプします。拡張機能は、その実行の複数のポイントでこのデータ項目に値 1 を配置できます。これは、拡張機能をデバッグするときに役立つ場合があります。	
フェールオーバー ロール	int (v4、v6、読み取り専用)
フェールオーバー サーバーの役割を決定します。フェールオーバー サーバーの役割は、次の 3 つの値のいずれかになります。 • • なし:フェールオーバーが構成されていません。 • メイン/バックアップ:フェイルオーバーが構成され、フェイルオーバー・サーバーの役割	
failover-state	int（v4、v6、読み取り専用）
フェールオーバー サーバーの状態を決定します。フェールオーバーの状態は、正常、パートナーダウン、通信中断、回復、競合の可能性、回復完了、起動、シャットダウン、または一時停止状態にすることができます。フェールオーバーが設定されていない場合、この値は None になります。	

DHCP 拡張ディクショナリ

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
host-name-changed	int（v4）
1 に設定すると、現在のパケットのホスト名は、DNS 更新で使用されるホスト名とは異なります。check-lease-acceptable 後および pre-packet-encode 前に読み取ります。	
ホスト名-イン DNS	int（v4、v6）
1 に設定すると、ホスト名は DNS 内にあります。check-lease-acceptable 後および pre-packet-encode 前に読み取ります。ホスト名が DNS に入った後に書き込まれます。	
リース・バインディング・アイイド	int（v6、読み取り専用）
IPv6 リース バインディング IAID。	
リース バインディング再バインド時間	int（v6、読み取り専用）
IPv6 リース バインディング再バインド時間。	
リースバインディング-更新時間	int（v6、読み取り専用）
IPv6 リース バインディングの更新時間。	
リース バインディング タイプ	string（v6、読み取り専用）
IPv6 リース バインディングの種類: "IA_NA"、"IA_TA"、または "IA_PD"	
リースクライアント予約済み	int（v4、v6、読み取り専用）
リースがクライアント予約の場合は 1 を返し、予約されていない場合は 0 を返します。	
リース作成時間	string（v6、読み取り専用）
IPv6 リースの作成時間。	
リース非アクティブ化	int（v4、v6、読み取り専用）
1 に設定すると、リースが非アクティブであることを報告します。	
リース-DNS-フォワード-バックアップ-サーバーアドレス	IP アドレス (v4、v6、読み取り専用)
リース DNS 転送サーバー アドレスで指定されたサーバーがダウンしている場合に、DHCPv4 および DHCPv6 のリースに対する DNS 更新を受信するバックアップDNS サーバーのアドレス。	
リース-DNS-フォワード-サーバーアドレス	IP アドレス（v4、v6、読み取り専用）
DHCPv4 および DHCPv6 リースの動的 DNS 更新を受信する DNS サーバーのアドレス。	
リース-DNS-フォワード-アップデート	string（v4、v6、読み取り専用）

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
DHCPv4 および DHCPv6 リースの DNS 更新に含める転送ゾーンを決定する更新の構成の名前。更新すべてまたは更新のみ fwdが設定されている場合は TRUE を返します。	
リース DNS 転送ゾーン名	string（v4、v6、読み取り専用）
DNS 更新用のオプションの転送ゾーンの名前。	
リース-DNSリバースバックアップ-サーバーアドレス	IP アドレス（v4、v6、読み取り専用）
リース-DNS-リバースサーバーアドレスで指定されたサーバーがダウンしている場合、DHCPv4 および DHCPv6 リースの DNS 更新を受信するバックアップ DNS サーバーのアドレス。	
リース-dns-リバース-ホストバイト	int（v4、読み取り専用）
リバース ゾーンに使用するリース IP アドレスのバイト数。	
リース-dns-リバース-プレフィックス長	int（v6、読み取り専用）
ip6.arpa 更新の逆ゾーンのプレフィックス長。	
リース-DNS-リバースサーバーアドレス	IP アドレス（v4、v6、読み取り専用）
DHCPv4 および DHCPv6 リースの動的 DNS 更新を受信する DNS サーバー アドレスのアドレス。	
リース-DNSリバース更新	string（v4、v6、読み取り専用）
DHCPv4 および DHCPv6 リースの DNS 更新に含めるリバース ゾーンを決定する更新構成の名前。 update-all または update-fwd-only が設定されている場合は、TRUE を返します。	
リース-dns-リバースゾーン名	string（v4、v6、読み取り専用）
PTR レコードで更新される DNS 逆引き (in-addr.arpa および ip6.arpa) ゾーン。	
リース-fqdn	string（v6、読み取り専用）
サーバーによって DHCPv6 リースに割り当てられた完全修飾ドメイン名 (DNS に正常に入力された場合もあります)。 リース fqdnは、リースの DNS に追加される予定の名前、または実際に追加された名前である可能性があります。ホスト名の dnsが true の場合、実際のリース FQDNは DNS 内にあります。	
リースジャッダー	IP アドレス (v4、読み取り専用)
リース・ジャドル	

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
リース-ipアドレス	IPv4 または IPv6 のアドレスまたはプレフィックス (v4、v6、読み取り専用)
DHCPv4 の場合、クライアントに関連付けられているリースのアドレス。DHCPv6 の場合、現在のコンテキストのリースの IPv6 アドレスまたは IPv6 プレフィックス (アドレスとプレフィックス長) (setObject メソッドを参照)。	
リース優先の有効期間	int（v6、読み取り専用）
IPv6 リースの優先有効期間。	
リースプレフィックス名	string（v6、読み取り専用）
IPv6 リースのプレフィックス名。	
リースリレーエージェント情報	blob（v4）
オプション 82 の内容全体。	
リースリレー-エージェント-回路-ID	blob（v4）
応答のリースに格納されているリレー エージェントの回線 ID にアクセスし、操作します。サブオプション番号 1 を最初のバイトとして指定する必要があります。リースリレーエージェント回線 ID-データ項目を支持して非推奨。	
リースリレー-エージェント-サーキット-IDデータ	blob (v4、廃止されたリースリレーエージェント-サーキット IDの代わりに使用)
応答のリースに格納されたリレー エージェント-回線 ID データにアクセスし、操作します。	
リースリレー-エージェント-デバイス-クラスデータ	blob（v4）
オプション 82 の device-class サブオプションの内容。	
リースリレーエージェント半径属性	blob（v4）
オプション 82 の半径サブオプションの内容。	
リースリレー-エージェント半径クラス	string（v4）
オプション 82 の radius サブオプションのカプセル化された class 属性。	
リースリレー-エージェント-半径-プール名	string（v4）
オプション 82 の radius サブオプションのカプセル化された framed-pool 属性。	
リースリレー-エージェント半径-ユーザー	string（v4）

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
オプション 82 の radius サブオプションのカプセル化された user 属性。	
リースリレー-エージェント-リモート ID	blob（v4)
応答のリースと共に格納されたリレーエージェントリモートIDデータにアクセスし、操作します。サブオプション番号 2 を最初のバイトとして指定する必要があります。リース・リレー・エージェント-リモート ID-データ項目を支持して非推奨。	
リースリレー-エージェント-リモートIDデータ	blob (v4、リースリレー-エージェント-リモート ID項目の代わりに使用)
応答のリースと共に格納されたリレー エージェント-リモート ID データにアクセスし、操作します。	
リースリレー-エージェント-サーバー-id- オーバーライドデータ	IP アドレス（v4）
応答のリースと共に格納されているリレー エージェント-サーバー ID オーバーライドデータにアクセスし、操作します。	
リースリレーエージェント-サブネット- 選択データ	IP アドレス（v4）
応答のリースと共に格納されたリレーエージェントサブネット選択データにアクセスし、操作します。	
リースリレーエージェント加入者 ID	string（v4）
オプション 82 のサブスクライバー ID サブオプションの内容。	
リースリレー-エージェント-vpn-id-データ	blob（v4)
応答のリースと共に格納されているリレー エージェント-vpn-idデータにアクセスし、操作します。	
リース要求の fqdn	string（v6、読み取り専用）
DHCPv6 リースのためにクライアントによって最後に要求された部分的または完全修飾ドメイン名。	
リース要求のプレフィックス長	int（v6、読み取り専用）
記録されたクライアントの要求されたプレフィックス長 (指定されている場合) IA_PD バインディング。クライアントが特定のプレフィックス長の要求を送信しなかった場合は、0 になります。	
リース予約済み	int（v4、v6、読み取り専用）

DHCP 拡張ディクショナリ

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
リースがリース予約されている場合は 1 を返し、リースが予約されていない場合は 0 を返します。	
リース開始時期の状態	int（v4、v6、読み取り専用）
1970年以降の秒数で、このリースが最初に現在の状態に置かれた時間。	
リース状態	string（v4、v6、読み取り専用）
リースの状態(利用可能、提供、リース、期限切れ、利用不可、リリース済、その他利用可能(DHCPv4 のみ)、保留あり(DHCPv4 のみ)、または取り消し(DHCPv6 のみ)	
リース状態の有効期限	int（v4、v6、読み取り専用）
リース状態の有効期限です。	
リースステータス	string（v4、v6、読み取り専用）
「存在しない」、「クライアントによって所有される」、「存在する」を返します。リースが存在するかどうか、および現在のクライアントがリースを所有しているかどうかを判断するために使用されます。"exists" が返された場合、リースは存在しますが、現在の所有者はリースを所有していません (リースに関する限られた情報が利用可能です)。	
リース有効な有効期間	int（v6、読み取り専用）
IPv6 リースの有効期間が有効です。	
リース VPN 説明	string（v4、v6、読み取り専用）
応答のリースと共に格納された VPN の説明。	
リース VPN-ID	int（v4、v6、読み取り専用）
応答のリースと共に格納される VPN の識別子。	
リース VPN-名前	string（v4、v6、読み取り専用）
応答のリースと共に格納された VPN の名前。	
リース VPN-VPN-ID	blob、通常 7 バイト、v4、v6、読み取り専用
応答のリースと共に格納された仮想プライベート ネットワーク (VPN) 識別子。	
リース vpn-vrf-name	string（v4、v6、読み取り専用）
応答のリースと共に格納された VPN の仮想ルーティングと転送テーブル識別子。	
mac-address	blob（v4)

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
クライアント パケットに入っている MAC アドレス。最初のバイトはハードウェアタイプ、2番目のバイトはハードウェアの長さ、残りのバイトはchaddrからの情報です。これは、DHCPパケットのhtype、 hlen、およびchaddrフィールドの便利な集約です。読み取り時には、これらのフィールドから構築されます。書き込まれると、これらのフィールドに配置されます。	
override-client-id	blob（v4、v6、読み取り専用）
現在のクライアント ID 値に使用される BLOB。着信パケットのクライアント ID を置き換えます (両方の値はリース状態データベースに保持されます)。	
オーバーライドクライアント ID データ型	文字列 (v4、v6、読み取り専用)
オーバーライドクライアント IDのデータ型 (文字列の場合は "nstr" 、BLOB の場合は "blob") を返します。	
オーバーライドクライアント ID 文字列	string（v4、v6、読み取り専用）
受信パケットのクライアント ID を置き換える文字列形式の現在のクライアント ID 値 (両方の値はリース状態データベースに保持されます)。 getの場合、オーバーライド クライアント IDが文字列でない場合、バイナリ データは BLOB データとして書式設定され、"string" として返されます。	
パケット	blob (v4、v6、でのみ使用post-packet-decode)
応答パケット。DHCPv4 の場合、これは client-packet と同じです。DHCPv6 では、リレーの場合、フル パケットとなり、リレーでない場合は、client-packet と同じとなります。これは、パケットエンコード後の拡張ポイントからの読み取りまたは書き込みのみにする必要があります。書き込まれると、サーバーは新しいパケットをクライアントに送信します。	
ping-clients	int（v4）
1 に設定すると、この要求のリースを提供する前に ping を実行します。クライアントリースの受け入れ可を判断する直前に読んでください。	
prefix-address	IPv6 プレフィックス (v6、読み取り専用)
プレフィックス アドレス (17 バイト— IPv6 アドレスおよびプレフィックス長)。	
プレフィックス割り当てランダム	int（v6、読み取り専用）
プレフィックスはランダムに割り当てられます。	
プレフィックス割り当て - 最適な方法で割り当て	int（v6、読み取り専用）
最適フィットを介して割り当てられたプレフィックス。	

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
プレフィックス割り当て-経由クライアント要求	int（v6、読み取り専用）
クライアント要求を介して割り当てられたプレフィックス。	
拡張子によるプレフィックス割り当て-割り当て	int（v6、読み取り専用）
拡張機能を介して割り当てられたプレフィックス。	
プレフィックス割り当てインターフェイス経由の識別子	int（v6、読み取り専用）
インターフェイス識別子を介して割り当てられたプレフィックス。	
予約によるプレフィックス割り当て	int（v6、読み取り専用）
予約を介して割り当てられたプレフィックス。	
プレフィックス割り当てグループ	string（v6、読み取り専用）
プレフィックスの割り当てグループ名。	
プレフィックス割り当てグループの優先順位	int（v6、読み取り専用）
プレフィックスの配賦グループの優先順位。	
プレフィックス非アクティブ化	int（v6、読み取り専用）
プレフィックスが非アクティブかどうかを示します。	
プレフィックス- dhcp タイプ	string（v6、読み取り専用）
プレフィックス DHCP タイプ。	
プレフィックスの有効期限	string（v6、読み取り専用）
プレフィックスの有効期限です。	
プレフィックス リンク グループ名	string（v6、読み取り専用）
リンクのリンク グループ名。	
プレフィックス リンク名	string（v6、読み取り専用）
プレフィックスのリンク。	
プレフィックス-リンクタイプ	string（v6、読み取り専用）
リンクの種類 (トポロジ、場所に依存しない、またはユニバーサル)。	

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
prefix-name	string（v6、読み取り専用）
プレフィックスの名前。	
prefix-range	IPv6 アドレス（v6、読み取り専用）
プレフィックスの IPv6 アドレス範囲。	
プレフィックス範囲終了	IPv6 アドレス（v6、読み取り専用）
プレフィックスの範囲の終点 (範囲の開始または範囲の終わりが構成されている場合)。	
プレフィックス範囲の開始	IPv6 アドレス（v6、読み取り専用）
プレフィックスの範囲の開始 (範囲の開始または範囲の終点が構成されている場合)。	
プレフィックス制限から予約へ	int（v6、読み取り専用）
1 に設定すると、プレフィックスの予約制限が有効になります。	
プレフィックス選択タグ	string（v6、読み取り専用）
プレフィックスの選択タグ。	
リレーカウント	int (v6、読み取り専用)
DHCPv6 リレー ホップの数。	
返信-ipアドレス	IPv4 または IPv6 アドレス (v4、v6)
DHCP クライアントに応答するときに使用する IP アドレス。ちょうど後にpre-packet-encode読んでください。で値をpre-packet-encode変更する場合は、その中に配置する IP アドレスは、(ブロードキャスト アドレスでない限り) ARP クエリに応答できるシステムに対するアドレスである必要があります。ユニキャストが有効で、ブロードキャスト フラグが DHCP 要求で設定されていない場合でも、ローカル ARP キャッシュは、DHCP 要求のMAC アドレス pre-packet-encodeへの新しい応答 IP アドレスからのマッピングで設定されません。	
応答ポート	int（v4、v6)
DHCP クライアントに応答するときに使用するポート。pre-packet encode 直後に読み取ります。	
応答ソース	string（v4、v6、読み取り専用）

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
応答のソース (拡張機能を呼び出した主要なアクティビティ)。出力値は、クライアント (受信クライアント パケット)、フェールオーバー (フェールオーバー パートナーからのバインディング更新の受信)、タイムアウト (リースの有効期限または猶予期間終了)、オペレータ (ユーザー インターフェイスからの要求)、1 クライアントあたりのリース (クライアントごとに 1 つのリースを削除する)クライアントは、新しいリースのために古いリースから、不明(上記のどれも) このデータ項目は、要求ディクショナリが存在するかどうかの処理を決定する拡張機能を支援します。(isValid メソッドは、ディクショナリが有効かどうかを判断するためにも使用できます)。	
逆の名前の dns	int（v4、v6)
1 に等しい場合、逆の名前は DNS に含まれています。DNS 操作を初期化する前に読んでください。	
スコープ許可ブート	int（v4、読み取り専用）
1 に設定すると、スコープは BOOTP を許可します。DNS 操作が完了した後に書き込まれます。	
スコープ許可- dhcp	int（v4、読み取り専用）
1 に設定すると、スコープは DHCP を許可します。	
スコープ許可動的ブート	int（v4、読み取り専用）
1 に設定すると、スコープは動的 BOOTP を許可します。	
スコープ利用可能なリース	int（v4、読み取り専用）
現在のスコープで使用可能なリースの数。	
スコープ非アクティブ化	int（v4、読み取り専用）
1 に設定すると、スコープは非アクティブになります。	
スコープ-DNS-フォワード-サーバーアドレス	IP アドレス（v4、読み取り専用）
DNS 転送アドレスに使用する DNS サーバー。	
スコープ-DNS転送ゾーン名	string（v4、読み取り専用）
スコープに構成された転送ゾーン名。	
スコープ-DNS-ホストバイト数	int（v4、読み取り専用）
DNS 更新を処理する DHCP サーバー コードによって使用されるホスト バイト数です。	

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
スコープ-DNS-逆サーバーアドレス	IP アドレス（v4、読み取り専用）
DNS 逆アドレスに使用する DNS サーバー。	
スコープ-DNS-逆ゾーン名	string（v4、読み取り専用）
スコープに構成された逆引きゾーン名。	
scope-name	string（v4、読み取り専用）
DHCP サーバーが処理しているリースを含むスコープの名前。	
スコープネットワーク番号	IP アドレス（v4、読み取り専用）
DHCP サーバーが処理しているリースを含むスコープのネットワーク番号。	
スコープ ping クライアント	int（v4、読み取り専用）
1 に設定すると、現在のリースに関連付けられたスコープは、リースを提供する前に ping 操作をサポートするように構成されています。	
スコープ-プライマリ ネットワーク番号	IP アドレス（v4、読み取り専用）
このプライマリ スコープのネットワーク番号。	
スコープ-プライマリ サブネット マスク	IP アドレス（v4、読み取り専用）
このプライマリ スコープのサブネット マスク。	
スコープの更新のみ	int（v4、読み取り専用）
1 に設定すると、スコープは更新のみになります。	
スコープ更新のみ期限切れ時刻	int（v4、読み取り専用）
1970 年 1 月 1 日以降の絶対時間 (秒単位) で、更新のみのスコープは更新のみ停止する必要があります。	
スコープ制限の予約	int（v4、読み取り専用）
1 に設定すると、スコープの予約制限が有効になります。	
スコープ選択タグ	string（v4、読み取り専用）
スコープ選択基準を含むコンマ区切りの文字列。このデータ項目は、スコープに基づく決定に使用します。	
スコープ-送信-ack-最初	int（v4、読み取り専用）
1 に設定すると、スコープは残りの処理を実行する前に ACK を送信します。	

データ項目	値 (プロトコル: v4=DHCPv4, v6=DHCPv6)
スコープ サブネット マスク	IP アドレス（v4、読み取り専用）
DHCP サーバーが処理しているリースを含むスコープのサブネット マスク。	
スコープ更新- DNS	string（v4、読み取り専用）
前方ゾーンまたは逆ゾーンの DNS 更新。出力値は、1 =すべて更新、2=更新-fwd のみ、3=更新-rev-only、および 0=更新なしです。	
スコープ更新-DNS が有効	ブール値 (v4、読み取り専用)
1 に設定すると、スコープの更新 DNS が転送ゾーンと逆方向ゾーンに対して有効になります。スコープ更新 DNSを優先して非推奨。	
スコープ更新- dns-for-bootp	int（v4、読み取り専用）
1 に設定すると、スコープの更新 DNS が BOOTP に対して有効になります。	
トレース ID	string（v4、v6、読み取り専用）
パケットをトレースするためにシステムが使用する ID。	
トランザクション時間	int（v4、v6、読み取り専用）
要求がデコードされた時間 (1970 年以降の秒単位)。	
VPN 説明	string（v4、v6、読み取り専用）
VPN の説明。	
vpn-name	string（v4、v6、読み取り専用）
VPN の名前。	
VPN-VPN-ID	blob、通常は 7 バイト (v4、v6、読み取り専用)
仮想プライベート ネットワーク（VPN）識別子。	
vpn-vrf-name	string（v4、v6、読み取り専用）
VPN の仮想ルーティングおよび転送テーブル(VRF)識別子。	

拡張ディクショナリ API

このセクションには、Tcl 拡張および共有ライブラリから辞書にアクセスするときに使用するディクショナリ メソッドの呼び出しが含まれています。

TCL 属性ディクショナリ API

属性ディクショナリでは、キーは Cisco Prime Network レジストラー DHCP サーバー設定で定義されている属性の名前に制限されます。値は、その特定の属性の有効な値の文字列表現です。たとえば、IP アドレスはアドレスのドット付き 10 進文字列表現で指定され、列挙値は列挙型の名前で指定されます。つまり、数値は、数値の文字列形式で指定されます。

属性ディクショナリは、キーのインスタンスを複数含めることができるという点で珍しいです。これらのインスタンスは順序付けされ、最初のインスタンスはインデックス 0 になります。属性ディクショナリ メソッドの中には、参照するインスタンスのリスト内の特定のインスタンスまたは位置を示すインデックスを許可するものもあります。

TCL の要求ディクショナリと応答ディクショナリ メソッド

属性ディクショナリでは、コマンドを使用して、ディクショナリ内の値を変更したり、値にアクセスしたりできます。次の表は、要求辞書と応答辞書で使用するコマンドの一覧です。この場合、dict変数を またはrequestresponseとして定義できます。

例については、インストールパス/例/dhcp/tcl/tclextension.tcl.tcl ファイルを参照してください。

表 *7: TCL* の要求ディクショナリと応答ディクショナリ メソッド

方法	構文
get	$dict get 属性[インデックス[bMore]]
ディクショナリから属性の値を返します。ディクショナリに属性が含まれていない場合は、代わりに空の文字列が返されます。インデックス値を含める場合、属性のインデックス番目のインスタンスが返されます。一部の属性は、要求パケットまたは応答パケットに複数回出現する可能性があります。インデックスは、返すインスタンスを選択します。 bMoreを含める場合、getメソッドは、返された属性の後に属性が多い場合は true に設定し、それ以外の場合は FALSE に設定します。これを使用して、 get をもう一度呼び出してその属性の他のインスタンスを取得するかどうかを決定します。	
getOption	$dict getOption arg型[arg-data]
オプションのデータを文字列として取得します。arg 表 8 : Tcl の arg 型と obj 型の値 （47 ページ）型の値については、「」を参照してください。次の引数が数値の場合は、数値、それ以外の場合は名前と見なされます。この関数は文字列へのポインタを返すことに注意してください。オプションが存在しない、またはオプションの長さが 0 の場合は、ポインタは長さ 0 になります。サンプルの使用方法については、「ベンダー クラス オプション データの処理 （66 ページ）」を参照してください。	
isValid isV4 isV6	$ディクストisValid $ディクトisV4 $isV6

DHCP 拡張ディクショナリ

方法	構文
(isValid渡されたディクショナリに応じて) 要求または応答がある場合、メソッドは TRUE を返します。それ以外の場合はFALSE。などの拡張機能はlease-state-change、ディクショナリが使用可能かどうかを判断するために、このメソッドを使用できます。 このisV4拡張が DHCPv4 パケットに対して呼び出されている場合、メソッドは TRUE を返します。それ以外の場合は FALSE。ルーチンからこのメソッドをinit-entry呼び出すと、FALSE が返されます。 このisV6拡張が DHCPv6 パケットに対して呼び出されている場合、このメソッドは TRUE を返します。それ以外の場合は FALSE。init-entry ルーチンからこのメソッドを呼び出すと、FALSE が返されます。	
log	$dict log level message ...
DHCP サーバーログシステムにメッセージを書き込みます。レベルは、LOG_ERROR、LOG_WARNING、またはLOG_INFOにする必要があります。残りの引数は連結され、指定されたレベルでロギングシステムに送信されます。 （注） サーバーはログ ファイルをこれらのレベルでログに記録するメッセージでフラッシュするので、LOG_ERROR レベルとLOG_WARNINGレベルを慎重に使用します。頻繁に発生しそうなメッセージ（クライアント要求など）に対してこれらのレベルを使用すると、ディスク I/O パフォーマンスに重大な影響を及ぼす可能性があります。	
moveToOption	$dict moveToOption arg型[arg-data] ..
後続の get、put、remove オプション操作のコンテキストを設定します。arg-type 値については、表 8 : Tcl の arg 型と obj 型の値 （47 ページ） を参照してください。オプションが削除されるとコンテキストが無効になる可能性があることに注意してください (などremoveOption) 。	
put	$dict put 属性値[インデックス]
ディクショナリ内の属性に値を関連付けます。インデックスを省略するか、特殊な値REPLACE に設定すると、属性の既存のインスタンスが単一の値に置き換えられます。インデックス値を特殊な値APPEND として指定すると、属性のインスタンスリストの末尾に属性の新しいインスタンスが追加されます。インデックス値を数値として含める場合は、指定された位置に属性の新しいインスタンスが挿入されます。インデックス値を特別な値AUGMENT に設定した場合、属性が追加されるのは、属性が存在しない場合のみです。	
putOption	$dict putOptionデータのarg型[arg-data] ..
オプションとそのデータを追加する、またはオプションのデータを変更します。arg-type 値については、表 8 : Tcl の arg 型と obj 型の値 （47 ページ） を参照してください。サンプルの使用方法については、「ベンダー クラス オプション データの処理 （66 ページ）」 を参照してください。	
remove	$dict remove 属性[インデックス]

方法	構文
ディクショナリから属性を削除します。インデックスを省略するか、REMOVE_ALL特殊な値に設定すると、属性の既存のインスタンスが削除されます。インデックスを数値として含める場合、指定された位置にある属性のインスタンスが削除されます。このメソッドは、ディクショナリがそのインデックスに属性を含んでいない場合でも、常に 1 を返します。	
removeOption	$dict removeOption arg-type [arg-data] ...
オプションを削除します。arg-type 値については、表 8 : Tcl の arg 型と obj 型の値 （47 ページ） を参照してください。サンプルの使用方法については、「ベンダー クラス オプション データの処理 （66 ページ）」を参照してください。	
setObject	$setObjectオブジェクト型[データ]
(DHCPv6 のみ)。get、put や removeメソッドのオブジェクトを設定し、新しいオプション メソッドが動作するメッセージを変更します。obj表 8 : Tcl の arg 型と obj 型の値 （47 ページ）型の値については、「」を参照してください。DHCPv6 拡張機能は、主にこのメソッドを使用して、クライアントとリンクで使用可能なリースおよびプレフィックスにアクセスしたり、リレーパケットからメッセージヘッダーフィールドまたはオプションを取得したりします。1 つのリースとスコープが応答に関連付けられている DHCPv4 とは異なり、DHCPv6 応答には複数のリースとプレフィックスが含まれる場合があります。オブジェクトが存在する場合は TRUE を返します。それ以外の場合は FALSE。サンプルの使用方法については、オブジェクト データの処理 （67 ページ）を参照してください。 （注） 現在のクライアントに関連付けられていないリースの場合、最小限の情報しか使用できません。	
trace	$dict trace level message ...
DHCP サーバー パケット トレース システム内のメッセージを返します。レベル 0 では、トレースは行われません。レベル 1 では、サーバーがパケットを受信して応答を送信した場合にのみトレースされます。レベル4では、すべてをトレースします。残りの引数は連結され、指定されたレベルでトレース システムに送信されます。デフォルトのトレースは、DHCP サーバー拡張トレースレベル属性を使用して設定されます。	

表 8 : Tcl の arg 型と obj 型の値

arg 型または obj 型	説明
enterprise number/name	オプションまたはサブオプションのオプション定義セットのエンタープライズ ID 番号または名前。
home	コンテキストが現在のクライアントまたはリレー メッセージの "top" にリセットされることを要求します。
index番号/キーワード	操作対象の配列インデックスの番号またはキーワード (置換、追加、拡張、生、またはremove_all)。

arg 型または obj 型	説明
index-count	オプションの配列インデックスエントリの数を返します。
instance number	オプションのインスタンス番号 (主に DHCPv6 に使用)。
instance-count	オプションが表示された回数を返します (0 の場合、オプションは存在しません)。
moretcl 変数名	オプション データに配列インデックス エントリが存在するかどうかに応じて、TRUE または FALSE に設定されている Tcl 変数の名前。
move-to	コンテキストをオプションに設定することを要求します。
option number/name	操作するオプション番号または名前。
parent	コンテキストを 1 つ上に移動することを要求します。
suboption number/name	操作するサブオプション番号または名前。
vendor name	オプションまたはサブオプションのオプション定義セットのベンダー名。
lease initial | index | address | prefix	と共setObjectに使用すると、応答ディクショナリ内のリース、バインディング、およびプレフィックス データ項目のコンテキストを、指定されたリースに設定します。キーワードinitialは、拡張機能が呼び出されたときの元のコンテキストを復元することを要求します。インデックスは、番号付きリース(0から始まる)が設定され、クライアントのすべてのリースを反復処理するために使用できることを要求します。アドレスまたはプレフィックスは、そのアドレスまたはプレフィックスを持つリースが設定されることを要求します (存在する場合)。
message initial | number	と共setObjectに使用すると、メッセージ データ項目のコンテキストと、要求ディクショナリまたは応答ディクショナリ内のオプションを、指定されたメッセージに設定します。キーワードinitialは、クライアント メッセージにコンテキストを設定します。この番号は、リレー メッセージにコンテキストを設定し、0はクライアントに最も近いリレーを指定します。

arg 型または obj 型	説明
prefix initial |インデックス|住所|プレフィックス|名前	と共setObjectに使用すると、応答ディクショナリ内のプレフィックスデータ項目のコンテキストを、指定されたプレフィックスに設定します。initial キーワードは、拡張が呼び出されたときの元のコンテキストが復元されるよう要求します。インデックスは、番号付きプレフィックス(0 から始まる)が設定され、リンク上のクライアントのすべてのプレフィックスを反復処理するために使用できます。アドレスまたはプレフィックスは、アドレスまたはプレフィックスのプレフィックスが設定されることを要求します(見つかった場合)。名前付きプレフィックスが見つかっていることを要求します。現在のリンクのプレフィックスのみが使用できることに注意してください。

TCL 環境ディクショナリ メソッド

次の表は、環境ディクショナリで使用するコマンドを示しています。この場合、次の手順の例のように、dict変数を としてenviron定義できます。

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

表 *9 : TCL* 環境ディクショナリ メソッド

方法	構文
clear	$dict clear
ディクショナリからすべてのエントリを削除します。	
containsKey	$ディクト containsKeyキー
ディクショナリにキーが含まれている場合は 1 を返し、それ以外の場合は 0 を返します。	
firstKey	$dict firstKey
ディクショナリの最初のキーの名前を返します。キーは名前順に格納されません。キーが存在しない場合は、空の文字列を返します。	
get	$ディクト getキー
ディクショナリからキーの値を返します。キーが存在しない場合は、空の文字列を返します。	
isEmpty	$dict isEmpty
ディクショナリにエントリがない場合は 1 を返し、それ以外の場合は 0 を返します。	
log	$ディクスト logレベルメッセージ..

方法	構文
DHCP サーバー ロギング システムのメッセージを返します。レベルは、LOG_ERROR、LOG_WARNING、またはLOG_INFOのいずれかでなければなりません。残りの引数は連結され、指定されたレベルでロギングシステムに送信されます。 （注） サーバーはログ ファイルをこれらのレベルでログに記録するメッセージでフラッシュするので、LOG_ERROR レベルとLOG_WARNINGレベルを慎重に使用します。頻繁に発生しそうなメッセージ（クライアント要求など）に対してこれらのレベルを使用すると、ディスク I/O パフォーマンスに重大な影響を及ぼす可能性があります。	
nextKey	$dict nextKey
firstKey または nextKey への最後のコールで返されるキーに続くディクショナリ内の次のキーの名前を返します。キーが存在しない場合は、空の文字列を返します。	
put	$ディクト putキー値
キーに値を関連付けて、キーの既存のインスタンスを新しい値に置き換えます。	
remove	$ディクト removeキー
ディクショナリからキーを削除します。ディクショナリにキーが含まれていなくても、常に 1 を返します。	
size	$dict size
ディクショナリ内のエントリ数を返します。	
trace	$dict trace level message ...
DHCP サーバー パケット トレース システム内のメッセージを返します。レベル 0 では、トレースは行われません。レベル 1 では、サーバーがパケットを受信して応答を送信した場合にのみトレースされます。レベル4では、すべてをトレースします。残りの引数は連結され、指定されたレベルでトレース システムに送信されます。デフォルトのトレースは、DHCP サーバー拡張トレースレベル属性を使用して設定されます。	

DEX 属性ディクショナリ API

C/C++ 用の DEX 拡張機能を記述する場合、属性名文字列表現またはタイプ (属性を定義するバイトシーケンス) としてキーを指定できます。つまり、これらのアクセス方法の中には、キーまたは値の文字列または型の組み合わせである 4 つの異なるバリエーションがあります。

基本的な DEX 拡張の例は次のようになります。

```
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;
}
```

例については、インストールパス/例/dhcp/デックス/デックスエクステンション.cファイルまたはそのディレクトリ内の他のファイルを参照してください。

DEX の要求ディクショナリと応答ディクショナリ メソッド

DEX 属性ディクショナリでは、メソッドと呼ばれるアクティブなコマンドを使用して、値の変更やアクセスを行うことができます。次の表は、要求ディクショナリと応答ディクショナリで使用するメソッドを示しています。この場合、pDict変数を またはpRequestpResponseとして定義できます。

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

pszAttributeは、const char *アプリケーションがアクセスする属性名へのポインターです。pszValueは、データを表const char *す文字列へのポインターです (getメソッドに対して返され、putメソッドに格納されます)。対応するiObjectType、iObjArgType、およびiArgTypeの各テーブルを参照してください。

ヒント

get、put、Option、Bytes、および OptionBytes メソッドの違い （58 ページ）とget、put、remove、および ByType メソッドの違い （58 ページ）も参照してください。

表 *10 : DEX* の要求ディクショナリと応答ディクショナリ メソッド

方法	構文
allocateMemory	void *pDict->allocateMemory(dex_AttributeDictionary_t *pDict, unsigned int iSize)
この要求の有効期間だけ保持される拡張機能にメモリを割り当てます。	
get	constchar pDict pDict *　->get(dex_AttributeDictionary_t * , pszAttribute iIndex pbもっとconstchar*,abool_t*),int
ディクショナリから属性のiIndex edインスタンスの値を返します。ディクショナリに属性(またはその多くのインスタンス)が含まれていない場合は、代わりに空の文字列が返されます。pbMoreが 0 以外getの場合、返された属性のインスタンスが多い場合はpbMoreを TRUE に設定し、それ以外の場合は FALSE に設定します。この機能を使用して、属性の他のgetインスタンスを取得するために、 を別の呼び出しを行うかどうかを判断します。	
getBytes	const abytes_t *pDict->getBytes(dex_AttributeDictionary_t *pDict, const char *pszAttribute, int iIndex, abool_t *pbMore)

DHCP 拡張ディクショナリ

方法	構文
ディクショナリから属性のiIndex ed インスタンスの値をバイトシーケンスとして返します。ディクショナリに属性 (またはその多くのインスタンス) が含まれていない場合は、代わりに 0 を返します。pbMoreが 0 以外getBytesの場合、返された属性のインスタンスが多い場合は TRUE に設定され、それ以外の場合は FALSE に設定されます。この機能を使用して、属性の他のgetBytesインスタンスを取得するために、 を別の呼び出しを行うかどうかを判断します。	
getBytesByType	const abytes_t *pDict-> getBytesByType(dex_AttributeDictionary_t *pDict,const abytes_t *pszAttribute, int iIndex, abool_t *pbMore)
ディクショナリから属性のiIndex ed インスタンスの値をバイトシーケンスとして返します。ディクショナリに属性 (またはその多くのインスタンス) が含まれていない場合は、代わりに 0 が返されます。pbMoreが 0 以外の場合、返された属性のインスタンスが多い場合は TRUE を指定し、それ以外の場合は FALSE に設定します。この機能を使用して、属性の他のgetインスタンスを取得するために、 を別の呼び出しを行うかどうかを判断します。	
getByType	const char *pDict->getByType(dex_AttributeDictionary_t *pDict, const abytes_t *pszAttribute, int iIndex, abool_t *pbMore)
ディクショナリから属性のiIndex edインスタンスの値を返します。ディクショナリに属性(またはその多くのインスタンス) が含まれていない場合は、代わりに空の文字列を返します。pbMore が 0 以外の場合、getByType メソッドは、インスタンスが返された後にさらにインスタンスがある場合は pbMore に TRUE を設定し、それ以外の場合は FALSE を設定します。これを使用して、他のインスタンスを取得するためにgetByType別の呼び出しを行うかどうかを判断します。	
getOption	constchar * pディ*クスト iArgType .. getOption(dex_AttributeDictionary_t , , int)
オプションのデータを文字列として取得します。この関数は、常に文字列へのポインターを返しますが、オプションが存在しない場合や長さがゼロである場合は長さがゼロになる可能性があります。オプションが存在するかどうかを確認するには、DEX_INSTANCE_COUNTをgetOptionBytes 使用または指定します。	
getOptionBytes	const abytes_t * getOptionBytes(dex_AttributeDictionary_t *pDict , int iArgType, ...)
オプションのデータをバイトシーケンスとして取得します。この関数は、オプションが存在しない場合は null ポインターを返しabytes_t、オプションが存在するが長さが 0 バイトの場合は長さ 0 のバッファーを持つポインターを返します。	
getType	const abytes_t * pディプト***, const char****->getType(dex_AttributeDictionary_t*pszAttribute)
属性名が構成済みの属性と一致する場合は、属性を定義するバイト シーケンスへのポインターを返します。	

DHCP 拡張ディクショナリ

方法	構文
isValidisV4isV6	abool_t isValid(dex_AttributeDictionary_t *pディクスト ***)abool_t isV6(dex_AttributeDictionary_t*** *)abool_t isV4(dex_AttributeDictionary_t *pDict pDict)
(isValid渡されたディクショナリに応じて) 要求または応答がある場合、メソッドは TRUE を返します。それ以外の場合はFALSE。などの拡張機能はlease-state-change、ディクショナリが使用可能かどうかを判断するために、このメソッドを使用できます。 このisV4拡張が DHCPv4 パケットに対して呼び出されている場合、メソッドは TRUE を返します。それ以外の場合は FALSE。ルーチンからこのメソッドをinit-entry呼び出すと、FALSE が返されます。 このisV6拡張が DHCPv6 パケットに対して呼び出されている場合、このメソッドは TRUE を返します。それ以外の場合は FALSE。init-entry ルーチンからこのメソッドを呼び出すと、FALSE が返されます。	
log	abool_tpDict ->log(dex_AttributeDictionary_t * eLevel , pszFormat .., const char * , int)
DHCP サーバー ログ システムでメッセージを返します。eLevelは、DEX_LOG_ERROR、DEX_LOG_WARNING、またはDEX_LOG_INFOのいずれかでなければなりません。pszFormat は printf スタイルの書式指定文字列として扱われ、残りの引数と共に書式が設定され、指定されたレベルでログ 記録システムに送信されます。 （注） DEX_LOG_ERROR レベルとDEX_LOG_WARNINGレベルは、サーバーがログ ファイルにログ ファイルをフラッシュし、ログ レベルでログに記録されるため、慎重に使用します。これらのレベルを頻繁に発生する可能性のあるメッセージ (クライアント要求など) に使用すると、ディスク I/O パフォーマンスに重大な影響を与える可能性があります。	
moveToOption	abool_t moveToOption(dex_AttributeDictionary_t *pDict, int iArgType, ...)
後続getの 、 、 put、removeおよびオプションの操作のコンテキストを設定します。オプションが削除された場合 (例えば、 を使用removeOption) するとコンテキストが無効になる可能性があることに注意してください。	
put	abool_tpディ->put(dex_AttributeDictionary_tプト*・プディ,プトconst・プcharッツ*・プロパティ,const一・サイ・インデックスchar*,int)

方法	構文
サーバー構成のpszAttributeの定義に従って、pszValueをバイトシーケンスに変換します。そのバイト シーケンスをディクショナリ内の属性に関連付けます。iIndexが特殊な値DEX_REPLACE場合、属性の既存のインスタンスを 1 つの値に置き換えます。特殊な値がDEX_APPEND場合、属性の新しいインスタンスがリストに追加されます。特殊値がDEX_AUGMENT場合、属性が存在しない場合にのみ属性を設定します。それ以外の場合は、指定された位置に新しいインスタンスを挿入します。属性名が構成されている属性と一致しない場合、または値を有効な値に変換できなかった場合は TRUE を返します。	
putBytes	abool_tプディ->putBytes(dex_AttributeDictionary_tプト*・プディ,プトconst・プcharッツ*・プロパティ,const一・サイ・インデックスabytes_t*,int)
ディクショナリ内のpszAttributeにpszValueを関連付けます。iIndexがDEX_REPLACE特殊な値である場合は、属性の既存のインスタンスを 1 つの新しい値に置き換えます。特殊な値がDEX_APPEND場合、属性の新しいインスタンスをリストに追加します。特殊値がDEX_AUGMENT場合、属性が存在しない場合にのみ属性を設定します。それ以外の場合は、指定された位置に新しいインスタンスを挿入します。属性名が構成済みの属性名と一致しない場合は TRUE を返します。	
putBytesByType	abool_tプディ->putBytesByType(dex_AttributeDictionary_tプト*・プディ,プトconst・プabytes_tッツ*・プロパティ,const一・サイ・インデックスabytes_t*,int)
ディクショナリ内のpszAttributeにpszValueを関連付けます。iIndexがDEX_REPLACE特殊な値である場合、属性の既存のインスタンスを新しい値に置き換えます。特殊な値がDEX_APPEND場合、属性の新しいインスタンスをリストに追加します。特殊値がDEX_AUGMENT場合、属性が存在しない場合にのみ属性を設定します。それ以外の場合は、指定された位置に属性の新しいインスタンスを挿入します。	
putByType	abool_tプディ->putByType(dex_AttributeDictionary_tプト*・プディ,プトconst・プabytes_tッツ*・プロパティ,const一・サイ・インデックスchar*,int)
サーバー構成のpszAttributeの定義に従って、pszValueをバイトシーケンスに変換します。そのバイト シーケンスをディクショナリ内の属性に関連付けます。iIndexがDEX_REPLACE特殊な値である場合は、属性の既存のインスタンスを 1 つの新しい値に置き換えます。特殊な値がDEX_APPEND場合、属性の新しいインスタンスをリストに追加します。特殊値がDEX_AUGMENT場合、属性が存在しない場合にのみ属性を設定します。それ以外の場合は、指定された位置に新しいインスタンスを挿入します。	
putOption	abool_tputOption(dex_AttributeDictionary_tを*指定します。, const char * , int ,)

方法	構文
オプションとそのデータを追加するか、オプションのデータを変更します。	
putOptionBytes	abool_t putOptionBytes(dex_AttributeDictionary_t *pディクスト**, *int***, const abytes_t *pValueiArgType.. ,)
オプションとそのデータを追加する、またはオプションのデータを変更します。	
remove	abool_tpDict ->remove(dex_AttributeDictionary_t * pszAttribute, int iIndex, const char *)
ディクショナリから属性を削除します。iIndexがDEX_REMOVE_ALL特殊な値である場合は、属性の既存のインスタンスを削除します。それ以外の場合は、指定された位置にあるインスタンスを削除します。属性名が構成されている属性と一致しない場合は、ディクショナリがそのインデックスに属性を含んでいない場合でも TRUE を返します。	
removeByType	abool_tpDict ->removeByType(dex_AttributeDictionary_t * pszAttribute, int iIndex, const abytes_t *)
ディクショナリから属性を削除します。iIndexがDEX_REMOVE_ALL値の場合は、属性の既存のインスタンスを削除します。それ以外の場合は、指定された位置でインスタンスを削除します。ディクショナリにインデックスの属性が含まれていない場合でも、常に TRUE を返します。	
removeOption	abool_tremoveOption(iArgType 、.. dex_AttributeDictionary * , int)
オプションを削除します。DEX_INDEXを省略すると、DEX_REMOVE_ALLのDEX_INDEXが想定されます (これはオプション全体を削除します)。	
setObject	abool_tをsetObject(**指定します。** dex_AttributeDictionary_t * int int ）
get、 、 およびputremoveメソッドのオブジェクトを設定し、新しいオプション メソッドが動作するメッセージを変更します。DHCPv6 拡張機能は、主にこのメソッドを使用して、クライアントとリンクで使用可能なリースおよびプレフィックスにアクセスしたり、リレー パケットからメッセージ ヘッダー フィールドまたはオプションを取得したりします。1 つのリースとスコープが応答に関連付けられている DHCPv4 とは異なり、DHCPv6 応答には複数のリースとプレフィックスが含まれる場合があります。オブジェクトが存在する場合はTRUEを返します。それ以外の場合はFALSE。サンプルの使用方法については、オブジェクト データの処理 （67 ページ）を参照してください。 （注） 現在のクライアントに関連付けられていないリースの場合、最小限の情報しか使用できません。	
trace	abool_tpDict ->trace(dex_AttributeDictionary_t * iLevel , pszFormat .., const char * , int)

方法	構文
DHCP サーバー パケット トレース システム内のメッセージを返します。レベル 0 では、トレースは行われません。レベル 1 では、サーバーがパケットを受信して応答を送信した場合にのみトレースされます。レベル4では、すべてをトレースします。残りの引数は連結され、指定されたレベルでトレース システムに送信されます。デフォルトのトレースは、DHCP サーバー拡張トレースレベル属性を使用して設定されます。	

DEX 環境ディクショナリ メソッド

環境ディクショナリでは、メソッドと呼ばれるアクティブなコマンドを使用して、辞書の値を変更したりアクセスしたりできます。次の表は、環境ディクショナリで使用するメソッドを示しています。この場合、pDict変数をpEnvironとして定義できます。

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

表 *11 : DEX* 環境ディクショナリ メソッド

方法	構文
allocateMemory	pディクスト ->allocateMemory(dex_EnvironmentDictionary_t*iSize, unsigned int void *)
この要求の有効期間だけ保持される拡張機能のメモリを割り当てます。	
clear	void pディクスト->clear(dex_EnvironmentDictionary_t * pDict)
ディクショナリからすべてのエントリを削除します。	
containsKey	abool_tpディ ->containsKey(dex_EnvironmentDictionary_tプト *pszKey, const char *)
ディクショナリにキーが含まれている場合は TRUE を返します。	
firstKey	const char *pDict->firstKey(dex_EnvironmentDictionary_t *pDict)
ディクショナリの最初のキーの名前を返します。キーは名前順に格納されません。キーが存在しない場合は、0 を返します。	
get	const char *pDict->get(dex_EnvironmentDictionary_t *pDict, const char *pszKey)
ディクショナリからキーの値を返します。キーが存在しない場合は、空の文字列を返します。	

DHCP 拡張ディクショナリ

方法	構文
isEmpty	abool_t pディクスト->isEmpty(dex_EnvironmentDictionary_t * pDict)
ディクショナリが 0 個のエントリを持つ場合は TRUE、それ以外の場合は FALSE を返します。	
log	abool_tpDict ->log(dex_EnvironmentDictionary_t * eLevel , pszFormat .., const char * , int)
DHCP サーバー ログ システムでメッセージを返します。eLevelは、DEX_LOG_ERROR、DEX_LOG_WARNING、またはDEX_LOG_INFOのいずれかでなければなりません。pszFormat は printf スタイルの書式指定文字列として扱われ、残りの引数と共に書式が設定され、指定されたレベルでログ 記録システムに送信されます。 （注） DEX_LOG_ERROR レベルとDEX_LOG_WARNINGレベルは、サーバーがログ ファイルにログ ファイルをフラッシュし、ログ レベルでログに記録されるため、慎重に使用します。これらのレベルを頻繁に発生する可能性のあるメッセージ (クライアント要求など) に使用すると、ディスク I/O パフォーマンスに重大な影響を与える可能性があります。	
nextKey	const char *pDict->nextKey(dex_EnvironmentDictionary_t *pDict)
最後の呼び出しで返されたキーの後に続くディクショナリ内の次のキーfirstKeyのnextKey名前を返します。キーが存在しない場合は、0 を返します。	
put	abool_tpDict ->put(dex_EnvironmentDictionary_t * pZKey, constpszValue char* , const char *)
キーの既存のインスタンスを新しい値に置き換えて、キーに値を関連付けます。	
remove	abool_t pDict->remove(dex_EnvironmentDictionary_t *pDict, const char *pszKey)
ディクショナリからキーと関連付けられた値を削除します。ディクショナリにキーが含まれていなくても、常に TRUE を返します。	
size	int pディクスト->size(dex_EnvironmentDictionary_t * pDict)
ディクショナリ内のエントリ数を返します。	
trace	abool_tpDict ->trace(dex_EnvironmentDictionary_t * iLevel , pszFormat .., const char * , int)

方法	構文
DHCP サーバー パケット トレース システム内のメッセージを返します。レベル 0 では、トレースは行われません。レベル 1 では、サーバーがパケットを受信して応答を送信した場合にのみトレースされます。レベル4では、すべてをトレースします。残りの引数は連結され、指定されたレベルでトレース システムに送信されます。デフォルトのトレースは、DHCP サーバー拡張トレースレベル属性を使用して設定されます。	

get、put、Option、Bytes、および OptionBytes メソッドの違い

次の DEX 拡張メソッドには、違いがあります。

- get および put
- getOption および putOption
- getBytes および putBytes
- getOptionBytes および putOptionBytes

メソッドgetとgetOptionメソッドは、文字列として書式設定された要求された情報を返します。サーバーは、ディクショナリ項目の予期されるデータ型に応じて、データを文字列に変換します。データ型が不明な場合、サーバーはデータを BLOB 文字列形式で返します。

メソッドgetBytesはgetOptionBytes、要求された情報を生のバイト (バッファーへのポインターとそのバッファーのサイズ) として返します。サーバーはこのバッファを読み取るだけで、オプションのデータだけが含まれている必要があります (たとえば、null 終端文字は追加されていません)。

putメソッドとputOptionメソッドは、データが書式設定された文字列として書き込まれると想定しています。サーバーは、ディクショナリ項目の予期されるデータ型に応じて、文字列からデータを変換します。データ型が不明な場合は、BLOB 文字列形式であることが想定されます。

サーバーはputBytes、未処理のバイトをメソッドputOptionBytes(バッファーへのポインターとそのバッファーのサイズ) に渡します。サーバーは、これらのバイトのみを読み取ります。

get、put、remove、および ByType メソッドの違い

次の DEX 拡張メソッドの間には違いがあります。

- get、put、および remove
- getByType、putByType、および removeByType

サーバーはget、 、put、およびremoveメソッドに、目的のデータ項目の名前を文字列として渡します。この場合、サーバーは文字列を内部データ テーブルにマップする必要があります。

サーバーは、文字列のgetByTypegetTypeメソッドputByTypeをremoveByType呼び出すことによって、サーバーが以前に取得した内部データ テーブル参照を渡します (拡張 init-entry など)。これにより、拡張機能の処理が高速化され、高いパフォーマンスを必要とするアプリケーションで重要な場合があります。

（注） getTypeメソッドが参照する内部データ テーブルは、要求または応答ディクショナリに対して要求されたかどうかにかかわらず同じです。同じデータ項目名にgetType対して、各ディクショナリで個別の呼び出しを行う必要はありません。

表 *12* : 値

オブジェクトタイプ	説明
一般定義: コンテキストを変更するオブジェクト。	
DEX_LEASE	リース (およびプレフィックス) コンテキストを変更します。応答ディクショナリのみ。を使用して次のコマンドを実行します。 DEX_BY_IPV6ADDRESS DEX_BY_IPV6PREFIX DEX_BY_INSTANCE DEX_INITIAL
DEX_MESSAGE	メッセージコンテキストをリレーメッセージまたはクライアントメッセージに変更します。要求辞書と応答辞書。次の iObjArgType を使用できます。 DEX_INITIAL DEX_RELAY DEX_BY_NUMBER
DEX_PREFIX	プレフィックスコンテキストを変更しますが、リースコンテキストは変更しません。応答ディクショナリのみ。を使用して次のコマンドを実行します。 DEX_BY_IPV6ADDRESS DEX_BY_IPV6PREFIX DEX_BY_INSTANCE DEX_BY_NAME DEX_INITIAL

表 *13* : 値

型	説明
一般定義: コンテキストが変更される方法。	

型	説明
DEX_BY_INSTANCE	DEX_LEASEまたは DEX_PREFIX iObjectTypeで使用します。インスタンス番号intを指定するために次の値を必要とします (0 から始まります)。利用可能なすべてのオブジェクトのリストを順を確認するために使用されますが、現在の要求または応答に適用可能なオブジェクトのリストを通してのみ使用 DEX_LEASEされます。DEX_PREFIXの場合、現在のリンクのプレフィックス (存在する場合)。DEX_RELAYの同義語であるDEX_MESSAGEで使用されます。
DEX_BY_IPV6ADDRESS	DEX_LEASEおよび DEX_PREFIX iObjectTypeでのみ使用されます。16 バイトのアドレスを指定するために、次のアドレスが必要です。const unsigned char *
DEX_BY_IPV6PREFIX	DEX_LEASE または DEX_PREFIX の iObjectType で使用します。17 バイトのプレフィックス バッファー (16 バイトのアドレスの後に 1 バイトのプレフィックス長) を指定する必要があります。 const unsigned char *
DEX_BY_NAME	DEX_PREFIX iObjectTypeと共にのみ使用されます。次の項目const char *を実行して、目的のオブジェクトの名前を指定する必要があります。
DEX_INITIAL	要求または応答の元のコンテキストに戻し、追加の引数はありません。エクステンションが最初に呼び出されたときのリースとプレフィックス (DEX_LEASE)、プレフィックス (DEX_PREFIX)、またはメッセージ (DEX_MESSAGE) を設定します(なしの場合があります)。
DEX_RELAY	iObjectTypeでのみ使用DEX_MESSAGE。リレーをint指定するために、次のことが必要です(0はクライアントに最も近いリレーを指定します)。メッセージコンテキストをクライアントに戻すには、setObject(pDict, DEX_MESSAGE, DEX_INITIAL)を使用します。

iArgType	説明
一般定義: コンテキストに従うアクションと引数。呼び出しには、iArgTypeインスタンスの数に任意の数を指定できます。	
DEX_ARG_ARRAY	引数リストを指定する代わりに、dex_OptionsArgs_t次の配列へのポインタを必要とします。各dex_OptionsArgs_t構造体には、次の 2 つのフィールドがあります。 • iArgType —このテーブルのiArgType DEX 値の 1 つ。 • pData—データ (整数)、データへのポインタ (文字列やその他のデータ型の場合)、または無視(iArgTypeが引数を取らない場合)。 サーバーが (引数リストまたは配列dex_OptionsArgs_t内の) DEX_ARG_ARRAYを検出すると、元のリスト内の後続の引数は無視されます。
DEX_END	（注） 必須で、追加の引数を持たない、引数リストの末尾をマークします。
DEX_ENTERPRISE_NAME	次に、オプション定義セット名を指定する必要があり、そこからサーバーがエンタープライズ ID を抽出してベンダー・オプション・データを取得します。 const char * ベンダー識別オプションに対してのみ有効です。ベンダーオプション定義セットが存在している必要があります。
DEX_ENTERPRISE_ID	ベンダーのintエンタープライズ ID を指定するには、次の手順を実行する必要があります。
DEX_HOME	コンテキストをクライアントまたはリレーメッセージ オプションに戻します。追加の引数はありません。常に成功を返します。使用する場合は、最初のiArgTypeである必要があります。有効なのはgetOption、 getOptionBytes、moveToOption 、および メソッドだけです。

iArgType	説明
DEX_INDEX	オプションintデータのインデックスを指定する必要があります(データの配列が処理される場合)。省略した場合、インデックス 0 は、DEX_REMOVE_ALLremoveOptionを除いてと見なされます。オプションデータ全体を取得、配置、または削除するには、特別な値DEX_RAWを使用します。ただし、DHCPv4ベンダー識別オプション(RFC 3925およびRFC 4243) では、DEX_RAWは 1 つのベンダー (インスタンスまたはエンタープライズIDに基づく) のデータのみを返し、オプション全体のデータを返しません。 特殊値DEX_RAWは、オプション(またはサブオプション) データ全体にアクセスします。データ型のデータ型と繰り返しカウントの観点からオプション定義で指定するデータに関係なく、データへの一貫したアクセスを提供します。データをデコードする汎用拡張機能に推奨されます。 DEX_REPLACE (値を置換する)、DEX_APPEND (endputOptionに追加putOptionBytesput)、およびDEX_AUGMENT (値が存在しない場合は追加) とメソッドを使用し、putByTypeメソッドputBytesはputBytesByType、、、、およびメソッドと同じように動作します。オプションを完全removeOptionに削除するには、[DEX_REMOVE_ALL を使用します。
DEX_INDEX_COUNT	結果として、intオプション データではなく、オプションのインデックス付きエントリの数を返す値が返されます。追加の引数を持たないため、DEX_INDEXやDEX_INSTANCE_COUNTには使用できません。DEX_END従わなければなりません。getOptionと でのみgetOptionBytes有効です。
DEX_INSTANCE	intオプションのインスタンスを指定する必要があります(複数のインスタンスを持つことができる DHCPv6 オプションでのみ有効)。0 は最初のインスタンスを指定します。

iArgType	説明
DEX_INSTANCE_COUNT	結果として、intオプション データではなく、オプションのインスタンス数のカウントを返す値が返されます。追加の引数を持たないため、DEX_INSTANCEで使用することはできません。DEX_END従わなければなりません。getOptionと でのみgetOptionBytes有効です。
DEX_MORE	フラグがabool_t書き込まれる場所を指定する必要*moreがあります。指定したインデックスを超えて配列項目が存在する場合、この場所DEX_INDEX TRUE に設定されます。メソッドとgetOptionメソッドgetOptionBytesに対してのみ有効です。
DEX_MOVE_TO	コンテキストは、DEX_MOVE_TO直前のオプションまたはサブオプションに置きます。これ以外の引数はありません。省略した場合、コンテキストは変更されません。データmoveToOptionを取得せずにコンテキストを移動するために使用します。getOption と getOptionBytes のメソッドに対してのみ有効です。 （注） 存在しないオプションまたはサブオプションに移動しようとすると、エラーが記録されます。拡張機能moveToOptionがオプションが存在することを以前に確認しなかった場合に使用します。
DEX_OPTION_NAME	必要なオプション名を指定するには、次の手順を実行する必要があります。 const char * オプション名はdhcpv4-config、または dhcpv6-configオプション定義セットに含める必要があります。
DEX_OPTION_NUMBER	必要なオプション名を指定するには、次の手順を実行する必要があります。 const char * オプション名はdhcpv4-config、または dhcpv6-configオプション定義セットに含める必要があります。

iArgType	説明
DEX_PARENT	コンテキストを親オプションに移動します。これ以外の引数はありません。クライアントメッセージまたはリレーメッセージを越えて移動せず、コンテキストが変更されない場合はFALSEを返します。使用する場合は、最初のiArgTypeである必要があります。有効なのはgetOption、 getOptionBytes、 moveToOption 、およびメソッドだけです。
DEX_SUBOPTION_NAME	その後に、目的のサブオプションの名前を指定する必要があります。 const char * サブオプションは現行オプション定義に含まれる必要があります。
DEX_SUBOPTION_NUMBER	必要なintサブオプション番号を指定するには、次の手順を実行する必要があります。定義が存在する必要はありませんが、サブオプション番号は現行のオプション定義に含まれている必要があります。ただし、サブオプションが存在しない場合は、データのバイト BLOB と見なされます。
DEX_VENDOR_NAME	ベンダー文字列を指定するには、次の手順を実行する必要があります。 const char * この文字列は、適切なオプション定義セットを検索するためだけに使用されます。

オブジェクトとオプションの処理

以下のセクションでは、拡張でDHCPオブジェクトとオプションを処理する特殊な方法について説明します。

オブジェクトとオプションの処理方法の使用

拡張機能は、DHCP オブジェクトを設定するメソッドを呼び出し、DHCP オプションの取得、移動、配置、および削除を行うことができます。メソッドはsetObject、 getOption、 moveToOptionputOption、 removeOption 、および Tcl および C/C++ のメソッドです。

これらの新しいコールバック メソッドは、主に DHCPv6 をサポートするために導入されました。ただし、DHCPv4 のオプション関連機能を使用できます。実際には、DHCPv4getでは、これらのメソッドは、元の [ ]Bytes、 get[ ]Bytes、ByTypeput[Bytes] 、 put[BytesByType]removeByType メソッドよりも豊富なオプションにアクセスできるため、DHCPv4 に使用することをお勧めします。

ヒント C/C++ でのこれらのメソッドの使用方法の違いについては、を参照してくださいDEX の要求ディクショナリと応答ディクショナリ メソッド （51 ページ）。

DHCPv6 の場合は、オプションsetObjectにgetOptionアクセスmoveToOptionするためにputOption、removeOption 、 、 、 、およびメソッドを使用する必要があります。このsetObjectメソッドは、拡張がアクセスする可能性がある多くのリース、プレフィックス、およびメッセージ (クライアントまたは複数のリレー) が存在する可能性があるため、DHCPv6 に導入されました。したがって、setObject要求と応答のディクショナリ データ項目とオプションを取得する後続の呼び出しのコンテキストを設定するのに役立ちます。サーバーが拡張機能を呼び出すと、コンテキストは現在のリース (該当する場合)、プレフィックス (該当する場合)、およびクライアント メッセージに設定されます。たとえば、サーバーが拡張ポイントをpre-packet-encode呼び出すと、要求および応答のディクショナリ メッセージ コンテキストのみが有効になり、この拡張ポイントに関連付けられたリースまたはプレフィックスがないため、対応するクライアント メッセージに設定されます。ただし、サーバーが拡張ポイントをlease-state-change呼び出すと、応答ディクショナリのリース コンテキストを状態が変更されたリースに設定し、応答ディクショナリ プレフィックス コンテキストをリースのプレフィックスに設定し、要求および応答ディクショナリ メッセージ コンテキストを対応するクライアント メッセージに設定します。

C/C++ のオプションとサブオプション

一部の C/C++ 拡張では、DHCP オプションとサブオプションを処理するための特殊な引数型の値が提供されます。DEX_OPTION_*引数タイプは、オプション (またはサブオプション) の下の定義ではなく、標準 DHCPv4 または DHCPv6 オプション定義セットを使用することを指定します。したがって、DEX_OPTION_*は、サーバーが標準 DHCPv4 または DHCPv6 オプション定義セット内のオプション名または番号を参照することを意味しますが、DEX_SUBOPTION_*は、サーバーが現行オプション定義のサブオプション名または番号(存在する場合) を参照することを意味します。

したがって、DHCPv6 でオプションにアクセスする場合、オプションがカプセル化されるときに、DEX_OPTION_*の後に DEX_OPTION_*を付けて使用することがよくあります。ベンダー オプションを調べるときは、DEX_SUBOPTIONを使用します。DHCPv4 の場合は、クライアント パケット レベルでDEX_OPTIONを使用し、ネスト レベルに応じて 1 回以上DEX_SUBOPTIONします。一般的に、エンタープライズ番号またはベンダー名を持つオプションのみが含まれていますが、このオプションは禁止されません。オプション定義セットは、何が有効かを決定します (ただし、定義を順に処理できますが、その時点ではすべてがバイナリバイトとして扱われるため、可能な限り制限され、オプション名またはサブオプション名を使用することはできませんが、数字を使用する必要があります)。

メソッドgetOptionmoveToOption、 、 、 、putOptionおよびremoveOptionメソッドのオプションの順序request付け規則は、式の構文に似ています。順序は一般的に次の要素で構成されます。

- 前文節 ([parent |home])
- オプション句option(vendor [|enterprise] [instance])
- サブオプション句suboption(vendor [| |enterprise] [instance])

- 終了句 ([|instance-countindex-count |[index] [more] end)

呼び出しは、前文節、ゼロ以上のオプション節、ゼロ以上のサブオプション節 (それ自体にオプションおよびサブオプション節が続く場合があります) を使用して、終了句を続けて作成できます。一部の処理getはinstance-count、 、 index-count、およびmoreなどのメソッドを通じてのみmove-to可能であり、コンテキストを現在のオプションまたはサブオプションに移動するために、任意の場所に表示できることに注意してください。

オプション定義によってデータ形式が決まりますが、これは、以前の関数が特定のオプションに対して返すデータ形式とは異なる場合があります。特定のオプションを処理するには、

- ベンダークラスのオプション (DHCPv4 の場合はv-i-vendor クラス[124]、DHCPv6 のベンダークラス[16]) では、オプションの特定のインスタンスを (エンタープライズ ID または名前ではなく) 要求する場合、エンタープライズ ID を取得する唯一の方法は、生データを要求する (DEX_RAW を使用するDEX_INDEX)。
- DHCPv4 ベンダーオプション(v-i-vendor-class[124] およびvv-i-vendor-opts[125]) では、未処理データ (DEX_RAWを使用したDEX_INDEX) に対する操作は、オプション全体ではなく、そのオプションのインスタンス (プリセット値 0) にのみ適用されます。このオプションputOptionのデータ全体を取得する方法はありません。DHCPv6 ベンダー・オプションは個別のオプションであるため、これは問題ではありません。
- DHCPv4 ベンダー オプション (124 または 125) の 1 つが正しく書式設定されていない場合、データ全体が BLOB として返されます (インスタンス 0 を求めて特定のエンタープライズ ID を指定しなかった場合)。ただし、操作によっては拡張機能を使用putOptionしようとすると、そのデータが既存のデータに追加され、結果の形式が正しく設定されません。
- putOption(pDict,"01:02",DEX_OPTION_NUMBER,124,DEX_END)ベンダーオプションの場合、オプションがない場合、enterprise-id が使用できないために失敗します。putOption(pDict, "00:00:00:09:04:03:65:66:67", DEX_OPTION_NUMBER, 124, DEX_END)ただし、00:00:00:09 がエンタープライズ ID であり、04 で始まるバイトが、そのエンタープライズ ID のオプション データの長さであるために機能します。この場合、長さバイトが検証されputOption、正しい長さがない場合は失敗します。データ追加の推奨方法は、putOption(pDict, "65:66:67", DEX_OPTION_NUMBER, 124, DEX_ENTERPRISE_ID, 9, DEX_END) を使用することです。

オプションとオブジェクトのメソッド コールの例

これらのセクションでは、DHCP オプションとオブジェクトデータを処理するメソッドの使用方法の例をいくつか紹介します。

ベンダー クラス オプション データの処理

DHCPv4 の場合、クライアントへの応答に 2 つのエンタープライズ ID のベンダー識別ベンダクラス オプション (124) データを含めるには、次の方法putOptionを使用するいくつかのサンプル Tcl コードを示します。

```
$response putOption 65:66:67 option 124 enterprise 999998
```

```
 #adds "abc" (65:66:67) under enterprise-id 999998
$response putOption 68:69:6a:6b option v-i-vendor-class enterprise 999998 index append
 #appends "defg" (68:69:6a:6b) under the same enterprise-id
$response putOption 01:02:03:04 option 124 enterprise 999999
 #adds 01:02:03:04 under enterprise-id 999999
```

オプションを取得するには、次のgetOption方法を使用します。

```
$response getOption option v-i-vendor-class instance-count
 #returns 2 because there were two instances added (enterprise id 999998 and enterprise
 id 999999)
$response getOption option 124
 #returns index 0 of instance 0, which is 65:66:67
$response getOption option 124 index-count
 #returns 2 because there were two vendor classes added for the first enterprise id
(9999998)
$response getOption option 124 index raw
 #returns 00:0f:42:3e:09:03:65:66:67:04:68:69:6a:6b for the complete encoding of the
enterprise-id 999998 data (see RFC 3925)
$response getOption option 124 index 1
 #returns 68:69:6a:6b
$response getOption option 124 instance 1 index-count
 #returns 1 because there is only one vendor class
$response getOption option 124 instance 1 index raw
 #returns 00:0f:42:3f:05:04:01:02:03:04 for the complete encoding of the enterprise-id
999999 data (see RFC 3925)
$response getOption option 124 enterprise 999999
 #returns 01:02:03:04
```

データを削除するには、2removeOptionつの個別のエンタープライズ ID があるため、2 つの呼び出しが必要です。

```
$response removeOption option 124
$response removeOption option 124
```

オブジェクト データの処理

pre-packet-encode拡張ポイントで、クライアントのすべてのリースのデータを抽出するとします。このメソッドを使用する TclsetObjectコードのサンプルを次に示します。

```
proc logleasesinit { request response environ } {
   if { [$environ get "extension-point"] == "initialize" } {
     # Set up for DHCPv6 only]
     $environ put dhcp-support "v6"
     $environ put extension-extensionapi-version 2
  }
}
proc logleases { request response environ } {
   for { set i 0 } { 1 } { incr i } {
     # Set context to next lease
     if { ![$response setObject lease $i] } {
     # Lease does not exist, so done
     break
  }
  # Log the lease address, prefix name, and prefix address
  $environ log LOG_INFO "Lease [$response get lease-ipaddress], Prefix\
     [$response get lease-prefix-name] - [$response get prefix-address]"
  }
```

DHCP 拡張ディクショナリ

```
  # Restore the lease context to where we started
    $response setObject lease initial
    # Do other things...
}
```

これに対する C++ と同等のコードは次のようになります。

```
// Print the current leases for the client
for( int i=0; ; i++ ) {
   if( !pRes->setObject( pRes, DEX_LEASE, DEX_BY_INSTANCE, i ) )
      break;
   const char *pszLeaseAddress =
      pRes->get( pRes, "lease-ipaddress", 0, 0 );
   if( pszLeaseAddress == 0 )
      pszLeaseAddress = "<error>";
   const char *pszPrefixName =
      pRes->get( pRes, "prefix-name", 0, 0 );
   if( pszPrefixName == 0 )
      pszPrefixName = "<error>";
   pEnv->log(pEnv, DEX_LOG_INFO,
     "Lease %s, Prefix %s",
   pszLeaseAddress, pszPrefixName );
}
```