拡張ポイントの使用

拡張は、Cisco Prime Network Registrar が DHCP 要求をどのように処理し、応答するかに影響を与えたり、通常はユーザー インターフェイスを使って行うことができない DHCP サーバーの動作を変更したりするように記述することができます。この章では、DHCPv4 および DHCPv6 の拡張を添付できる拡張ポイントについて説明します。

拡張機能の使用

Tcl または C/C++ で記述できる拡張機能、機能を使用して、Cisco プライムネットワーク レジストラー DHCP サーバーの動作を変更およびカスタマイズできます。

DHCP サーバーで使用する拡張機能を作成するには、次の手順に従います。

1. 実行するタスクを決定します。どの DHCP パケット プロセスを変更しますか?
2. 使用するアプローチを決定します。パケットプロセスを変更する方法を教えてください。
3. 拡張機能をアタッチする拡張ポイントを決定します。
4. 言語 (Tcl または C/C++) を選択します。
5. 拡張機能を書き込む (また、コンパイルとリンクも可能です)。
6. DHCP サーバー構成に拡張機能を追加します。
7. 拡張ポイントに拡張子をアタッチします。
8. DHCP サーバーをリロードして、拡張を認識します。
9. 結果をテストしてデバッグします。

（注） Cisco Primeネットワーク レジストラーをアップグレードする際は、すべての DHCP C/C++ 拡張(dex エクステンション)を再コンパイルすることをお勧めします。

拡張機能の作成、編集、および添付

拡張機能を作成、編集、および添付できます。

拡張機能ポイントごとに複数の拡張機能を関連付けることができます。各拡張機能は、添付ファイルの作成時に使用されたシーケンス番号で指定された順序で実行されます。Web UI では、拡張が [DHCP 拡張ポイントのリスト（List DHCP Extension Points）] ページに拡張ポイントごとに表示される順序。CLI では、シーケンス番号の値をコマンドと共に**dhcp attachExtension** 使用します。

拡張ポイントごとの複数の拡張機能の詳細については、「」を参照してください複数の拡張機能に関する考慮事項 （7 ページ）。

ローカル アドバンスド Web UI

拡張機能を作成して添付するには、次の操作を行います。

ステップ 1 メニューから**DeployExtensions[DHCP]**サブメニューの下で[DHCP拡張のリスト/追加]ページを開きます。

ステップ 2 アイコンを**Add Extensions**クリックして、[DHCP サーバー拡張の追加] ダイアログ ボックスを開きます。

ステップ 3 拡張機能を作成した後、このページの 1 つ以上の拡張ポイントに添付できます。拡張機能をアタッチできる拡張ポイントを表示するには、[DHCP 拡張の一覧/追加] ページで**DHCP Extension Points**タブをクリックします。

ステップ 4 各拡張ポイントに複数の拡張子をアタッチする場合は、矢印キーをクリックしてエントリを並べ替えることで、その拡張子が処理される順序を変更できます。拡張を削除するには、[削除（Delete）] アイコンをクリックします。

CLI コマンド

このコマンド**extension**を使用するには、次の構文が必要です。

```
nrcmd> extension name create language extension-file entry-point
```

エントリ ポイントは、拡張子ファイル内のエントリ ポイントの名前です。また、DHCP サーバーがファイルをロードするたびに、初期エントリ ポイントに対してオプションの*init-entry*属性インイット・エントリー （32 ページ）値を設定することもできます(を参照)。この関数は、このモジュールにバインドされている任意の拡張ポイントから呼び出すことができます。拡張機能を 一**extension list**覧表示することもできます。

拡張機能をアタッチしてデタッチするには、**dhcp attachExtension**次**dhcp detachExtension**の構文が必要な DHCP サーバーを使用します。

```
nrcmd> dhcp attachExtension extension-point extension-name [sequence-number]
nrcmd> dhcp detachExtension extension-point [sequence-number]
```

シーケンス番号は、拡張ポイントごとに複数の拡張をアタッチする場合に適用され、シーケンスの順序は 1 から 32 まで増加します。省略した場合、デフォルトは 1 になります。

現在登録されている拡張機能を表示するには、**dhcp listExtensions**コマンドを使用します。

タスクの決定

拡張を適用するタスクは、通常、環境のニーズを満たすように、DHCPサーバー処理の変更です。要求の受信からクライアントへの応答まで、これらのDHCPサーバーの各処理ポイントで拡張機能を適用できます。

1. パケットを受信してデコードします。
2. クライアント クラスを検索、変更、および処理します。
3. 応答の種類を作成します。
4. サブネット (DHCPv6 の場合はリンク) を決定します。
5. 既存のリースを検索します。
6. リース要求をシリアル化します。
7. クライアントのリース受け入れ可否を決定します。
8. 応答パケットを収集し、エンコードします。
9. パケットの安定したストレージを更新します。
10. パケットを返します。

これらの手順の詳細な一覧 (各ステップで使用する拡張ポイント) が拡張を使用した DHCP 要求処理 （13 ページ）に表示されます。

たとえば、BOOTP 構成を使用する異常なルーティング ハブがある場合があります。このデバイスは、イーサネット・ハードウェア・タイプ (1) およびMAC アドレスを指定した BOOTP 要求を*chaddr*フィールドに出します。その後、同じ MAC アドレスを持つ別の BOOTP 要求を送信しますが、ハードウェアタイプはトークンリング (6) です。2 つの異なるハードウェアの種類を指定すると、DHCPサーバーは2つのIPアドレスをデバイスに割り当てます。通常、DHCP サーバーは、ハードウェア タイプ 1 の MAC アドレスとタイプ 6 の MAC アドレスを区別し、異なるデバイスと見なします。この場合、DHCP サーバーが同じデバイスに 2 つの異なるアドレスを渡すことを防ぐ拡張機能を作成できます。

アプローチの決定

多くの場合、単一の問題に対して多くのソリューションが使用できます。書き込む拡張子の種類を選択する場合は、まず入力DHCPパケットを書き換えることを検討する必要があります。DHCP サーバーの内部処理を知る必要がないため、これは良いアプローチです。

でタスクの決定 （3 ページ）説明する問題については、次のいずれかの方法で拡張機能を記述して解決できます。

- トークン リング (6) ハードウェア タイプ パケットをドロップします。
- パケットをイーサネット パケットに変更し、終了時に再度スイッチを戻します。

2 番目の方法では、より複雑な拡張が必要ですが、DHCP クライアントは DHCP サーバーからの応答を使用できます。2 番目の方法では、パケットの書き換えが**post-packet-encode**行われます(このポスト パケット エンコード （46 ページ）場合は、拡張ポイントを使用します)。他の方法では、他の拡張と拡張ポイントが必要です。

拡張言語の選択

Tcl または C/C++ で拡張機能を記述できます。DHCP サーバーに関する限り、各言語の機能は似ていますが、アプリケーション プログラミング インターフェイス (API) は言語設計の 2 つの非常に異なるアプローチをサポートするために若干異なります。

- **Tcl**— Tcl でのスクリプトは C/C++ でのスクリプトよりもやや簡単ですが、解釈され、シングルスレッドで、より多くのリソースが必要になる場合があります。ただし、C/C++ よりも深刻なバグが発生する可能性が低く、サーバー障害の可能性も低くなります。Cisco プライムネットワーク レジストラーは、Tcl バージョン 8.6 を現在サポートしています。
- **C/C++**：この言語では、外部プロセスとの通信を含む、可能な限り最大のパフォーマンスと柔軟性を実現できます。ただし、C/C++ API は Tcl API よりも複雑です。また、C/C++ では、拡張機能のバグが原因でサーバー障害が発生する可能性も高くなります。

言語に依存しない API

以下の概念は、Tcl または C/C++ で拡張機能を記述するかどうかに依存しません。

ルーチン署名

ファイル内に、複数の拡張関数を含めることができるルーチンとして、拡張機能を定義する必要があります。次に、1 つ以上の DHCP サーバー拡張ポイントに拡張機能を接続します。DHCP サーバーは、その拡張ポイントに到達すると、拡張機能が定義するルーチンを呼び出します。ルーチンは成功または失敗を返します。拡張エラー時にパケットをドロップするように DHCP サーバーを構成できます。

構成された各拡張機能に異なるエントリ ポイントを指定することで、1 つのファイル (Tcl ソース ファイル、C/C++ .dll または .so ファイル) を DHCP サーバーに対して複数の拡張子として構成できます。

サーバーは、少なくとも 3 つの引数 (要求、応答、および環境) の 3 つのディクショナリを使用して、すべてのルーチン エントリ ポイントを呼び出します。各ディクショナリには、キーと値のペアである、多くのデータ項目が含まれています。

- この拡張機能は、特定のデータ項目のディクショナリに対して get メソッドを実行することで、DHCP サーバーからデータ項目を取得できます。
- この拡張機能は、同じ名前付きデータ項目の多くについて、put 操作または remove 操作を実行してデータ項目を変更できます。

すべての拡張ポイントですべての辞書を使用することはできませんが、すべてのルーチンの呼び出しシーケンスは、すべての拡張ポイントで同じです。特定の拡張ポイントに存在しない

ディクショナリを参照しようとすると、拡張機能でエラーが発生します。（拡張ディクショナリ （25 ページ） を参照。）

ディクショナリ

要求、応答、およびサーバーのデータには、ディクショナリ インターフェイスを介してアクセスします。拡張ポイントには、要求、応答、環境という 3 種類のディクショナリが含まれています。

- **Request** — DHCP 要求に関連付けられた情報と、要求自体に含まれる**dictionary**すべての情報。データは、文字列、整数、IP アドレス、および BLOB 値です。
- **Response** DHCP クライアントに返す DHCP 応答パケットの生成に関連**dictionary**する情報。データは、文字列、整数、IP アドレス、および blob の値です。
- **Environment** — DHCP サーバーと拡張の間で渡**dictionary**される情報。

辞書の説明については、を参照してください拡張ディクショナリ （25 ページ）。

環境ディクショナリを使用して、異なる拡張ポイントにアタッチされた拡張機能間で通信することもできます。拡張が構成されている最初の拡張ポイントが検出されると、DHCP サーバーは環境ディクショナリを作成します。環境ディクショナリは、DHCP サーバーが許容されるデータ項目の名前を修正しない唯一のディクショナリです。環境ディクショナリを使用して、文字列値のデータ項目を挿入できます。

DHCP クライアントの要求と応答の間の制御フロー内のすべての拡張ポイント (変更の原因に**lease-state-change**応じて、 を除くすべての拡張ポイント) は、同じ環境ディクショナリを共有します。したがって、拡張は何らかの条件が存在することを判断し、環境辞書にセンチネルを置いて、後続の拡張が同じ条件を決定するのを避けることができるようにすることができます。

前の例では、**post-packet-decode**拡張ポイントの拡張は、特定の製造元のデバイス、BOOTP、およびトークン リングから、特定の製造元のデバイスからは、パケットが対象となっていたと判断し、トークン リングからイーサネットにハードウェアの種類を書き換えます。また、環境ディクショナリに sentinel を配置し、**post-packet-encode**拡張ポイントの非常に単純な拡張で、ハードウェアの種類をトークン リングに書き換えます。

ディクショナリでのユーティリティ メソッド

各ディクショナリには、拡張のトレース レベルをリセットし、出力ファイルに値を記録できるユーティリティ メソッドが関連付けられます。

設定エラー

拡張機能は、さまざまな理由で失敗する可能性があります。次に例を示します。

- サーバーはファイルを見つけることができません。
- エントリ ポイントまたは**init-entry**エントリ ポイントは、ファイルに表示されません。
- 拡張機能自体は、呼び出しから**init-entry**エラーを返すことができます。

それ自体では、拡張エラーは致命的ではなく、DHCPサーバーの起動を妨げません。ただし、任意の拡張ポイントで失敗した拡張機能を構成した場合、サーバーは起動しません。したがって、構成プロセスをデバッグするには、拡張**init-entry**ポイントにアタッチせずに、その時点で拡張機能インイット・エントリー （32 ページ）を構成できます(を参照)。このプロセスが正常に完了したら、拡張機能を拡張ポイントにアタッチできます。

外部サーバーとの通信

外部サーバーまたはデータベースと通信する拡張機能を作成して、クライアントクラスに影響を与えたり、着信DHCPクライアント要求を検証したりできます。このような拡張機能を記述することは複雑な作業であり、かなりのスキルとデバッグの専門知識が必要です。このような拡張機能はマルチスレッド化する必要があり、DHCPサーバーのパフォーマンスが許容レベルに維持される場合は、外部サーバーと非常に迅速に通信する必要があります。

パフォーマンスの低下は、要求を処理しているスレッドを停止拡張機能が原因で発生する可能性があります。拡張機能が外部サーバーと通信している間、スレッドが停止します。この対話に50～100ミリ秒以上かかる場合、サーバーのパフォーマンスに大きく影響します。この拡張機能を展開する特定の環境では、この影響を受ける場合と影響を与えない場合があります。

外部サーバーとの通信を同期化する(つまり、外部サーバーとの通信のために着信DHCPクライアント要求処理が停止する)ことを回避する1つの方法は、DHCPクライアント要求の処理中にこの通信を実行しないようにすることです。. これは明らかに聞こえるし、それはまた、それの顔に、不可能に聞こえる。ただし、DHCPクライアントサーバープロトコルの性質上、外部サーバーへのアクセスをDHCPクライアント要求処理から切り離す方法があります。

このボトルネックを回避するには、拡張機能の一部としてキャッシュ メカニズムを使用します。サーバーが要求に対して拡張機能を呼び出すときは、クライアントデータのキャッシュをチェックし(マルチスレッドの問題を回避するために適切なロックを使用して)します。クライアントが次の場合:

- キャッシュ内(および有効期限がない)では、キャッシュ内のデータに応じて、要求を受け入れるか拒否するかを拡張機能に依頼します。
- キャッシュ内に存在しない場合は、拡張キューに外部サーバーへの要求をキューに入れ(できればUDP経由で)、DHCPクライアント要求をドロップします。クライアントが要求を再送信する時点で、データはキャッシュ内に格納されます。

このキャッシング・メカニズムでは、拡張機能に受信側スレッド**(init-entry**拡張ポイントで開始および停止)が必要です。このスレッドは、ソケットを読み取り、応答でキャッシュを更新します。このスレッド(または別のスレッド)もタイムアウトし、キャッシュから古い項目を削除する必要があります。ただし、単一スレッドを使用する場合は、より大きな受信ソケット・バッファー・サイズの設定が必要になる場合があります。

これらの方法は、DHCPサーバーの負荷が高く、外部サーバーの速度が十分でない場合にのみ必要です。しかし、この状況は実際にはあまりにも一般的であることが判明しました。また、外部サーバーに到達できない場合(接続タイムアウトが秒ではなく分数の場合)に何が起こるかを考慮してください。

拡張機能の認識

DHCPサーバーは、最初に起動時または再ロード時に自身を構成する場合にのみ、拡張機能を認識します。拡張機能または拡張機能の構成は、一般的に変更できます。ただし、サーバーをリロードまたは再起動するまでは、変更は無効です。DHCPサーバーの再読み込みを忘れることは、拡張機能のデバッグ中に頻繁に発生するエラーの原因になることがあります。

Cisco Prime Network レジストラーでリロードが必要な理由は、エクステンションを事前にロードし、サーバー設定時に準備することで、処理への影響を最小限に抑えるためです。この方法は実稼働モードでは便利ですが、拡張機能をデバッグするときには、ある程度の不満が生じる可能性があります。

複数の拡張機能に関する考慮事項

任意の拡張ポイントで複数の拡張機能を登録できます。DHCPサーバーは、処理を再開する前に、拡張ポイントに接続されているすべての拡張機能を実行します。

- 拡張機能が明示的にデータ項目を設定しない限り、拡張機能は明示的にデータ項目を設定しないでください。たとえば、(表 31-25 の表 31-5 のドロップ環境ディクショナリデータ項目について説明されているように)、拡張機能は、ほとんどの拡張ポイントでクライアント・パケットのドロップを要求できます。

  サーバーは、ドロップ・セットが False の拡張ポイントで登録された最初の拡張を呼び出します。1 つ以上の拡張機能を True または False に設定できます。すべての拡張機能が明示的にドロップを True または False に設定した場合、サーバーは最後に実行された拡張機能が要求した任意のアクションを実行します。

  これは望ましい動作ではない場合があります。したがって、このデータ項目の場合、パケットをドロップする場合にのみ、拡張機能がドロップを True に設定する方が良いでしょう。このようにすれば、すべての拡張機能がこの規則で再生された場合、いずれかのエクステンションが要求した場合にパケットがドロップされます。

- 別の拡張機能がパケットの破棄を望む場合、その処理を行う必要がなくなる可能性があるため、ドロップが True の場合は、拡張機能がすぐに返される場合があります。
- 後の拡張ポイントで使用する項目を格納するために環境ディクショナリを使用する場合、それらのデータ項目名は、その拡張機能に固有の接頭辞またはサフィックスを使用する必要があります。これにより、データ項目名の競合が発生する可能性が低くなります。
- 少なくとも 1 つの環境ディクショナリ データ項目、リース (DHCPv4 の場合) またはクライアント (DHCPv6) を使用してデータを格納するために使用できるユーザー定義データ(表 5 : 一般的な環境ディクショナリ データ項目を参照) には、特別な注意が必要です。

  これらの拡張機能が互いの値を保持し、認識するために特別な注意を払っていない限り、このデータ項目を複数の拡張機能を使用するのは困難な場合があります。したがって、複数の拡張機能がこのデータ項目を使用できると想定することはできません。

- 拡張機能を最初に実行するか、必要に応じて最後に実行するかを指定する必要があります。たとえば、サーバーが最初に特定のパケットをドロップする拡張機能を実行する必要

があります。なぜなら、これはサーバーの処理の負荷を軽減するためです(ドロップがtrueの場合、残りの拡張が直ちに戻ると仮定します)。

TCL 拡張

Tcl で拡張機能を記述する場合は、Tcl API、エラーとブール変数の処理方法、および Tcl 拡張機能の初期化方法を理解する必要があります。Cisco Prime Network Registrar は TCL バージョン 8.6 を使用します。

（注） 単一の TCL インタープリタが DHCP サーバーによって使用されます。これは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。TCL 拡張機能は、高性能なマルチスレッド DEX 拡張機能や非常にシンプルで高速な操作に変更する前のより複雑なロジックのプロトタイピングに最適です。

TCL アプリケーション プログラム インターフェイス

すべての Tcl 拡張は、同じルーチンシグネチャを持っています。

```
proc yourentry { request response environ } { # your-code }
```

辞書のデータ項目を操作するには、これらの引数をコマンドとして扱う必要があります。したがって、入力パケットの*giaddr*を取得するには、次の書き込みを行います。

```
set my_giaddr [ $request get giaddr ]
```

これにより、パケット内の*giaddr*の文字列値にmy_giaddr Tcl 変数が設定されます。たとえば、10.10.1.5 または 0.0.0.0 などです。

次の Tcl ステートメントを使用して、入力パケットの*giaddr*を書き換えることができました。

```
$request put giaddr "1.2.3.4"
```

複数の拡張ポイントに対して 1 つのルーチンエントリを構成し、サーバーが呼び出す拡張ポイントに応じてその動作を変更するには、DHCP サーバーは、環境ディクショナリの拡張ポイント**extension-point**の ASCII 名をキーの下に渡します。

Tcl 拡張機能の例については、Cisco Prime Network Registrar ディレクトリ /opt/nwreg2/local/examples/dhcp/tcl（デフォルト）を参照してください。

TCL エラーの処理

次の場合は、Tcl エラーが発生します。

- 使用できない辞書を参照します。
- 使用できないディクショナリ データ項目を参照します。
- 無効なデータ項目 (たとえば、無効な IP アドレス) に対して put 操作を要求します。

このような場合、ステートメントを catch error ステートメントで囲む場合を除き、拡張は直ちに失敗します。

```
catch { $request put giaddr "1.2.3.a" } error
```

Tcl 拡張機能の構成

Tcl 拡張を構成するには、それを書き込み、次の拡張ディレクトリに配置します。

/var/nwreg2/local/extensions/dhcp/tcl

DHCP サーバーは、起動時に拡張を構成すると、Tcl ソース ファイルをインタプリタに読み込みます。ソースファイル内の Tcl インタプリタがファイルをロードできない場合、構文エラーは拡張に失敗します。通常、DHCP サーバーは、エラーを見つけるために Tcl からログ ファイルにエラー トレースバックを生成します。

TCL でのブール変数の処理

環境ディクショナリでは、ブール変数は文字列値で、値は**true**OR になります。 **false** DHCP サーバーは、値を**true**または**false**に設定する拡張を要求します。ただし、要求ディクショナリまたは応答ディクショナリでは、ブール値は 1 バイトの数値形式であり、**true1false**です**0**。C/C++ 拡張機能の方が効率的ですが、この方法では Tcl API が少し複雑になります。

TCL での init-entry 拡張ポイント

Tcl 拡張は**init-entry**拡張ポイントをサポートインイット・エントリー （32 ページ）しています (を参照してください) 、 *init-args*パラメータでコマンド ラインに渡す**arguments**引数は、key に関連付けられた環境辞書に表示されます。

単一の TCL インタープリタが DHCP サーバーによって使用されます。これにより、情報フローの問題が回避され、クライアント要求のフォローに使用できる情報を保存するためのグローバル変数を使用できますが、パフォーマンスに重大な影響を与えます。

すべての TCL 拡張は TCL インタープリタを共有していることに注意してください。Tcl 拡張が、グローバル変数を初期化したり、プロシージャを定義したりする場合は、これらが他の Tcl 拡張グローバル変数またはプロシージャ名と矛盾していないことを確認してください。

C/C++ 拡張

すべての DHCP C/C++**dex**拡張は、DHCP 拡張の略で拡張です。

C/C++ API

C/C++ API **entryinit-entry**のルーチンと ルーチンの両方のルーチン署名は次のとおりです。

```
typedef int (DEXAPI * DexEntryPointFunction)(
int iExtensionPoint,
```

```
dex_AttributeDictionary_t* pRequest,
dex_AttributeDictionary_t* pResponse,
dex_EnvironmentDictionary_t* pEnviron );
```

3 つの構造体へのポインターと共に、拡張ポイントの整数値は、各ルーチンのパラメーターの 1 つです。

C/C++ API は、共有ライブラリを Cisco Prime Network レジストラー DHCP サーバー ファイルとリンクする必要がないように、特に構築されています。拡張機能を構成するときに、ルーチンのエントリを構成します。要求ディクショナリ、応答ディクショナリ、および環境ディクショナリに対して実行する操作に必要なコールバック情報は、拡張ルーチンに渡される 3 つのディクショナリ パラメータを構成する構造体に含まれています。

DHCP サーバーは、すべてのバイナリ情報をネットワーク順に返しますが、実行アーキテクチャに対して正しく配置されるとは限りません。

C/C++ でのタイプの使用

型を使用する多くの C/C++ ルーチンが使用できます**getByType()**。これらのルーチンは、パフォーマンスに影響を受けやすい環境で使用するように設計されています。これらのルーチンの背後にある理由は、拡張機能が**init-entry**、たとえば、ポイントで、型へのポインターを取得し、その後 C/C++ API のルーチンを呼び出すときに、文字列値の名前の代わりにポインターを使用する可能性があります。この方法で型を使用すると、実行の拡張処理フローから 1 つのハッシュ テーブルルックアップが削除され、拡張機能のパフォーマンスが (少なくともわずかに) 向上する必要があります。

C/C++ 拡張機能のビルド

ディレクトリ /opt/nwreg2/local/examples/dhcp/dex には、サンプルの C/C++ 拡張コードと、サンプル拡張機能を構築するために設計された短いメイクファイルが含まれています。独自の拡張子を作成するには、このファイルを変更する必要があります。このセクションには、Visual C++ および GNU C++ のセクションがあります。コメント行を移動するだけで、ご使用の環境に合わせてファイルを構成できます。

拡張機能はインクルード ファイル dex.h を参照する必要があります。このファイルには、プログラムが C/C++ API を使用するために必要な情報が含まれています。

.so ファイル（すべての dex 拡張は共有ライブラリ）を作成したら、/var/nwreg2/local/extensions/dhcp/dex ディレクトリに移動する必要があります。その後、それらを設定できます。

C/C++ でのスレッドセーフな拡張の使用

DHCP サーバーはマルチスレッドなので、その DHCP サーバー用に記述された C/C++ 拡張機能はスレッド セーフである必要があります。複数のスレッド 、および場合によっては複数のプロセッサは、同じエントリ ポイントでこれらの拡張機能を同時に呼び出すことができる必要

があります。Cisco Prime Network レジストラー用の C/C++ 拡張機能を設計する前に、マルチスレッド環境用のコードを記述した経験が豊富である必要があります。

⚠

注意 C/C++ 拡張機能はすべてスレッド セーフである必要があります。そうしないと、DHCP サーバーは正しく動作せず、診断が非常に困難な方法で失敗します。これらの拡張機能が使用するすべてのライブラリおよびライブラリ ルーチンもスレッド セーフである必要があります。

いくつかのオペレーティング システムでは、使用するランタイム関数がスレッド セーフであることを確認する必要があります。各関数のマニュアルを確認します。いくつかのオペレーティングシステムでは、スレッドセーフな特別なバージョンが提供されています(多くの場合、関数名_r)。

スレッドセーフでない呼び出しを行うスレッドがある場合、そのスレッドは、その呼び出しの安全なバージョンまたはロックされたバージョンを構成するスレッドに影響を与えます。これにより、メモリの破損、サーバー障害などが発生する可能性があります。

これらの問題の原因が明らかになることはめったにないので、これらの問題を診断することは非常に困難です。サーバー障害を引き起こすには、非常に高いサーバー負荷または多数のプロセスを持つマルチプロセッサ マシンが必要です。数日間の実行時間が必要な場合があります。多くの場合、拡張実装の問題は、一定期間の重い負荷が続くまで現れなくなることがあります。

ランタイムまたはサードパーティのライブラリによっては、スレッドセーフでない呼び出しを行う可能性があるため、検出できない外部ファイルが (UNIX**nm**上で) リンクされている場合は、実行可能ファイルを確認してください。

ライブラリーのルーチンが、次の表に示す _r 接尾部を持たないルーチンを呼び出す場合、ライブラリーはスレッド・セーフではなく、使用できません。これらのライブラリ ルーチンのスレッド セーフ バージョンへのインターフェイスは、オペレーティング システムによって異なる場合があります。

asctime_r	getgrid_r	getnetent_r	getrPCbynumber_r	lgamma_r
ctermid_r	getgrnam_r	getprotobyname_r	getrPCent_r	localtime_r
ctime_r	gethostbyaddr_r	getprotobynumber_r	getservbyname_r	nis_sperror_r
fgetgrent_r	gethostbyname_r	getprotoent_r	getservbyport_r	rand_r
fgetpwent_r	gethostent_r	getpwnam_r	getservent_r	readdir_r
fgetspent_r	getlogin_r	getpwent_r	getspent_r	strtok_r
gamma_r	getnetbyaddr_r	getpwuid_r	getspnam_r	tmpnam_r
getgrent_r	getnetbyname_r	getrPCbyname_r	gmtime_r	ttyname_r

C/C++ 拡張の設定

サーバーの実行中は .dll ファイルと .so ファイルがアクティブであるため、上書きすることはお勧めできません。サーバーを停止した後、.dll ファイルと .so ファイルを新しいバージョンで上書きできます。

C/C++ 拡張のデバッグ

C/C++ 共有ライブラリは DHCP サーバーと同じアドレス空間で実行され、DHCP サーバー内の情報へのポインタを受け取るため、C/C++ 拡張機能のバグによって DHCP サーバーのメモリが非常に簡単に破損し、サーバーの障害が発生する可能性があります。このため、C/C++ 拡張機能の作成とテストを行う場合は、細心の注意を払います。多くの場合、Tcl 拡張機能を持つ拡張機能へのアプローチを試して、パフォーマンスを向上させるために C/C++ で拡張機能をコーディングする必要があります。

C/C++ における DHCP サーバー メモリへのポインター

C/C++ 拡張インターフェイス ルーチンは、次の 2 つの形式で DHCP サーバー メモリにポインターを返します。

- 一連のバイトへの char* ポインタ。
- abytes_tと呼ばれる構造体へのポインタで、関連付けられた長さ (dex.h で定義される) を持つ一連のバイトへのポインタを提供します。

どちらの場合も、DHCP サーバー メモリへのポインタは有効ですが、拡張ポイントで拡張機能が実行されます。また、この要求を処理するシリーズの残りの拡張ポイントにも有効です。したがって、**post-packet-decode**拡張ポイントで返されるabytes_tポインターは、**post-send-packet** 拡張ポイントで有効です。

ポインターは、環境ディクショナリーに入れられた情報が有効である限り、有効です。ただし、1 つの例外があります。1 つの C/C++ ルーチンである**getType**、 は、型を参照するabytes_t へのポインターを返します。これらのポインターは、拡張機能の有効期間全体を通じて有効です。通常、サーバーは、拡張ポイントでこのルーチンを**init-entry**呼び出し、共有ライブラリの静的データの型を定義するabytes_t構造体へのポインターを保存します。返**getType**されるabytes_t 構造体へのポインターは、初期化の**init-entry**呼び出しから初期化解除の呼び出しまで有効です。

C/C++ での init-entry エントリ ポイント

DHCP サーバーは、**init-entry**拡張機能を構成するときに 1 回、拡張機能を構成解除するときに 1 回、拡張ポイント（インイット・エントリー （32 ページ）を参照）を呼び出します。dex.h ファイルは、構成および構成解除の DEX_UNINITIALIZE DEX_INITIALIZE呼び出しの拡張ポイントとして渡される 2 つの拡張ポイント値を定義します。拡張ポイント データ項目の環境ディクショナリ値は、**initialize**各呼**uninitialize**び出しの中またはまたは呼び出しの中にあります。

拡張ポイントを**init-entryinitialize**呼び出すときに、環境ディクショナリ データ**persistent**項目に値**true**が含まれている場合は、呼び出しから戻る前に、いつでも**uninitialize**環境ディクショナリ ポインタを保存して使用できます。このようにして、バックグラウンド スレッドは、環境ディクショナリ ポインタを使用して、サーバー ログ ファイルにメッセージを記録できます。一度に1つのスレッドがディクショナリへの呼び出しを処理するように、ディクショナリへのすべてのアクセスをインターロックする必要があることに注意してください。保存されたディクショナリ ポインタは、**uninitialize**拡張が呼び出しから戻ったときに使用できます。このようにして、バックグラウンド スレッドは、終了時にメッセージをログに記録できます。

拡張を使用した DHCP 要求処理

Cisco プライムネットワーク レジストラー DHCP サーバーには、独自のエクステンションをアタッチできる拡張ポイントがあります。制御の処理フロー内で、それらを使用する場所を示すわかりやすい名前が付きます。

拡張ポイントは DHCP クライアントからの入力要求の処理に関連しているため、DHCP サーバーが要求をどのように処理するかを理解しておくと役立ちます。要求処理は、次の3つの一般的なステージで行われます。

1. 初期要求処理 (表 1 : 拡張機能を使用した初期要求処理を参照)
2. DHCPv4 または DHCPv6 処理 (表 2 : 拡張機能を使用した DHCPv4 または DHCPv6 要求処理を参照)
3. 最終応答処理 (表 3 : 拡張機能を使用した最終応答処理を参照)

表 *1* : 拡張機能を使用した初期要求処理

クライアント要求処理ステージ	使用する拡張ポイント
1. DHCP クライアントからパケットを受信します。	**pre-packet-decode**
2. パケットをデコードします。	**post-packet-decode**
3. クライアント クラスを決定します。	
4. クライアント クラスを変更します。	**post-class-lookup**
5. クライアント クラスを処理し、クライアントを検索します。	**pre-client-lookup post-client-lookup**
6. 要求から応答コンテナーを作成します。	

表 *2:* 拡張機能を使用した *DHCPv4* または *DHCPv6* 要求処理

クライアント要求処理ステージ	使用する拡張ポイント
1. DHCPv4 で、このクライアントに既に関連付けられているリースがあれば、そのクライアントのリースを探すか、または新しいリースを見つけます。	
2. このクライアントに関連付けられているすべての要求をシリアル化します(要求がシリアル化キューの先頭に到達すると処理が続行されます)。	
3. DHCPv6 では、クライアント要求を処理し、必要に応じてリースを生成します。サーバーは、バインディングごとに使用できるプレフィックスごとに、少なくとも 1 つの優先リースをクライアントに提供しようとします。 リースを生成し、クライアント要求に対してリース状態を変更できますが、予約済みのリースに対しては生成できません。	**generate-lease**(DHCPv6**lease-state-change** では両方で複数の呼び出しが可能)
4. リースがこのクライアントに対して (まだ) 許容できるかどうかを判断します (DHCPv6 では複数回発生する可能性があります)。	**check-lease-acceptable**
5. 必要に応じて DNS 更新操作を開始します (DHCPv6 で複数回発生する可能性があります)。	

表 *3:* 拡張機能を使用した最終応答処理

クライアント応答処理ステージ	使用する拡張ポイント
1. 応答パケットに含めるすべてのデータを収集します。	
2. リース データベースに書き込みます。	
3. エンコード用の応答パケットを準備します。	**pre-packet-encode**
4. クライアントに送信する応答パケットをエンコードします。	**post-packet-encode**
5. パケットをクライアントに送信します。	**post-send-packet**
6. クライアントと要求のすべてのコンテキストを解放します。	**environment-destructor**

これらの手順と拡張機能を使用するその他の機会については、次のセクションで説明します。拡張ポイントは に示**bold**されています。

DHCPv6 拡張の有効化

デフォルトでは、拡張はDHCPv4のみをサポートすると想定されます。DHCPv6拡張を記述するには、次の**init-entry**必要がある拡張ポイントを実装する必要があります。

1. dhcpサポート環境データ項目**v4**を (DHCPv4 の場合のみ、プリセット値)、(DHCPv6**v6**の場合**v4,v6**のみ)、または(DHCPv4およびDHCPv6の場合)に設定します。このデータ項目は、拡張機能がサポートする内容をサーバーに示します。
2. 拡張拡張*API*バージョン環境データ項目を**2**に設定します。(拡張拡張 *api* バージョンが**2**に設定されていない場合、*dhcp*サポート データ項目は無視されます。

パケット形式、DHCP プロトコル、および内部サーバー データの違いにより、DHCPv4 とDHCPv6用の拡張を個別に作成する必要がある場合があります。ただし、両方の種類の拡張機能の基本は非常に同じです。

サーバーは、処理時に基本的に同じ場所でこれらの拡張ポイントを呼び出しますが、クライアントごとに複数のリース要求が発生する可能性があるため、一部のDHCPv6拡張ポイントを複数回呼び出すことができます。

パケットの受信

DHCP サーバーは、ポート 67 の DHCPv4 パケットをポート 547 (DHCP 入力ポート) で受信し、それらを処理用にキューに入れます。UDP 入力キューをできるだけ早く空にしようと試み、空きスレッドが処理可能になるとすぐに、受信したすべての要求を内部リストに保持し、処理を行います。このキューの長さは設定でき、設定された最大長を超えてはなりません。

パケットのデコード

フリー スレッドが使用可能な場合、DHCP サーバーは、入力要求を処理するタスクを割り当てます。最初に行われる操作は、入力パケットをデコードして、それが有効なDHCP クライアント パケットかどうかを判断することです。このデコード プロセスの一部として、DHCP サーバーは、すべてのオプションが有効かどうかを確認します。つまり、オプションの長さが要求パケットの全体的なコンテキストで意味を持つかどうかを確認します。また、DHCP 要求パケット内のすべてのデータもチェックしますが、この段階ではパケット内のデータに対しては何も処理を行いません。

入力パケット**pre-packet-decode**を書き換える場合は、拡張ポイントを使用します。DHCP サーバーがこの拡張ポイントを通過した後、パケットからのすべての情報を複数の内部データ構造体に格納し、後続の処理をより効率的にします。

クライアントクラスの決定

クライアント クラスルックアップ *ID*で式を構成すると、この段階で DHCP サーバーが式を評価します (式の説明式の使用方法については、「」を参照)。式の結果は、<null> または文字列に変換された値です。文字列の値は、クライアント クラス名または <none> のいずれかである必要があります。<none> の場合、サーバーは、クライアント クラスルックアップ *ID*が構成さ

れていない場合と同じ方法でパケットの処理を続行します。<null>応答の場合、またはクライアント クラスルックアップ *ID*を評価するエラーの場合、サーバーはエラー メッセージをログに記録し、パケットを破棄します(**post-class-lookup**拡張ポイントで構成された拡張がパケットをドロップしないようにサーバーに指示しない限り)。クライアントクラスを設定するプロセスの一部として、DHCP サーバーはそのクライアントクラスに対して設定された制限*ID*を評価し、要求と共に保存します。

クライアントクラスの変更

DHCP サーバーは、クライアントクラスルックアップ *ID*を評価し、クライアント クラスを設定した後、拡張ポイントにアタッチ**post-class-lookup**された任意の拡張を呼び出します。この拡張機能を使用して、クライアント クラスが要求に関連付けられるデータ (制限 *id*など) を変更できます。また、クライアント クラスルックアップ *ID*の評価によってパケットがドロップされた場合も、拡張機能は学習します。この拡張は、パケットをドロップする必要があるかどうかを調べますが、サーバーにパケットをドロップしないようにサーバーに指示します。

また、**post-class-lookup**拡張ポイントで実行されている拡張機能は、要求に対して新しいクライアントクラスを設定し、現在のクライアントクラスではなくそのクライアントクラスのデータを使用できます。これは、クライアントクラスを設定する唯一の拡張ポイントで、実際にそのクライアントクラスを要求に使用します。

クライアントクラスの処理

クライアントクラス処理を有効にした場合、DHCP サーバーはこの段階で処理を実行します。

拡張ポイント**pre-client-lookup**を使用して、検索を妨げたり、既存のデータを上書きするデータを提供したりして、クライアントが参照するように影響を与えます。DHCPサーバーは、拡張ポイント**pre-client-lookup**を通過した後、クライアントをローカルデータベースまたはLDAP データベース (構成されている場合) で検索します (拡張機能が特に禁止しない限り)。

サーバーは、クライアントを参照した後、クライアントエントリのデータを使用して、追加の内部データ構造を入力します。DHCP サーバーは、指定されたクライアント クラス エントリのデータを使用して、クライアントエントリで指定されていないデータを完成させます。DHCP サーバーは、追加の処理のために内部データ構造のさまざまな場所に格納されているすべてのデータを取得すると、次の拡張ポイントを実行します。

拡張ポイント**post-client-lookup**を使用して、クライアント クラスの処理から入力された内部サーバーデータ構造を調べるなど、クライアントクラスの参照プロセスの操作を確認します。また、拡張ポイントを使用して、DHCPサーバーが追加の処理を行う前にデータを変更することもできます。

応答コンテナの作成

この段階では、DHCPサーバーは要求の種類を決定し、入力に基づいて適切な応答コンテナーを構築します。たとえば、要求が DHCPDISCOVER である場合、サーバーは DHCPOFFER 応

答を作成して処理を実行します。入力要求が BOOTP 要求の場合、サーバーは BOOTP 応答を作成して応答処理を実行します。

DHCPv6 の場合、サーバーは要求に応じて、アドバタイズ パケットまたは REPLY パケットを作成します。

ネットワークとリンクの決定

DHCPサーバーは、すべての要求の発信元のサブネットを特定し、IPアドレスを含む一連のアドレスプール、スコープ、プレフィックス、またはリンクにマッピングする必要があります。

DHCPv4 の場合、DHCP サーバー内部はネットワークの概念であり、この場合は LAN セグメントまたは物理ネットワークを指します。DHCPサーバーでは、すべてのスコープまたはプレフィックスが 1 つのネットワークに属します。

スコープまたはプレフィックスの中には、ネットワーク番号とサブネットマスクが同じであるため、同じネットワーク上でグループ化されているものもあります。その他のグループは、主スコープまたはプレフィックス ポインターを通じて関連付けられているため、グループ化されます。

Cisco Prime Network レジストラー DHCP サーバーは、次の順序で DHCP クライアント要求を処理するために使用するネットワークを決定します。

1. ソース・アドレスを判別する場合は、*giaddr*か、*giaddr*がゼロの場合は、要求が到着したインターフェースのアドレスを判別します。
2. このアドレスを使用して、このアドレスと同じサブネット上にあるサーバーで構成されたスコープまたはプレフィックスを検索します。サーバーがスコープまたはプレフィックスを見つけられない場合は、要求を削除します。
3. スコープまたはプレフィックスを見つけた後、そのネットワークを使用して以降の処理を行います。

DHCPv6 処理については、リンクとプレフィックスの決定を参照してください。

リースの検索

DHCPv4の場合、DHCPサーバーがネットワークを確立すると、ネットワーク レベルで保持されているハッシュ テーブルが検索され、ネットワークが既にクライアント *ID*を認識しているかどうかを確認できます。このコンテキストでは、このクライアントが以前にこのネットワークでオファーまたはリースを受け取り、その時点以降、別のクライアントにリースが提供されなかったりリースされたりしていないことを意味します。したがって、現在のリースまたは使用可能な期限切れのリースがネットワーク レベルのハッシュ テーブルに表示されます。DHCPサーバーは、リースを検出した場合、次の手順に進みます。

DHCP サーバーがリースを検出せず、これが BOOTP または DHCPDISCOVER 要求である場合、サーバーはネットワーク内のスコープまたはプレフィックスから予約済みリースを検索します。

予約済みリースが見つかった場合、サーバーはスコープまたはプレフィックスとリースの両方が受け入れられるかどうかを確認します。予約済みリースと、それを含むスコープまたはプレフィックスに関して、以下の条件を満たす必要があります。

- リースは使用可能である必要があります (別の DHCP クライアントにはリースされません)。
- スコープまたはプレフィックスは、要求の種類 (BOOTP または DHCP) をサポートする必要があります。
- スコープまたはプレフィックスは、非アクティブ化された状態であってはなりません。
- リースは非アクティブ化された状態であってはなりません。
- 選択タグには、クライアント選択基準をすべて含める必要があり、クライアント選択基準から除外されるものは含まれていなければなりません。
- スコープまたはプレフィックスは、更新専用の状態にすることはできません。

予約済みのリースが許容される場合、サーバーは次の手順に進みます。このクライアントの既存のリースまたは予約済みリースが見つからなかった場合、サーバーはこのクライアントに使用可能な IP アドレスを見つけようとします。

DHCPサーバーが使用する一般的なプロセスは、このネットワークに関連付けられたすべてのスコープまたはプレフィックスをラウンドロビン順にスキャンし、クライアントに対して許容可能なスコープと使用可能なアドレスを探します。有効なスコープまたはプレフィックスには、次の特性があります。

- クライアントに選択基準が関連付けられている場合、選択タグにはクライアント包含基準がすべて含まれている必要があります。
- クライアントに選択基準の除外が関連付けられている場合、選択タグにはクライアント除外基準が含まれていなければなりません。
- スコープまたはプレフィックスがクライアント要求タイプをサポートする必要がある- クライアント要求が DHCPREQUEST である場合は、DHCP のスコープまたはプレフィックスを有効にする必要があります。同様に、要求が BOOTP 要求である場合は、BOOTP と動的 BOOTP のスコープまたはプレフィックスを有効にする必要があります。
- 更新のみの状態にすることはできません。
- 非アクティブ状態にすることはできません。
- 使用可能なアドレスが必要です。

サーバーが許容範囲またはプレフィックスを見つけられない場合、メッセージをログに記録してパケットを廃棄します。

DHCPv6 処理については、リンクとプレフィックスの決定を参照してください。

リース要求のシリアル化

1 つのクライアントとリースに対して複数の DHCP 要求を同時に処理できるため、DHCPv4 要求をリース レベルでシリアル化する必要があります。サーバーは、リースでキューに登録し、キューイングの順序で処理します。

DHCPv6の場合、サーバーはクライアント(リンク単位)でシリアル化され、リースではシリアル化されません。

リースの受け入れの決定

DHCPv4の場合、DHCPサーバーは、クライアントに対してリースが(まだ)受け入れられるかどうかを判断します。初回クライアントの新規取得リースの場合は、許容されます。ただし、サーバーが既存のリースの更新を処理する場合、サーバーがリースを許可してから受け入れ可能な条件が変更されている可能性があるため、その受け入れ可能性を再度確認する必要があります。

クライアントの現在のリースとは異なる予約がある場合、サーバーは最初に予約済みリースが許容できるかどうかを判断します。リリースの受け入れ基準は次のとおりです。

- 予約済みリースが使用可能である必要があります。
- 予約済みリースは非アクティブ状態にしないでください。
- スコープまたはプレフィックスは非アクティブ状態にしないでください。
- 要求がBOOTPの場合、スコープまたはプレフィックスはBOOTPをサポートする必要があります。
- 要求がDHCPの場合、スコープまたはプレフィックスがDHCPをサポートしている必要があります。
- クライアントに選択基準がある場合、選択タグにはクライアントの包含条件がすべて含まれている必要があります。
- クライアントに選択基準の除外がある場合、選択タグにはクライアントの除外基準が含まれていなければなりません。
- このリースに以前関連付けられているクライアントが現在のクライアントではない場合、スコープまたはプレフィックスは更新専用の状態であってはなりません。

予約済みリースがこれらの基準をすべて満たしている場合、DHCPサーバーは現在のリースを受け入れられないと見なします。このクライアントに予約されたリースがない場合、または予約済みリースが受け入れ可能な条件を満たしていない場合、DHCPサーバーは現在のリースを受け入れ可能な状態で調べます。

受け入れ可能な基準は次のとおりです。

- リースは非アクティブ状態にしないでください。
- スコープまたはプレフィックスは非アクティブ状態にしないでください。
- 要求がBOOTPの場合、スコープまたはプレフィックスはBOOTPをサポートする必要があります。要求がDHCPの場合、スコープまたはプレフィックスがDHCPをサポートしている必要があります。
- クライアントがこのリースの予約を持っておらず、要求がBOOTPである場合、スコープまたはプレフィックスは動的BOOTPをサポートする必要があります。
- クライアントがこのリースの予約を持っていない場合、他のクライアントもできません。
- クライアントに選択基準がある場合、選択タグにはクライアントの包含条件がすべて含まれている必要があります。

- クライアントに選択基準の除外がある場合、選択タグにはクライアントの除外基準が含まれていなければなりません。
- このリースに以前関連付けられているクライアントが現在のクライアントではない場合、スコープまたはプレフィックスは更新専用の状態であってはなりません。

ヒント　DHCP サーバーの処理のこの時点で、拡張ポイントを**check-lease-acceptable**使用できます。これを使用して、受け入れ性テストの結果を変更できます。これは細心の注意を払って行うだけです。

リースが受け入れられないと判断した場合、DHCP サーバーは、現在処理されている特定の DHCP 要求に応じて、異なるアクションを実行します。

- **DHCPDISCOVER**DHCP サーバーは現在のリースを解放し、このクライアントに対して別の許容可能なリースを取得しようとします。
- **DHCPREQUEST** :リースが無効であるため、DHCP サーバーは DHCP クライアントに NACK を送信**SELECTING**します。クライアントは、すぐにディスカバー要求を発行して新しい DHCPOFFER を取得する必要があります。
- **DHCPRENEW**()DHCP**DHCPREBIND**サーバーは、DHCP クライアントを強制的に INIT フェーズに入れようとする NACK を DHCP クライアントに送信します(DHCP クライアントが DHCPDISCOVER 要求を強制的に発行するように試みます)。クライアントが実際に要求を発行するまで、リースは有効です。
- **BOOTP**：DHCP サーバーは、現在のリースを解放し、このクライアントに対して受け入れ可能な別のリースを取得しようとします。

注意　延長点には細心**check-lease-acceptable**の注意を払ってください。拡張ポイントが返す答えが、DHCPDISCOVER 要求または動的 BOOTP 要求で実行された使用可能なリースの検索での受け入れ可能なチェックと一致しない場合、無限のサーバー ループが発生する可能性があります (即時または次の DHCPDISCOVER または BOOTP のいずれか要求)。この場合、サーバーは新しく使用可能なリースを取得し、それが受け入れられないと判断し、新しく使用可能なリースを取得し、連続ループで許容できないリースを判断します。

DHCPv6 リース

DHCP サーバーは、クライアントのIA_NA、IA_TA、およびIA_PDオプションをスキャンして、IPv6 リース要求をDHCPv6 バインディング処理します (を参照)。これらのオプションごとに、サーバーはクライアントが明示的に要求するリースを考慮します。クライアントとバインディング (IA オプションおよび IAID) にリースがすでに存在する場合、サーバーは、リースがまだ受け入れられるかどうかを判別します。クライアントがクライアントに対してまだ存在しないリースの場合、サーバーは次の場合にクライアントにそのリースを与えようとします。

- 別のクライアントまたはバインディングがリースをまだ使用していません。

- リースのプレフィックスには、割り当てアルゴリズム属性にクライアント要求フラグが設定されています。
- リースは使用でき、使用できるプレフィックス (を参照してくださいDHCPv6 プレフィックスのユーザービリティ （21 ページ）)。

次に、サーバーは、クライアントが予約を使用していること、およびクライアントが、リンク上の各使用できるプレフィックスに対して、優先する有効期間がゼロ以外の、有効なリースを持っていることを確認しようとします。したがって、サーバーはこれらの各バインディングを次のように処理します。

1. プレフィックス割り当てアルゴリズム属性で予約フラグが設定されている場合は、バインディングにクライアント予約(まだ使用されていない)を追加します。サーバーは、予約に対して適切なタイプの最初のバインディングを使用します。つまり、IA_NAバインディングにアドレス リースを使用し、IA_PD バインドのプレフィックス リースを使用します。
2. クライアントが使用できるプレフィックスごとに優先されるゼロ以外の有効期間を持つリースがない場合、サーバーはクライアントにリースを割り当てようとします。プレフィックス割り当てアルゴリズムフラグは、サーバーがリースを割り当てる方法を制御します。

DHCPv6 プレフィックスのユーザービリティ

使用できるプレフィックス:

- 非アクティブ化されません。
- 期限切れではありません。
- バインディング・タイプのリースを許可します。
- クライアントの選択基準 (存在する場合) に一致します。
- クライアント選択除外条件 (存在する場合) に一致しません。

DHCPv6 リースのユーザービリティ

使用できるリースは次のとおりです。

- 使用不可でないこと。
- 失効していない。
- 非アクティブ化されていません。
- 別のクライアント用に予約されていません。
- すべての更新を阻害したり、再起動時に更新を禁止したりしない.
- 更新された場合は更新可能 (IA_TAリースは更新可能ではありません)。
- 有効な有効期間がゼロ以外の場合は、リーアスブルです。

DHCPv6 リースの割り当て

サーバーは、プレフィックスに新しいリースを割り当てる必要がある場合、プレフィックス**generate-lease**拡張フラグがアロケーション アルゴリズム属性に設定されている場合、拡張ポイントで登録されている拡張を呼び出します。（リースの生成 （42 ページ） を参照）。拡張

機能は、割り当てるアドレス (IA_NAまたはIA_TAバインディング) またはプリフィックス (IA_PDバインディング) を指定するか、サーバーが通常の割り当てアルゴリズムを使用するように要求するか(割り当てアルゴリズムで有効になっている場合)、またはこのプレフィックスのリースの割り当てをスキップするようサーバーに要求します。サーバーが無効なアドレスまたはプレフィックスを指定した場合、または既に使用中の場合、サーバーは拡張を再度呼び出す可能性があります。

拡張が許可されていない場合、拡張機能が登録されていないか、拡張機能がサーバーの通常の割り当てアルゴリズムを要求する場合、サーバーはランダムに生成されたアドレスを割り当てるか、(プレフィックス割り振りアルゴリズム属性によって制御される)最初の最適な使用可能なプレフィックスを見つけてリースを作成します。

サーバーがリースを取得し、そのサーバーで受け入れ可能なDHCPv6 リースのユーザービリティ （21 ページ） チェックを行うと (を参照)、サーバーは**check-lease-acceptable**エクステンションポイントで登録されているエクステンションを呼び出して、エクステンションがリースの受け入れ可能を変更できるようにします。（check-lease-acceptable （44 ページ） を参照）。通常、この拡張ポイントを使用して、許容できる結果を許容できない結果に変更します。ただし、サーバーでは許容できない結果を許容可能な結果に変更できますが、悪影響を及ぼす可能性があるため、この方法は推奨されません。リースが受け入れられない場合、サーバーは別のリースを割り当てようとする可能性があります。したがって、無限ループを避けるために注意してください。場合によっては、クライアントが取得**check-lease-acceptablegenerate-leasegenerate-lease**するリースのフル コントロールに対して 、および 拡張ポイントが必要になる場合があります。

サーバーは、各**check-lease-acceptable**リースの各クライアント要求の拡張ポイントを呼び出します。

応答パケット データの収集

この処理の段階では、DHCP サーバーはDHCP応答で返送するすべてのデータを収集し、応答を送信するアドレスとポートを決定します。拡張ポイントを**pre-packet-encode**使用して、応答でDHCP クライアントに返送されるデータを変更したり、DHCP応答を送信するアドレスを変更したりできます。（pre-packet-encode （46 ページ） を参照）。

注意 拡張ポイントでドロップされたパケットは、DHCP パケットでも BOOTP パケットでも、残りのリース時間の間は Cisco Prime Network レジストラーのリース状態データベースにリースされるアドレスを示します。 **pre-packet-encode** このため、パケットを早い時点でドロップすることをお勧めします。

応答パケットの符号化

この段階では、DHCP は応答データ構造内の情報をネットワーク パケットにエンコードします。この DHCP クライアントが DNS アクティビティを必要とする場合、DHCP サーバーは DHCP サーバーの DNS 処理サブシステムに対して DNS 作業要求をキューに入れます。この要

求は、可能な限り実行されますが、通常はクライアントにパケットを送信する前には実行されません。（pre-packet-encode （46 ページ） を参照。）

安定ストレージの更新

この段階で、DHCP サーバーは、続行する前に、情報のディスク上のコピーが IP アドレスに関して最新の状態であることを確認します。DHCPv6 の場合、これには複数のリースが含まれる場合があります。

パケットの送信

DHCP**post-send-packet**要求/応答サイクルポスト送信パケット （47 ページ）の重大な時間制約の外部で実行する処理については、拡張ポイント (を参照) を使用します。サーバーがパケットをクライアントに送信すると、この拡張ポイントがコールされます。

DNS 応答の処理

ここでは、DHCP サーバーが DNS に名前を追加する処理を簡単に示します。

1. :DHCP サーバーは、転送(A レコード)DNS 要求で使用する名前を作成します。 **Builds up a name to use for the A record** DHCPv6 の場合、これらは AAAA レコードです。DNS 名は、DHCP 要求のオプションから通常取り込まれるクライアント要求ホスト名およびクライアント ドメイン名データ項目、および DNS 更新設定 (ホスト名生成/v6 ホスト名生成式を含む) など、さまざまなソースから取得されます。
2. - この段階では、DNS 名更新要求の前提条件は、名前が存在しないことを示します。 **Tries to add the name, asserting that none exists yet** 成功した場合、DHCP サーバーは逆レコードの更新を続行します。
3. サーバーはホスト名を追加しようとし、ホストが存在し、送信されたレコードと同じ TXT レコード(DHCPv6 の DHCID レコード)を持っていることを主張します。 **Tries to add the name, asserting that the server should supply it**

   - これが成功した場合、サーバーは次の手順に進みます。
   - 失敗した場合、サーバーは名前付け再試行が終了したかどうかをチェックします。
   - 名前付けエントリが使い果たされなかった場合は、最初のステップに戻り、A レコードの名前を作成します。

   DHCPv6 の場合、サーバーは TXT レコードの代わりに DHCID レコードを使用します。また、DHCPv6 クライアントは複数のリースを持つことができますが、転送ゾーンは同じか、または異なる可能性があります。

4. :DHCP サーバーは、リバース (PTR) レコードに関連付ける名前を認識したので、レコードの所有者であると見なすことができるため、前提条件なしでリバース レコードを更新できます。 **Updates the reverse record** 更新に失敗した場合、DHCP サーバーはエラーを記録します。

リース状態変更のトレース

サーバーは、リース**lease-state-change**が状態を変更するたびに (および、状態が変更された場合にのみ) 拡張ポイントを呼び出します。既存の状態は、応答ディクショナリ**lease-state**データ項目にあります。新しい状態は、 の環境ディクショナリ**new-state**にあります。これは**new-state**、既存の状態と等しくない(存在する場合、サーバーは拡張機能を呼び出しません)。サーバーはさまざまな場所で呼び出すため、この拡張機能は読み取り専用であり、ディクショナリ項目を変更しないようにする必要があります。この拡張ポイントは、リース状態の変更を追跡する場合にのみ使用します。

有効なリースクエリ通知の制御

サーバーは、dhcp リスナーの*leasequery-send-all*属性に基づいて、アクティブなリースクエリ通知用にリースがキューに入っているかどうかを判断します。この属性が有効な場合、DHCP サーバーは常にアクティブな leasequery クライアントに通知を送信します。無効にするか、または設定解除した場合、DHCP サーバーは、アクティブな leasequery クライアントで正確な状態を維持するために必要な通知のみを送信します。

顧客が書き込んだ拡張を使用してリースの送信を制御できるように (特定の状態の変更に関する場合など)、新しいデータ項目であるアクティブリースクエリコントロールが要求ディクショナリと応答ディクショナリの両方に追加されました。これらのデータ項目には、次の 3 つの値があります。

- 0 - 未指定(サーバーが通知を送信するかどうかを決定します)
- 1- 送信 (サーバーが通知を送信します)
- 2 -送信しない(サーバーは通知を送信しません)

アクティブリースクエリコントロールデータ項目は 0 として初期化され、未指定です。

（注） これらのデータ項目は書き込みおよび読み取りできますが、読み取られる値は、以前に書き込まれた値のみです。

これらのデータ項目は、書き込み後に DHCP サーバーに特定のアクションを強制実行させることができますが、前に書き込みせずに読み取ると、常に 0 (未指定) が返されます。これらのデータ項目では、処理中のリースに対する変更 (存在する場合) に関するメッセージをアクティブな leasequery クライアントに送信するかどうかを決定する際に DHCP サーバーが行う選択を決定することはできません。したがって、これらのデータ項目は技術的には読み取り/書き込み可能ですが、読み取りでは以前に書き込んだ内容を判断することしかできません。

これらのデータ項目は、リースがアクティブなリースクエリ通知のためにキューに入れられたときと同様に、内部リース状態データベースにリースが書き込まれるときに検査されます (応答ディクショナリが最初に調べられますが、次に要求が返されます)。これは、チェック-リース許容およびリース状態変更拡張ポイントの後、パケットエンコード前の拡張ポイントより前に発生します。したがって、これらの属性に対してパケットエンコード前の拡張ポイントまたはそれ以降に行われた変更は無視されます。

リースがアクティブなリースクエリ通知のキューに入っているかどうかは、次のように決定されます。

応答のアクティブリースクエリ制御	要求のアクティブリースクエリコントロール	リースクエリ送信-すべて	操作
0：指定なし	0：指定なし	偽または未設定	条件付き (リースクエリ送信-すべて属性の説明を参照)
0：指定なし	0：指定なし	[はい（True）]	Sent
0：指定なし	1：送信する	Ignored	Sent
0：指定なし	2 - 送信しない	Ignored	送信されない
1：送信する	Ignored	Ignored	Sent
2 - 送信しない	Ignored	Ignored	送信されない

（注） 応答と要求のアクティブ・リース照会制御は、リース照会-*send-all*属性の検査の前に検査されます。

これらのディクショナリ データ項目のいずれかが未指定以外の値を持つ場合、その値は dhcp リスナの*leasequery-send-all*属性で設定されている値をオーバーライドします。

（注） アクティブなleasequery情報の送信を制御するには、リース状態変更拡張ポイントでのみ実行される単一の拡張を書き込むことはできません。

リース状態の変更は、予期した場合には発生しない場合があります。たとえば、リースがリースされている場合、同じクライアントがディスカバー/オファー/リクエスト/ACKサイクルを通過すると、リース状態変更拡張ポイントは呼び出されません。したがって、アクティブなleasequeryクライアントへの情報の転送を絶対的に制御するには、要求処理でアクティブ・リースクエリ制御属性を初期化し、場合によって、それを変更するか、または、リース状態変更拡張点で応答ディクショナリ値で操作することによってオーバーライドする必要があります。

拡張ディクショナリ

すべての拡張は、3 つの引数を持つルーチンです。これらの引数は、要求ディクショナリ、応答ディクショナリ、および環境ディクショナリを表します。すべての辞書がすべての拡張に使用できるわけではありません。次の表は、拡張機能ポイントと、それらのポイントで使用できるディクショナリを示しています。

表 *4:* 拡張ポイントと関連する辞書

拡張ポイント	ディクショナリ
init-entry	環境
pre-packet-decode	要求、環境
post-packet-decode	要求、環境
pre-client-lookup	要求、環境
post-client-lookup	要求、環境
post-class-lookup	要求、環境
generate-lease	要求、応答、環境
lease-state-change	対応,環境
check-lease-acceptable	要求、応答、環境
pre-packet-encode	要求、応答、環境
post-packet-encode	要求、応答、環境
post-send-packet	要求、応答、環境
environment-destructor	環境

（注） サーバーが DHCPv6 再設定メッセージを送信すると、要求**pre-packet-encode**なしで**post-packet-encode**、**post-send-packet**および 拡張ポイントを呼び出すことができます。

要求ディクショナリと応答ディクショナリの場合、このメソッドを**isValid**使用して、辞書が拡張ポイントで使用できるかどうかを調べることができます。

3 つのディクショナリはそれぞれ、名前と値のペアで構成されています。環境ディクショナリは、すべての拡張ポイントで使用でき、最も単純なディクショナリです。要求ディクショナリと応答ディクショナリは複雑で、データが入力されます。したがって、これらのディクショナリの1つに値を設定する場合は、データ型を値に一致させる必要があります。値の取得、書き込み、および削除にはディクショナリを使用できます。

環境ディクショナリ

環境ディクショナリは、すべての拡張ポイントで使用できます。厳密には、名前と値の両方が文字列である名前と値のペアのセットです。

DHCPサーバーは、環境ディクショナリを使用して、拡張機能のさまざまな点で異なる方法で拡張機能と通信します。一部の拡張ポイントでは、サーバーは、変更する拡張機能の情報を環境ディクショナリに配置します。その他の場合、拡張機能は、拡張機能の処理が完了した後、フローまたはデータを制御する環境ディクショナリ内の値を配置できます。

環境ディクショナリは、拡張機能が名前と値のペアを入れることができるという特徴で一意です。文書化されていない名前と値のペアを使用してもエラーは発生しませんが、サーバーはこれらを認識しません。これらの名前と値のペアは、拡張機能ポイント間でデータを通信する場合に役立ちます。

DHCP サーバーは、DHCP 要求が到着し、処理を通じてその要求にディクショナリが残ると、環境ディクショナリを作成します。したがって、**post-packet-decode**拡張ポイントで実行される拡張機能は、環境ディクショナリにデータを格納し、**pre-packet-encode**拡張ポイントで実行される拡張機能は、ディクショナリからそのデータを読み取ることができます。

（注） **init-entry**拡張ポイントには、固有の環境ディクショナリがあります。

一般的な環境ディクショナリ データ項目

次の表のデータ項目は、すべての拡張ポイントで環境ディクショナリで有効です。（各辞書のデータ項目に固有の環境辞書の各セクションを参照してください。）

データ項目は、入力、出力、またはその両方です。

- 入力：DHCP サーバーは値を設定し、それを拡張に入力します。
- 出力：値は DHCP サーバーに出力され、DHCP サーバーは DHCP サーバーに出力され、DHCP サーバーに対して動作します。1 つの拡張ポイントで複数の拡張機能が存在する可能性があるため、拡張ポイントで実行されている以前の拡張機能がこれを設定している可能性があるため、これは、その拡張ポイントで実行される後の拡張機能への「入力」になる可能性があります。テーブルが「入力」ではないことを示している場合、DHCPサーバーがその拡張ポイントで拡張を呼び出す前に明示的にこれを設定しなかったことを意味します。

表 *5:* 一般的な環境ディクショナリ データ項目

環境データ項目	説明
drop（input[1]/出力）	拡張機能が終了したときに、ドロップ値が文字列**true**と等しい場合、DHCP サーバーは入力パケットをドロップし、ログ ファイルにメッセージを記録します。最初は**false**に設定します。ほとんどの拡張ポイントで使用できますが、すべてではありません (**generate-lease**など)。 （注） 拡張機能ポイントごとに複数の拡張機能の*drop*を使用する方法の複数の拡張機能に関する考慮事項 （7 ページ）推奨事項については、「」を参照してください。

環境データ項目	説明
拡張子名(入力)	拡張機能が構成された名前。同じコードを複数の異なる拡張機能として、また複数の異なる拡張ポイントで構成できます。 これにより、構成方法に応じて、1 つのコードで異なる処理を実行できます。コードでは、この文字列を使用して、自身の名前を知る必要がある拡張名シーケンス文字列内で自分自身を見つけることもできます。
拡張子名シーケンス(入力)	この拡張ポイントに対して構成されている拡張機能を表すコンマ区切りの文字列を提供します。拡張機能は、その前と後に実行できる拡張機能を決定することができます。拡張名データ項目は、現在実行中の拡張機能を提供します。 たとえば、最初の拡張子として**tclfirst**構成し**dexscript**、5 番目として構成した場合、拡張子の名前のシーケンスには**"tclfirst,,,,dexscript"**が含まれます。
拡張ポイント(入力)	拡張ポイントの名前。たとえば、**post-packet-decode** のようになります。
拡張シーケンス(入力)	拡張ポイントの拡張のシーケンス番号を表す文字列。
ジャダーオーバーライド(出力)	このデータ**pre-packet-decode**項目は、 、 **post-packet-decode**、**pre-client-lookup**、**post-client-lookup**および**post-class-lookup**拡張によって設定され、クライアントのネットワークの場所を決定する際に使用する IPv4 アドレスまたはスコープ名を指定できます (giaddr または受信インターフェースのアドレスの代わりに)。これは DHCPv4 要求に対してのみ使用されます (DHCPv6 では無視されます)。スコープ名を指定した場合、クライアントの場所を特定するためだけに使用され、クライアントがそのスコープからリースを取得する必要はありません。
リンク・アドレス・オーバーライド(出力)	このデータ**pre-packet-decode**項目は、 、 **post-packet-decode**、**pre-client-lookup**、**post-client-lookup**および**post-class-lookup**拡張子によって設定され、クライアントのネットワークロケーションを決定する際に使用する IPv6 アドレスまたはプレフィックス名を指定できます (Relay-Forw のリンクアドレスまたは受信インターフェースのアドレスの代わりに)。これは DHCPv6 要求に対してのみ使用されます (DHCPv4 では無視されます)。プレフィックス名を指定した場合、クライアントの場所を決定するためだけに使用され、クライアントがそのプレフィックスからリースを取得する必要はありません。リンクとプレフィックスの決定を参照してください。
ログドロップメッセージ(出力)	ドロップ**true,**値が文字列と等しく、*log-drop-message*値が拡張が終了したときに文字列**false**と等しい場合、DHCP サーバーは入力パケットをドロップしますが、ログ ファイルにメッセージを記録しません。 には適用されません**init-entry**。

環境データ項目	説明
IP によるリリース (出力)	これを有効にするには、**パケットデコード前**、**パケットデコード後**、**クライアント参照前**、**ポストクライアントルックアップ**、クラス**後参照**拡張ポイントで呼び出される拡張によって設定する必要があります。 これは、DHCPRELEASE 要求にのみ適用されます。に**true**設定すると、DHCPRELEASE 要求から派生したクライアント *ID*によってリースを取得できない場合、IP アドレスによってリースを解放するようにサーバーに指示します。
トレース・レベル (出力)	この番号を設定すると、この要求を処理するすべての拡張機能の拡張トレース・レベル・サーバー属性の現行設定が、その番号になります。
ユーザー定義データ(出力)	要求処理の前にリースと共に保存されたリースのユーザー定義データ属性を使用して設定します。このファイルは、 の**pre-packet-encode**前にディスクに書き込むことができますが、使用することはできません。 null に設定すると、サーバーはリースからのユーザー定義データを無視します。NULL 文字列値を使用して以前の値を削除することはできません。応答にのみ適しています。 サーバーがユーザー定義データをリースに書き込む場合、読み取り専用クライアントユーザー定義データ応答ディクショナリー項目はその値を想定します。 （注） このデータ項目を拡張ポイントの複数の拡張機能で使用する場合は注意が必要です。複数の拡張機能に関する考慮事項 （7 ページ） を参照してください。

[1] **post-client-lookup** と **post-class-lookup** 以外のすべて。*drop* は出力にすぎません。**クライアント参照後**および**クラス後参照**の場合、指定したクライアントクラスが存在する場合、サーバー セットは**false**に設定されます。**クライアントクラス**が存在しない場合は*true* (したがって、拡張の変更が*false*に**ドロップ**しない限り、サーバーはこのパケットの処理を続行しません)。

初期環境ディクショナリ

*init-args*と**init-entry**. または、環境ディクショナリから読み取る拡張機能の構成情報を指定できます。一連の属性と値のペアを持つDHCPプロパティの初期環境ディクショナリを設定でき、各組み合わせはすべての環境ディクショナリで使用できます。この機能を使用すると、さまざまな構成情報およびカスタマイズ情報を指定できます。任意の拡張は、*init-args*または**init-entry**メソッドで必要とされる静的データ領域に格納しなくても、このデータを環境ディクショナリから直接読み取ることができます。

初期環境ディクショナリーを使用して定義された値を、任意の環境ディクショナリーから読み取ることができます。初期環境ディクショナリに表示される任意の属性に対して新しい値を定義することもできます。これらの新しい値は、その環境ディクショナリ (通常は処理される要求パケットの存続時間) の有効期間で使用できます。ただし、他の環境ディクショナリの内容

は変更されません。(別の要求に関連付けられている) 新しい環境ディクショナリは、DHCP サーバーの初期環境ディクショナリプロパティによって定義された属性と値のペアを参照します。

さらに、これらの初期環境ディクショナリ属性と値のペアは、環境ディクショナリの値の列挙には表示されません。これらは、環境ディクショナリーで現在定義されていない属性値を要求する場合にのみ使用できます。属性と値のペアは、実際には環境ディクショナリに表示されません。したがって、いずれかの属性に新しい値を定義すると、その新しい値は環境ディクショナリに表示されます。後で値を削除した場合、元の値は、要求する必要がある場合は再び使用可能になります。

要求ディクショナリと応答ディクショナリ

要求ディクショナリと応答ディクショナリには、アクセス可能な名前の固定セットがあります。ただし、すべての拡張ポイントからすべての名前にアクセスすることはできません。これらのディクショナリは、内部サーバーのデータ構造を拡張機能で読み取り/書き込みアクセス、場合によっては読み取り専用アクセスに使用できるようにします。各データ項目には、特定のデータ型があります。PUT 操作で正しいデータ型 (C/C++ 拡張の場合) を省略した場合、または DHCP サーバーが正しいデータ型 (Tcl 拡張の場合) に変換できない場合、拡張は失敗します。

要求ディクショナリは、要求の処理の開始時に使用できます。サーバーが応答を作成すると、要求ディクショナリと応答ディクショナリの両方が使用できるようになります。応答ディクショナリが使用可能になる前にアクセスするとエラーになります。

一般に、拡張機能を使用してサーバーの情報データを変更することはできません。ただし、拡張機能を使用して構成済みのデータを変更できる場合もありますが、その1つの要求に対してのみ処理を行う間のみです。

付録 Bには、受信したクライアント要求 (要求ディクショナリ) と送信された応答 (応答ディクショナリ) で使用できるオプションとデータ項目の詳細が記載されています。

復号化された DHCP パケット データ項目

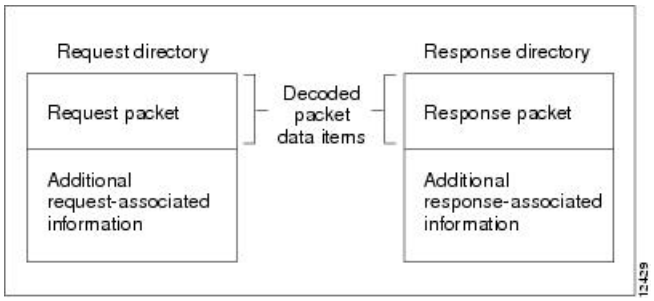
DHCP プロトコルは要求-応答 UDP ベースのプロトコルであり、したがって、DHCP サーバー操作の刺激は通常、クライアントからの DHCP 要求です。通常、そのクライアントに返される DHCP 応答が返されます。

DHCP 拡張機能は、DHCP 要求の情報入力を、ほとんどの拡張ポイントで拡張に対して使用可能にし、**pre-packet-encode**拡張ポイントで利用可能な DHCP 要求への応答として送信される情報をpre-packet-encode （46 ページ）提供します (を参照してください)。

この DHCP パケット ベースの情報に加えて、DHCP 要求を処理するときに DHCP サーバーが使用する追加データがあります。このデータは、サーバーのアーキテクチャの一部として、DHCP 要求または DHCP 応答のいずれかに関連付けられます。このデータの多くは拡張でも利用できるようになっており、その多くは読み取りと書き込みの両方が可能です。

したがって、要求ディクショナリと応答ディクショナリには、各ディクショナリに2つのクラスのデータが含まれています。これらは、デコードされたパケットデータ項目だけでなく、他の要求または応答に関連付けられたデータ項目が含まれています。デコードされたパケットデータ項目は、DHCP要求またはDHCP応答に直接含まれている、またはDHCPからのデータ項目です。デコードされたパケットデータ項目にアクセスすると、DHCP要求パケットとDHCP応答パケットを読み取り、場合によっては書き換えができます。次の図は、要求ディクショナリと応答ディクショナリの関係を示しています。

図 *1*:拡張機能の要求と応答の辞書



要求ディクショナリのデコードされたパケット データ項目を使用して、*giaddr*、 *ciaddr*、およびすべての着信DHCPオプションなどのDHCP要求パケットからの情報にアクセスできます。同様に、*giaddr*と*ciaddr*を設定し、応答ディクショナリ内のデコードされたパケット データ項目にアクセスして、発信 DHCP 応答の DHCP オプションを追加および削除できます。

デコードされたパケットデータ項目によって提供されるパケット情報へのアクセスは、すべて利用できるわけではないことを認識することが重要です。その拡張ポイントで使用できる特定のデータ項目は、各拡張ポイントの説明に一覧表示されます。デコードされたパケットデータ項目は常にグループとしてアクセス可能であるため、グループとして一覧表示されます。

名前によって DHCP オプションにアクセスします。このオプションが存在しない場合、サーバーはそのオプションのデータを返しません。デコードされた要求またはデコードされた応答にオプションを配置すると、put 操作でデータを既存のデータに追加する場合を除き、デコードされた要求またはデコードされた応答に既に同じ名前を持つオプションが置き換えられます。

一部の DHCP オプションには、複数の値を指定できます。たとえば、ルーター オプションには、1つ以上のIPアドレスを関連付けることができます。これらの複数の値へのアクセスは、オプション名に対するインデックス付きの操作によって行われます。

ヒント 要求**clear**または応答ディクショナリの操作は、デコードされたパケットのすべてのオプションを削除します。

パラメータ リスト オプションの使用

DHCPサーバーが特別に処理するオプション*dhcp-*パラメータ要求リストは、次のいずれかの方法で処理します。

- 名前の下のバイトの複数値のオプション*dhcp-*パラメータ要求リスト。
- 名前の下の BLOB (バイトのシーケンス) オプションは、名前の下で*- dhcp* パラメーター要求リスト *blob*です。

いずれかの名前を使用してオプションを取得または設定できます。DHCP サーバーは、応答ディクショナリ内で、要求ディクショナリ内と異なる方法で*dhcp* パラメーター要求リスト(およびその*-blob*バリアント) を処理します。要求ディクショナリでこのオプションにアクセスすると、要求ディクショナリ内の別のDHCPオプションにすぎません。ただし、応答辞書では、特別な処理が行われます。

応答ディクショナリの*dhcp*パラメータ要求リストオプションを使用して、DHCPまたはBOOTPクライアントに返されるオプションの順序を制御できます。応答辞書にオプションを入れると、DHCP サーバーは既存のオプションを並べ替えて、オプションにリストされているオプションが最初に、リストに表示される順序になるようにします。その後、残りのオプションは、リスト内の最後のオプションの後に現在の順序で表示されます。DHCPサーバーはリストを保持し、リストを新しいものに置き換えるまで、そのリストを使用して、応答に入れる将来のオプションを並べ替えます。

拡張機能は、応答ディクショナリ内の*dhcp*パラメーター要求リストの取得操作を行う場合、オプションを検索するためにデコードされた応答パケットを検索しません。代わりに、DHCP サーバーは、デコードされた応答パケットに現在含まれているすべてのオプションのリストを含む 1 つを合成します。

拡張ポイントの説明

以下のセクションでは、各拡張ポイント、それらのアクション、およびデータ項目について説明します。すべての拡張ポイントについて、環境ディクショナリで トレース**extension-point**レベルのデータ項目値を読み取り、設定できます。ほとんどの拡張ポイントでは、パケットをドロップするようにサーバーに指示することもできます。

インイット・エントリー

拡張**init-entry**ポイントは、DHCPサーバーが拡張機能を構成または構成解除するときに呼び出す追加のポイントです。このエントリ ポイントは、拡張機能の他のエントリ ポイントと同じシグネチャを持ちますが、環境ディクショナリのみを使用できます。拡張機能**init-entryinit-entrydhcp**を CLI で設定するのではなく、既に構成済みの拡張機能に定義することで暗黙的に設定**attachExtension**します。

（注） DHCPv6**init-entry**の拡張ポイントを有効にする (または DHCPv4 の場合は拡張ポイントを無効にする) 拡張ポイントを指定する必要があります。

エントリ ポイントの名前を**init-entry**持つ を構成するだけでなく、**init-entry**ポイントを呼び出す前に、DHCPサーバーが環境ディクショナリの文字列引数の下に読み込む引数の文字列を構

成することもできます。引数を使用すると、異なる初期化引数を指定してカスタマイズされた拡張を作成できるため、異なる動作を引き出すためにコードを変更する必要はありません。

（注） サーバーが拡張ポイントで拡張機能を**init-entry**呼び出す順序は、リロードからリロード、リリースからリリースまで異なる場合があります。

注意 拡張は、uninitializeに呼び出されたときに、作成したスレッドを終了し、それ自体の後にクリーンアップしてから戻る必要があります。拡張が返されると、DHCPサーバーは、拡張機能をメモリからアンロードします。

init-entry の環境ディクショナリ

init-entryに固有の環境ディクショナリ データ項目については、次の表を参照してください。

表 *6 : init-entry* 環境ディクショナリ データ項目

環境ディクショナリ データ項目	説明
dhcp サポート(入力/出力)	サーバーが拡張機能の登録済み拡張ポイントを呼び出す必要があるDHCPのバージョンまたはバージョン。には**v4**、、**v6**または**v4,v6**、 を指定できます。
終了状態(出力)	**リース状態変更**拡張ポイントに接続された拡張の場合、指定した場合、**リース状態変更**拡張ポイントは、リースの現在の状態が*exiting-state*で指定された状態である場合にのみ呼び出されます。拡張は、指定された状態が終了した場合にのみ呼び出されます。指定されていない場合、拡張が**リース状態変更**拡張ポイントにアタッチされている場合、すべての状態変更に対して拡張が呼び出されます。指定した場合、終了状態は有効なリース状態(利用可能、提供済み、リース済み、期限切れ、使用不可、解放済、その他利用可能、保留状態、取り消し済み)を指定する必要があります。 **注:**厳密な状態遷移表はありません。フェールオーバー環境では、バインド更新メッセージを受信するサーバーは、特定の状態遷移を必要とせずに、パートナーが通知する通知に状態を設定します。
拡張拡張-拡張-バージョン(出力)	拡張機能に必要な拡張機能 API の最小バージョン。現在の**2**API バージョンに設定します。
イニット引数(入力)	既存の拡張ポイントに*init-args*を設定して、引数を構成します。これらの引数は、エントリ ポイントの設定呼び出しと**init-entry**構成解除呼び出しの両方に対して存在します。 構成呼び出しの拡張ポイント名**initialize**は です。 **uninitialize**

環境ディクショナリデータ項目	説明
サーバー *dhcp* サポート(入力)	サーバーは、このデータ項目を設定して、サーバーがサポートするように構成されていることを示します。DHCP**v4v6**サーバー DHCP **v4,v6** サポート属性設定 (エキスパート属性の可視性が 3 の設定が必要) と、プレフィックスがコンフィグレーションされているかどうかに応じて、、または 、、または 、以下を指定できます。 • *dhcp* サポート=**both**およびプレフィックスが構成されていない場合、サーバー*-dhcp* サポートは に**v4**設定されます。 • *dhcp* サポート=**both**で 1 つ以上のプレフィックスが設定されている場合、サーバー DHCP**v4,v6**サポートは に設定されます。 • *dhcp* サポート=**v4**の場合、サーバー *dhcp* サポートは**v4**に設定されます。 • *Dhcp* サポート= と1つ以上のプレフィックスが設定されている場合 **v6** 、サーバー *dhcp* サポートはに設定され **v6** ます。
サーバー拡張-拡張-バージョン(入力)	サーバー拡張 API のバージョン。値は**2**です。

事前パケットデコード

pre-packet-decode で使用可能なディクショナリは、要求と環境です。

この拡張ポイントは、要求が到着したときにDHCPサーバーが最初に検出したポイントです。サーバーは、パケットを受信した後、(拡張ポイントで)パケットをデ**post-packet-decode**コードする前に呼び出します。拡張機能は、この拡張ポイントを使用してパケットを検査し、サーバーがデコードする前にパケットを変更したり、サーバーにドロップを発生させることができます。

要求ディクショナリの2つの主要なデータ項目は**pre-packet-decode**、クライアントパケットとパケットです。これらは、受信したパケットを調べて、パケットを変更し、それを書き戻すために使用できます。

⚠

注意 要求ディクショナリクライアントパケットとパケットデータアイテム**pre-packet-decode** は、要求ディクショナリを持つ任意の拡張ポイントで使用できます。ただし、パケットを変更したり、**pre-packet-decode**以外の拡張ポイントで置き換えたりすることは、予期しない副作用を引き起こす可能性があるため、直接変更したり、置き換えたりしないでください。たとえば、サーバーがパケットに対する変更を取得しない場合や、処理中にオプション データが予期せず変更される場合があります。

getBytes をクライアントパケットまたはパケットで使用する拡張は、返されたバッファに書き込みによってパケットのバイト数を直接変更します。ただし、拡張子は**put**または**putBytes**を使用してパケットの長さを調整する必要があります (パケットが大きすぎる場合は操作が失敗す

る可能性があります)。DHCPv6の場合、パケットのクライアント部分の長さを調整する場合、リレーされる場合は、パケット内のリレー メッセージ オプションの長さを更新する必要があります。

パケットの解析を処理して必要なものを見つけ、意図している場合はパケットを適切に変更する必要があります。

サーバーは受信パケットをまだデコードしていないため、ほとんどの要求ディクショナリデータ項目は使用できません(通常は、サーバーが受信パケットからパケットを入力するため)。したがって、この拡張ポイントはパケットから直接データを抽出する必要があります。また、拡張子は正しくフォーマットされていないパケットを処理する必要があります。

着信パケット詳細ログを有効にすると、サーバーはこの拡張ポイントで登録された拡張機能を呼び出した後に受信パケットをログに記録します。DHCPサーバーのデバッグ トレースが3以上の場合、少なくとも1つの拡張機能が登録されている場合、サーバーはこの拡張ポイントに登録されている拡張機能を呼び出す前にパケットもログに記録します。

⚠

注意 この拡張は、受信したパケットが何らかの方法で検証される前に、そのパケットにアクセスします。したがって、拡張は、完全または部分的に無効なDHCPパケットを処理するように記述する必要があります。

post-packet-decode

post-packet-decode で使用可能なディクショナリは、要求と環境です。

拡張の説明

この拡張ポイントは、入力パケットのデコードの直後に続き、パケット内のデータに対する処理の前に行われます。この時点での拡張機能の主なアクティビティは、入力パケットから情報を読み取り、それを使用して何かを行う操作です。たとえば、入力パケットを書き換えるために使用できます。

post-packet-decode拡張ポイントは、使用する最も簡単な拡張ポイントの1つです。入力DHCPまたはBOOTPパケットの書き換えとしてサーバーの動作の変更を表現できる場合は、この拡張ポイントを使用する必要があります。パケットはデコードされたが、処理されていないため、副作用の数は非常に限られています。

post-packet-decodeデコードされた入力パケットを変更し、サーバーがすべての変更を認識できるようにする唯一の拡張ポイントです。拡張機能でパケットをドロップし、環境ディクショナリのドロップデータ項目を使用して、さらに処理を終了させることができます。

クライアントIDの上書き

クライアント識別子 (ID) をオーバーライドするには、クライアント クラスのオーバーライド-クライアント*ID*属性の式の値を設定するか、拡張ポイントでオーバーライドクライアント*ID*データ項目を**post-packet-decode**使用します。拡張メソッドは、クライアントをサーバーとは異なる識別子にマップします。

オーバーライド クライアント ID を文字列として取得または配置できる拡張データ項目のバリエーションがあります。また、オーバーライドクライアント ID のデータ型は、読み取り専用のオーバーライドクライアント *ID* データ型データ項目を使用して要求することもできます。

オーバーライドクライアント ID またはそのオーバーライドクライアント*ID* 文字列バリアントの書き込み方法と取得方法に基づいて、異なる値が返されます (いくつかの例については、次の表を参照してください)。

表 *7:* クライアント *ID* の上書きの書き込みと取得

操作	使用されるデータ項目	値を入れる	値の取得
put	*override-client-id*	01:02:03:04	
putBytes	*override-client-id*	01 02 03 04	
get	*override-client-id*		01:02:03:04 (ブロブ)
getBytes	*override-client-id*		01 02 03 04 (生バイト)
get [**Bytes**]	オーバーライドクライアント *ID* 文字列		01:02:03:04 (blob-as-string)
get [**Bytes**]	オーバーライドクライアント *ID* データ型		blob

表 *8:* クライアント *ID* の上書きの書き込みと取得

操作	使用されるデータ項目	値を入れる	値の取得
put [**Bytes**]	オーバーライドクライアント *ID* 文字列	01:02:03:04 テスト	
get [**Bytes**]	オーバーライドクライアント *ID* 文字列		01:02:03:04 テスト (文字列)
get [**Bytes**]	*override-client-id*		30:31:3a:30:32:3a:30:33:3a:30:34:74:65:73:74 (「01:02:03:04 テスト」のブロブ)
get [**Bytes**]	オーバーライドクライアント*ID*データ型		nstr

同等のクライアントオーバーライドクライアント*ID*データ項目 (応答ディクショナリが有効な後の拡張ポイントで使用できます) は、読み取り専用ですが、同じように機能します。

（注） *[v6-]* オーバーライドクライアント *ID*式を使用する場合、クライアント *ID*による leasequery 要求は、クライアントのリースに関する情報を正しく取得するために、オーバーライドクライアント *ID*属性を指定する必要があります。

注意 この拡張は、サーバーがパケット構文解析を行った後、検証が適用される前に呼び出されます。したがって、潜在的に無効なパケットを処理するために拡張機能を作成する必要があります。

post-packet-decode の環境ディクショナリ

パケット デコード後に固有の環境ディクショナリ データ項目については、次の表を参照してください。

表 *9:* ポスト パケット デコード環境ディクショナリ データ項目

環境ディクショナリ データ項目	説明
cnr-転送-*dhcp* 要求(入力)	これらのデータ項目はいずれも DHCPv4 専用です。拡張が戻るときに *cnr-forward-dhcp-request*が**true**に設定されている場合、*cnr-request-forward-address-list*には、サーバーが要求を転送する IPv4 アドレス (およびオプションでポート番号) のリスト (コンマ区切り) を含める必要があります。転送されると、サーバーは要求を破棄します。コンマ区切りリストの各エントリは、*ipv4-address*または*ipv4-address:*ポート番号(ポート番号が指定されていない場合は、デフォルトの dhcp サーバー ポートが使用されます) です。詳細については、DHCP 転送の設定を参照してください。
cnr 要求-転送アドレスリスト(出力)	

ポストクラスルックアップ

使用できる辞書**post-class-lookup**は、要求と環境です。

サーバーはこの拡張ポイントを呼び出す場合、クライアント クラスルックアップ *ID*が存在する場合に限ります。それ以外の場合は、**post-packet-decode**に似ています。サーバーは、クライアント**post-class-lookup**クラスルックアップ *ID*を評価し、このクライアントのクライアント クラスデータを設定した後、拡張ポイントを呼び出します。

この拡張ポイントへの入力時に、環境ディクショナリのドロップデータ項目が**true**または**false**に設定されます。この設定を変更して、パケットをドロップ(またはドロップしない)に変更すると、サーバーは変更を認識します。サーバーは、ログ ドロップ メッセージを調べ、ドロップをログに記録するかどうかを決定します。

post-class-lookup の環境ディクショナリ

クラスルックアップ後に固有の環境ディクショナリデータ項目については、次の表を参照してください。

表 *10*：ポストクラスルックアップ環境辞書データ項目

環境ディクショナリ データ項目	説明
client-class-name（出力）	以前のクライアントクラスに関係なく、パケットの名前付きクライアントクラスを設定します。この設定は、拡張ポイントの終了時にドロップ環境ディクショナリデータ項目の値が**false**の場合にのみ有効です。

pre-client-lookup

pre-client-lookup で使用可能なディクショナリは、要求と環境です。

この拡張ポイントは、DHCP サーバーに対してクライアントクラス処理を有効にしている場合にのみ使用できます。この拡張ポイントを使用すると、拡張機能で次のアクションの一部またはすべてを実行できます。

- クライアント クラスの処理中にサーバーが検索するクライアントを変更します。
- 個別のデータ項目を指定して、指定したクライアント・エントリーまたはクライアント・クラスから見つかったデータ項目をオーバーライドします。
- クライアントの検索を完全にスキップするようにサーバーに指示します。この場合、使用されるクライアントデータは、環境ディクショナリで提供される拡張機能のデータだけです。

要求ディクショナリは、この拡張ポイントで実行されている拡張機能の操作に関する決定を行うために使用できますが、環境ディクショナリはすべての操作を制御します。

pre-client-lookup の環境ディクショナリ

次の表の環境ディクショナリ データ項目は、クライアントおよびクライアント クラス **pre-client-lookup**で使用できるコントロール データ項目です。

表 12：事前クライアントルックアップ環境ディクショナリがデータ項目をオーバーライドする で環境ディクショナリ データ項目を設定すると、その値はクライアントルックアップ (内部データベースまたは LDAP のいずれか) から決定された値よりも優先されます。ディクショナリに何も追加しない場合、サーバーはクライアントルックアップで使用可能な内容を使用します。

表 *11* : 事前クライアントルックアップ環境辞書コントロールデータ項目

環境ディクショナリ データ項目	説明
クライアント指定子(入力/出力)	クライアント クラスの処理コードが CNRDB または LDAP でルックアップするクライアントの名前。この拡張ポイントで名前を変更すると、DHCP サーバーは指定したクライアントを検索します。
デフォルトクライアントクラス名(出力)	次の場合に、デフォルトクライアントクラス名オプションに関連付けられた値をクラス名として使用するようにサーバーに指示します。 • **pre-client-lookup**クライアント指定子のデータ項目がスクリプトで指定されていません。 • サーバーは特定のクライアントエントリを見つけることができなかった。 その後、*default-client-class-name*データ項目は、デフォルトクライアントに関連付けられたクラス名よりも優先されます。
スキップクライアントルックアップ(入力/出力)	値は、サーバー構成によって決まります。に**true**設定すると、DHCP サーバーは、この拡張機能の終了時にすぐに実行される通常のクライアントルックアップをスキップします。 このクライアントを記述するために使用されるデータ項目は、環境ディクショナリに置かれるものだけです (下の表を参照)。

表 *12* : 事前クライアントルックアップ環境ディクショナリがデータ項目をオーバーライドする

環境データ項目	説明
アクション(出力)	この文字列を数値に変換し、結果をアクションとして使用します。使用できる数値は**0**(なしの場合)、(**1**除外の場合) です。
認証終了(出力)	1970 年 1 月 1 日からの絶対時間 (秒単位)。クライアント認証の有効期限を示すために使用します。 クライアント認証の有効期限が切れると、DHCP サーバーはクライアント クラスではなく、クライアントの認証されていないクライアント クラス オプションの値を使用して、クライアント エントリに不足しているデータ項目を入力します。
クライアント クラス名(出力)	このデータ項目で指定されたクライアント クラスを使用して、クライアント エントリの不足している情報を入力します。指定された名前に対応するクライアント クラスがない場合、DHCP サーバーは警告をログに記録し、処理を続行します。 を指定**none**すると、DHCP サーバーはクライアント エントリにクライアント クラス名が含まれていないかのように動作します。

環境データ項目	説明
ドメイン名(出力)	このドメイン名は、DNS 更新の構成で指定されたクライアント DNS 操作よりも優先して使用します。スコープまたはプレフィックスのドメインのプライマリ サーバーとして表示される DNS サーバーは、指定したドメインのプライマリ サーバーである必要があります。 クライアントまたはクライアント クラスのエントリでドメイン名が上書きされない場合、DHCP サーバーはスコープまたはプレフィックスのドメイン名を使用します。 クライアント エントリまたは拡張機能に という単語**none**が含まれている場合、DHCP サーバーはスコープまたはプレフィックスのドメイン名を使用します。
ホスト名(出力)	入力パケットで指定されたホスト名オプション、またはクライアントまたはクライアントクラスエントリからのデータに優先して、クライアントに対してこれを使用します。 これをに**none**設定すると、DHCP サーバーはクライアントまたはクライアント クラスのエントリからの情報を使用せず、クライアント要求の名前を使用します。
ポリシー名(出力)	このポリシーは、クライアント エントリに指定されたポリシーとして使用し、そのクライアントエントリで指定されたポリシーを上書きします。
selection-criteria（出力）	コンマ区切りの文字列のリストで、各文字列はクライアントの選択基準を指定します(この入力パケットに対して)。クライアントが使用するスコープまたはプレフィックスには、これらの選択タグがすべて含まれる必要があります。 このデータ項目を使用して、クライアントまたはクライアント クラスのエントリで指定された条件をオーバーライドします。この場合、DHCP サーバーは、ローカル データベースまたは LDAP データベースのいずれに格納されているかに関係なく、クライアント エントリの選択基準を使用しません。 このデータ項目を に**none**設定すると、DHCP サーバーはパケットの選択タグを使用しません。 これを null 文字列に設定すると、DHCP サーバーは設定されていないものとして扱い、クライアントまたはクライアント クラスエントリの選択基準を使用します。

環境データ項目	説明
認証されていないクライアントクラス名(出力)	サーバーがクライアントを認証しない場合に使用するクライアント クラスの名前。認証されていないクライアント クラス名を指定せずに指定する場合は、このデータ項目の値として無効なクライアント クラス名を使用します。 値**none** またはクライアント クラス名以外の任意の名前を使用できます。DHCP サーバーは、クライアント クラスが存在しないエラーをログに記録します。

ポスト クライアント ルックアップ

post-client-lookup で使用可能なディクショナリは、要求と環境です。

この拡張ポイントを使用して、クライアント クラスの処理操作全体の結果を調べ、その結果に基づいてアクションを実行できます。結果の一部を書き換えたり、パケットをドロップしたりするために使用できます。**post-client-lookup**クライアントクラスの処理から、拡張ポイントで実行されている拡張から返されるパケットのホスト名をオーバーライドする場合は、要求ディクショナリ内のクライアントが要求したホスト名データ項目に hostname を設定します。これにより、Cisco Prime Network レジストラーは、そのデータ項目で指定した文字列でパケットが入ってきたかのように、サーバーを検索します。

また、この拡張ポイントを使用して、環境ディクショナリにデータ項目を配置して、**pre-packet-encode**拡張ポイントで実行されている拡張の処理に影響を与pre-packet-encode （46 ページ）えることができます (を参照) 、 応答パケットに異なるオプションを読み込む場合や、その他のアクションを実行する可能性があります。

post-client-lookup の環境ディクショナリ

に固有の環境ディクショナリ データ項目については、次の**post-client-lookup**表を参照してください。

表 *13* : ポスト クライアントルックアップ環境ディクショナリ データ項目

環境ディクショナリ データ項目	説明
クライアント指定子(入力)	クライアント クラスの処理が検索したクライアントの名前。

環境ディクショナリ データ項目	説明
cnr-ldap クエリに失敗しました(入力)	DHCP サーバーは、クライアント参照後スクリプトが LDAP サーバー障害に対応できるように、LDAP サーバー障害からの回復を容易にするためにこの属性を設定します。 クライアントルックアップ後の DHCP サーバーは、LDAP サーバー **true** ・エラーが原因で LDAP 照会が失敗した場合にこのフラグを設定します。サーバーが LDAP サーバーから応答を受信した場合、次の 2 つの条件のいずれかが発生します。 • フラグを に設定**false**します。 • *cnr-ldap-query-failed* 属性は、環境ディクショナリに表示されません。

リースの生成

generate-lease で使用可能なディクショナリは、要求、応答、および環境です。この拡張ポイントは、DHCPv6 でのみ使用できます。

この拡張ポイントを使用して、DHCPv6アドレスまたはプレフィックスを生成し、拡張機能でアドレスまたはプレフィックスを制御できます。拡張機能が生成されたアドレス値を返すとき、サーバーは、エクステンションがリースアクティビティを制御していると仮定して、返されるアドレスまたはプレフィックスに対する多くの制限を緩和します。これには、フェールオーバーの制約が含まれます (したがって、奇数アドレスはバックアップによって割り当てることができ、偶数アドレスはメインによって割り当てることができ、他に利用可能なデリゲートされたプレフィックスを割り当てることができます)。拡張機能は、アドレスまたはプレフィックスの委任領域を管理します。

アドレス割り**generate-lease**当てまたはプレフィックスの委任時に拡張機能を呼び出すことを許可するようにプレフィックスが構成されている場合にのみ、サーバーが呼び出します。サーバーがリースの生成拡張機能を呼び出すと、次のようになります。

- サーバーは、応答ディクショナリのプレフィックスコンテキストを、リースが作成されるプレフィックスに設定します。(DEX_PREFIX**setObject**とDEX_INITIALを使用して呼び出すと、このコンテキストに戻ります。
- サーバーがまだリースを作成していないため、リース コンテキストは存在しません。しかしながら、リース結合データ項目、特にリース結合タイプおよびリース結合*iaid*は利用できる。(DEX_LEASE**setObject**とDEX_INITIALを使用して呼び出すと、リース コンテキストは 3 つのコンテキスト(リース、バインディング、およびプリフィックス)を設定するため、このコンテキストに戻り、プレフィックスも設定します。
- サーバーは、スキップリース環境ディクショナリデータ項目を false に設定します。
- サーバーは、(読み取り専用の)環境ディクショナリデータ項目を、このリースを作成するためにエクステンションを呼び出した回数 (1 から始まる) に設定します。
- プレフィックスの委任では、次の環境ディクショナリ データ項目を使用できます。

- **prefix-length-** プレフィックスの長さ (要求されたプレフィックス長またはデフォルトのプレフィックス長)。
- **default-prefix-length**— デフォルトのプレフィックス長(ポリシーからの)
- **longest-prefix-length**— 許容される最長プレフィックス(ポリシーから)
- **shortest-prefix-length**— (ポリシーからの) 最短許容プレフィックス。

拡張機能が返されるときに、次のことができます。

- 生成されたアドレス環境ディクショナリ データ項目にアドレスを設定して、明示的なアドレス(ステートフルアドレス割り当て用)を要求します。クライアントのアドレスが使用できない場合(つまり、アドレスが既に使用中の場合)、またはプレフィックスに含まれていない場合、サーバーはこの拡張機能を再度呼び出す可能性があります。
- 生成されたプレフィックス環境ディクショナリ データ項目にプレフィックスを設定して、明示的なプレフィックス(プレフィックス委任の割り当て)を要求します。クライアントに対してプレフィックスが使用できない場合、またはプレフィックスに含まれていない場合、サーバーはこの拡張機能を再度呼び出す可能性があります。次の条件の場合、クライアントではプレフィックスを使用できません。
  - プレフィックスが既に使用されている場合
  - 既に委任されている短いプレフィックスに含まれている場合
  - それより長いプレフィックスが既にサーバーによって委任されている場合

  ポリシーで許可されているプレフィックスが短い場合、または長い場合、サーバーはプレフィックスを拒否しません。
  - スキップリース環境ディクショナリデータ項目をtrueに設定して、サーバーがこのプレフィックスのリースを割り当てないようにします。サーバーは次のプレフィックス(存在する場合) に進みます。
  - 上記のいずれかを設定しないことで、通常のアドレス割り当てまたはプレフィックスの委任を許可します。

サーバーは、各リースに対して最大 500 回の拡張ポイントを呼び出します (この制限は、サーバーがランダムにリースを生成するときに現在適用される制限と同じです)。サーバーは、使用できないアドレスまたはデリゲートされたプレフィックス (プレフィックスの範囲外または既に存在する) を提供する場合にのみ、拡張機能を複数回呼び出します。

（注） この拡張ポイントでパケットをドロップするようサーバーに要求することはできません。

generate-lease の環境ディクショナリ

リースの生成に固有の環境ディクショナリ データ項目については、次の表を参照してください。

表 *14 :* リース環境ディクショナリ データ項目の生成

環境ディクショナリ データ項目	説明
試み *(*入力)	サーバーが単一のリースに対してこの内線番号を呼び出す回数。
デフォルトプレフィックス長(入力)	デリゲートされたプレフィックスの割り当てに使用する既定のプレフィックス長を指定します。デフォルトのプレフィックス長に設定します (ポリシー階層から)。
生成アドレス(出力)	サーバーがリースに使用する拡張機能をアドレス指定します。
生成されたプレフィックス(出力)	委任されたDHCPv6プレフィックスは、サーバーがリースに使用する拡張機能を必要とします。
プレフィックスの長さ(入力) に制限	生成されたプレフィックスをクライアントが要求した**プレフィックス長プレフィックス長のプレフィックス長**に制限するようにサーバーが拡張を要求している場合は*true*に設定します。それ以外**の場合は false。**クライアントがプレフィックス長を要求した場合、サーバーは最初に、その長さのデリゲートされたプレフィックスを取得しようとして**、リースの生成**エクステンションを呼び出します。サーバーは、クライアントの要求された長さを最短プレフィックス長と最長プレフィックス長の間に制限することに注意してください。
最長プレフィックス長(入力)	デリゲートされたプレフィックスの割り当てに使用する最長のプレフィックス長を指定します。(Expert モード)の最長プレフィックス長(ポリシー階層からの) に設定します (ポリシー階層から) - デフォルトはデフォルトのプレフィックス長(未設定の場合) に設定されます。
prefix-length（入力）	要求されたプレフィックス長または既定のプレフィックス長に設定します。
最短プレフィックス長(入力)	デリゲートされたプレフィックスの割り当てに使用する最短のプレフィックス長を指定します。(expert モード) の最短プレフィックス長(ポリシー階層から) に設定します (ポリシー階層から) - デフォルトはデフォルトのプレフィックス長(未設定の場合) に設定されます。
スキップリース(出力)	拡張機能がサーバーにリースを生成させたくない場合は、**true**に設定します。

check-lease-acceptable

使用できる辞書**check-lease-acceptable**は、要求、応答、および環境です。

この拡張ポイントは、現在のリースがこのクライアントに対して許容されるかどうかをサーバーが判断した直後に取得されます。この拡張機能を使用すると、その操作の結果を調べ、ルーチンが異なる結果を返すようにすることができます。リースの受け入れの決定 （19 ページ） を参照してください。

check-lease-acceptable の環境ディクショナリ

チェック-リース許容に固有の環境ディクショナリ データ項目については、次の表を参照してください。

表 *15*：チェックリース許容環境辞書データ項目

環境ディクショナリ データ項目	説明
受け入れ可能(入力)	このクライアントでリースが受け入れられるかどうかに応じて、DHCP サーバーが初期化する読み取り/書き込みデータ項目。この結果を読み取り、変更することができます。許容データ項目を**true**に設定すると、それが許容可能であることを示します。**false**に設定すると、受け入れられません。
デフォルトプレフィックス長(入力)	デリゲートされたプレフィックスの割り当てに使用する既定のプレフィックス長を指定します。デフォルトのプレフィックス長に設定します (ポリシー階層から)。
最長プレフィックス長(入力)	デリゲートされたプレフィックスの割り当てに使用する最長のプレフィックス長を指定します。(Expert モード)の最長プレフィックス長(ポリシー階層からの) に設定します (ポリシー階層から) - デフォルトはデフォルトのプレフィックス長(未設定の場合) に設定されます。
prefix-length（入力）	クライアントが 1 を指定した場合は、クライアントが要求したプレフィックス長を指定します。
最短プレフィックス長(入力)	デリゲートされたプレフィックスの割り当てに使用する最短のプレフィックス長を指定します。(expert モード) の最短プレフィックス長(ポリシー階層から) に設定します (ポリシー階層から) - デフォルトはデフォルトのプレフィックス長(未設定の場合) に設定されます。

リース状態の変更

使用できる辞書**lease-state-change**は応答と環境です。

既存の状態は、リース状態応答ディクショナリ データ項目にあります。新しい状態は、環境ディクショナリ データ項目の新しい状態にあります。新しい状態が既存の状態と一致する場合、サーバーは拡張ポイントを呼び出しません。

この拡張ポイントは、主に読み取り専用の目的で使用しますが、他の拡張ポイントが後で取得できるように、環境ディクショナリにデータを配置できます。

また**lease-state-change**、リースの有効期限など、別の環境ディクショナリを持つことができます。

lease-state-change の環境ディクショナリ

リース状態の変更に固有の環境ディクショナリ データ項目については、次の表を参照してください。

表 *16 :* リース状態変更環境ディクショナリ データ項目

環境ディクショナリ データ項目	説明
新しい開始時の状態(入力)	新しい状態の開始時刻。前の状態の開始時刻は、応答ディクショナリのリース開始状態情報データ項目にあります。
新しい状態(入力)	リースの変更後の状態。現在の状態は、応答ディクショナリのリース状態リース情報データ項目にあります。

pre-packet-encode

pre-packet-encode で使用可能なディクショナリは、要求、応答、および環境です。

（注） DHCPv6 再設定メッセージの場合、要求ディクショナリはありません (再構成はサーバーによって開始されたメッセージであるため)。したがって、有効になっている拡張機能は、応答*msg* タイプの ADVERTISE または**isValid**REPLY を調べるか、要求で再設定メッセージが存在することを確認するために使用する必要があります。

ポスト パケット エンコード

post-packet-encode で使用可能なディクショナリは、要求、応答、および環境です。

（注） DHCPv6 再設定メッセージの場合、要求ディクショナリはありません (再構成はサーバーによって開始されたメッセージであるため)。したがって、有効になっている拡張機能は、応答 msg タイプを調べて、または**isValid**要求に対して応答メッセージタイプをチェックして、要求ディクショナリが存在することを確認します。

サーバーは、パケットをエンコードした後、クライアントに送信する前に、この拡張ポイントを呼び出します。これにより、サーバーはパケットをクライアントに送信する前にパケットを検査して変更するか、拡張機能によってサーバーがパケットをドロップする可能性があります (ただし、サーバは内部データとディスク上のデータに変更を加えた可能性があります。六色)。

クライアント パケットおよびパケットデータ項目が、 の要求ディクショナリで説明されているような動作で応答ディクショナリに事前パケットデコード （34 ページ）追加されました。この拡張ポイントは、応答クライアントパケットまたはパケットデータ項目を要求できる唯一のポイントであることに注意してください。また、サーバーはパケットに対して行われた変更を処理しません。サーバは、変更されたパケットをクライアントに送信するだけです。

発信パケット詳細ログを有効にすると、サーバーはこの拡張ポイントで登録された拡張機能を呼び出した後にパケットをログに記録します。DHCP サーバーのデバッグ トレースが X>=3 で設定されている場合、サーバーは、この拡張ポイントに登録されている拡張機能を呼び出す前に、少なくとも 1 つの拡張機能が登録されている場合にのみ、パケットをログに記録します。

ポスト送信パケット

DHCP**post-send-packet**要求/応答サイクルの重大な時間制約の外部で実行する処理には、拡張ポイントを使用します。サーバーは、クライアントにパケットを送信した後、この拡張ポイントを呼び出します。

（注） DHCPv6 再設定メッセージの場合、要求ディクショナリはありません (再構成はサーバーによって開始されたメッセージであるため)。したがって、有効になっている拡張機能は、応答 msg タイプを調べて、または**isValid**要求に対して応答メッセージタイプをチェックして、要求ディクショナリが存在することを確認します。

環境デストラクタ

拡張**environment-destructor**ポイントを使用すると、拡張は、そのエクステンションが保持している可能性のあるコンテキストをクリーンアップできます。この拡張ポイントで使用できる唯一のディクショナリは環境です。

環境ディクショナリは、単一のクライアント要求に対して呼び出されるすべての拡張ポイントで使用できます。一部の拡張機能では、単一のクライアント要求のために呼び出される複数の拡張ポイント間のコンテキスト情報を維持する必要があり、サーバーが処理中に複数の場所で要求をドロップする可能性があるため、拡張機能を確実に解放できないその要求に対して作成した可能性のあるコンテキスト。環境デストラクター拡張ポイントにより、何らかの理由で要求の処理が完了したときに、このコンテキストを確実に削除できるようになりました。

（注） サーバーは、他の接続ポイントで**environment-destructor**各拡張機能を呼び出さなかった場合でも、拡張ポイントに接続されているすべての拡張機能を呼び出します。

翻訳について
このドキュメントは、米国シスコ発行ドキュメントの参考和訳です。リンク情報につきましては、日本語版掲載時点で、英語版にアップデートがあり、リンク先のページが移動/変更されている場合がありますことをご了承ください。あくまでも参考和訳となりますので、正式な内容については米国サイトのドキュメントを参照ください。