



## 使用する前に

- [StarOS の初期設定](#) (1 ページ)
- [StarOS CLI を使用した初期設定](#) (1 ページ)
- [システム管理ユーザーの設定](#) (4 ページ)
- [リモートアクセス用のシステムの設定](#) (5 ページ)
- [SSH オプションの設定](#) (8 ページ)
- [2 番目の IP アドレスを使用した管理インターフェイスの設定](#) (20 ページ)
- [Open SSH から Cisco SSH へのアップグレードと移行](#) (21 ページ)
- [VM ハードウェアの検証](#) (23 ページ)

## StarOS の初期設定

すべての VM での VPC-DI のインストールが正常に完了したら、アクティブな制御機能 (CF) VM を使用して StarOS パラメータのセットを設定する必要があります。その後で、VPC-DI インスタンス内の VM が再起動されるたびにアクセスされる、アクティブな CF 上の設定ファイルにこれらの設定を保存します。スタンバイ CF およびすべてのサービス機能 (SF) VM は、この設定ファイルをアクティブ CF から読み取ります。

CUPS を搭載した UGP の場合は、USP VM にログインして StarOS の一連のパラメータを設定する必要があります。これらの設定は設定ファイルに保存され、VM が再起動されるたびに、CPF VNFC VM に送信されます。

この章では、アクティブな CF コンソールポートに接続し、最初のローカルコンテキスト管理設定を作成する手順について説明します。

## StarOS CLI を使用した初期設定

初期設定は次のように構成されています。

- コンテキストレベルのセキュリティ管理者とホスト名の設定
- vNIC でのイーサネット インターフェイスの設定

- SSHを介したリモート CLI アクセスのための VPC-DI インスタンスの設定

この項では、CLI を使用してこれらのタスクを実行するための手順を説明します。

**ステップ1** ハイパーバイザを介して、アクティブな CF VM のコンソールポートにログインします。

**ステップ2** CLI プロンプトで、次のように入力します。

```
[local]cf_host_name configure[local]cf_host_name(config)
```

**ステップ3** 次のコマンドを入力してコンテキスト コンフィギュレーション モードを開始します。

```
[local]cf_host_name(config) context local[local]cf_host_name(config-ctx)
```

ローカルコンテキストは、VPC-DI インスタンスの管理コンテキストです。コンテキストを使用すると、サービスまたはインターフェイスを論理的にグループ化することができます。1つのコンテキストは複数のサービスで構成でき、複数のインターフェイスにバインドできます。

**ステップ4** コンテキストレベルのセキュリティ管理者を VPC-DI インスタンスに設定するには、次のコマンドを入力します。

```
administrator user_name [ encrypted ] password password
| [ ecs ] [ expiry-date date_time ] [ ftp ] [ li-administration ] [ nocli ] [ noecs ]
|
|
```

(注) 初期設定時にコンテキストレベルのセキュリティ管理者を設定する必要があります。初期設定プロセスが完了し、CLI セッションを終了した後、セキュリティ管理者が設定されていない場合は、CLI アクセスがロックされます。このコマンドの詳細については、『*Command Line Interface Reference*』の「*Context Configuration Mode Commands*」の章を参照してください。

**ステップ5** プロンプトで次のコマンドを入力して、コンテキストのコンフィギュレーション モードを終了します。

```
[local]cf_host_name(config-ctx) exit
[local]cf_host_name(config)
```

**ステップ6** 次のコマンドを入力して、VPC-DI インスタンスがネットワーク上で認識されるホスト名を設定します。

```
[local]cf_host_name(config) system hostname cf_host_name
```

*cf\_host\_name* は、VPC-DI インスタンスがネットワーク上で認識される名前です。ホスト名は、大文字と小文字が区別される 1 ~ 63 文字の英数字文字列です。デフォルトのホスト名は「qvmc-di」です。

**ステップ7** vNIC 上のネットワーク インターフェイスを次のように設定します。

a) 次のコマンドを入力して、コンテキスト コンフィギュレーション モードを開始します。

```
[local]cf_host_name(config) context local
[local]cf_host_name(config-ctx)
```

b) インターフェイスの名前を指定するには、次のコマンドを入力します。

```
[local]cf_host_name(config-ctx) interface interface_name
```

*interface\_name* は、大文字と小文字が区別される 1～79 文字の英数字の文字列で表されるインターフェイスの名前です。StarOS がイーサネットインターフェイスのコンフィギュレーションモードを開始すると、次のプロンプトが表示されます。

```
[local]cf_host_name(config-if-eth)
```

- c) 次のコマンドを入力して、前のステップで設定したインターフェイスの IP アドレスを設定します。

```
{ ip address | ipv6 address } ipaddress subnetmask
```

(注) クイックセットアップウィザードで誤って設定されたアドレスまたはサブネットを修正するためにこのコマンドを実行する場合は、デフォルトルートとポートバインドの設定を確認する必要があります。この手順のステップ 11 とステップ 6 を使用します。問題がある場合は、ステップ 7e～7k を実行して情報を再設定します。

- d) 次のコマンドを入力して、イーサネットインターフェイスのコンフィギュレーションモードを終了します。

```
[local]cf_host_name(config-if-eth) exit  
[local]cf_host_name(config-ctx)
```

- e) 必要に応じてスタティックルートを設定して、VPC-DI インスタンスをデフォルトゲートウェイに指定します。次のコマンドを入力します。

```
{ ip | ipv6 } route gw_address interface_name
```

- f) コンテキストのコンフィギュレーションモードを終了するには、次のように入力します。

```
[local]cf_host_name(config-ctx) exit  
[local]cf_host_name(config)
```

- g) イーサネットポートのコンフィギュレーションモードを開始します。

```
[local]cf_host_name(config) port ethernet slot/port
```

VPC-DI の場合、*slot* は仮想シャーシ内の CF または SF VM に対応します。ハイパーバイザは VPC-DI インスタンスの初期設定時に、各 VM に一意のスロット番号を割り当てます。スロット番号 1 と 2 は CF VM に割り当てられ、スロット番号 3～32 は SF VM に割り当てられます。CF はポート 1 のみをサポートします。各 SF は、1～4 番の 4 つの vNIC をサポートし、対応する仮想イーサネットポート 10～14 番を使用します。SF ポート番号 10 を設定する必要があります。

- h) ステップ 7b で作成したインターフェイスにポートをバインドします。バインドにより、ポートとそのすべての設定がインターフェイスに関連付けられます。次のコマンドを入力します。

```
[local]cf_host_name(config-port-slot/port) bind interface interface_name local  
[local]cf_host_name(config-port-slot/port) no shutdown
```

*interface\_name* は、ステップ 7b で設定したインターフェイスの名前です。

- i) 次のコマンドを入力して、イーサネットインターフェイスのコンフィギュレーションモードを終了します。

```
[local]cf_host_name(config-port-slot/port) exit  
[local]cf_host_name(config)
```

- (注) 管理ポートは、VLANもサポートしています。詳細については、「インターフェイスとポート」の章の「VLAN」の項を参照してください。

## システム管理ユーザーの設定

この項では、セキュリティ管理者がユーザーアカウントを制御できるようにするセキュリティ機能の一部について説明します。

### 同時 CLI セッション数の制限

セキュリティ管理者は同時対話型 CLI セッションの数を制限できます。同時対話型セッションの数を制限すると、システム全体のリソースの消費量が削減されます。また、ユーザーがすでに使用されている機密ユーザー情報にアクセスする可能性を防ぎます。

ほとんどの特権アカウントでは、複数の同時ログインは必要ありません。



- (注) 21.9以降のリリースでは、1つのCLIセッションでの複数のチャンネルはサポートされていません。



**重要** すべての特権アカウントには、セッションの最大数を設定することを推奨します。

セキュリティ管理者は、その特定のユーザーアカウントに使用される認証方式に応じて、3つの異なる方法で同時インタラクティブ CLI セッションの数を制限できます。

StarOS は次の3つのログイン認証方式をサポートしています。

- TACACS+ サーバーユーザー
- ローカルユーザーのユーザー
- AAA コンテキストユーザー

TACACS+ サーバーユーザーの最大セッション数の設定の詳細については「[動作](#)」を参照してください。ローカルユーザーのユーザーと AAA コンテキストユーザーの最大セッション数の設定の詳細については「[Configuring Context-Level Administrative Users](#)」を参照してください。

各認証方式は、3つの認証方式のそれぞれが同じユーザー名を使用できるため、個別に設定する必要があります。

## CLI セッションの自動ログアウト

セキュリティ管理者は、特定のユーザーアカウントの自動ログアウトを設定できます。対話型 CLI セッションが使用可能な時間を分単位で制限すると、システム全体のリソースの消費量が削減されます。また、アイドル状態のままになっている端末ウィンドウで、ユーザーがユーザーアカウントにアクセスする可能性を防ぐこともできます。この項で説明されているすべての認証方式は、アイドルセッションタイムアウトの手法と絶対セッションタイムアウトの手法の両方をサポートしています。

ほとんどの特権アカウントは、無期限のログインタイムアウトの制限を必要としません。



**重要** すべての特権アカウントには、セッションタイムアウトを設定することを推奨します。

**show tacacs summary** コマンドと **show tacacs session id** コマンドのアイドルタイムアウトおよびセッションタイムアウトのフィールドを使用すると、管理者は特定のアカウントの自動ログアウトを設定できます。

**セッションタイムアウト**：セキュリティ管理者は、セッションが自動的に切断される前に、ユーザーがセッションにログオンできる最大時間を分単位で指定できます。

**アイドルタイムアウト**：セキュリティ管理者は、セッションが自動的に切断される前に、セッションがアイドル状態を維持できる最大時間を分単位で指定できます。



**重要** セッションタイムアウトとアイドルタイムアウトのフィールドは排他的ではありません。両方が指定されている場合は、低いセッションタイムアウトが常に最初に到達するため、アイドルタイムアウトは常にセッションタイムアウトよりも低くする必要があります。

対話型 CLI セッションを使用できる最大時間を分単位で設定する方法の詳細については、『*CLI Reference*』の **dle-sessions threshold** コマンドと **clear tacacs sessions** CLI コマンド、および『*Statistics and Counter Reference*』の **show tacacs summary** と **show tacacs session id** を参照してください。

## リモートアクセス用のシステムの設定

リモートアクセス用にシステムを設定します。管理ユーザーは、管理ネットワークを介してリモートの場所からインスタンスにアクセスできます。

- Telnet
- セキュア シェル (SSH)
- File Transfer Protocol (FTP) (セキュアまたは非セキュア)
- Trivial File Transfer Protocol (TFTP)



(注) 2つの同時 **telnet** セッションがあり、1人の管理者が他の管理者がログに記録するコンテキストを削除した場合は、削除されたコンテキストの管理者が自動的にローカルコンテキストに退出させられることはありません。削除されたコンテキストはCLIプロンプトに引き続き表示されますが、コンテキスト固有のコマンドによってエラーが生成されます。



(注) セキュリティを最大限にするには、**SSH v2** を使用します。



(注) **FTP** および **telnet** はサポートされていません。

**ステップ1** 次のコマンドを入力してコンテキスト コンフィギュレーション モードを開始します。

```
[local]cf_host_name(config) context local
[local]cf_host_name(config-ctx)
```

**ステップ2** 必要に応じて、Telnet アクセスを許可するようにシステムを設定します。

```
[local]cf_host_name(config-ctx) server telnetd
```

**ステップ3** 必要に応じて、SSH アクセスを許可するようにシステムを設定します。

```
[local]cf_host_name(config-ctx) ssh generate key [ type v2-rsa ]
```

(注) **v2-rsa**は推奨されるキータイプです。

(注) **v2-dsa** キーワードは コンテキスト コンフィギュレーション モードの **ssh generate** CLI コマンド内に隠されています。以前のリリースでサポートされていたキーワードが後続のリリースでは隠されている可能性があります。システムは、以前のリリースで作成された既存のスクリプトや設定ファイル内の隠されたキーワードを引き続き解析します。ただし、新しいスクリプトやコンフィギュレーション ファイルで使用するために、コマンドシンタックスに隠されたキーワードは表示されなくなりました。疑問符 (?) を入力しても、ヘルプテキストの一部として隠しキーワードは表示されません。削除されたキーワードを指定すると、解析時にエラーメッセージが生成されます。

```
[local]cf_host_name(config-ctx) server sshd
[local]cf_host_name(config-sshd) subsystem sftp
[local]cf_host_name(config-sshd) exit
```

**ステップ4** 必要に応じて、次のコマンドを入力して、FTP アクセスを許可するようにシステムを設定します。

```
[local]cf_host_name(config-ctx) server ftpd
```

**ステップ5** 次のコマンドを入力して、コンフィギュレーション モードを終了します。

```
[local]cf_host_name(config-ctx) end
[local]cf_host_name
```

**ステップ6** 次のコマンドを入力して、設定を確認します。

```
[local]cf_host_name show configuration
```

CLI 出力は、次の出力例のようになります。

```
context local
  interface interface_name
    ip address ipaddress subnetmask
    exit
  subscriber default
    exit
  administrator admin_name password admin_password
  server telnetd
  server ftpd
  ssh generate key
  server sshd
  subsystem sftp
  exit
port ethernet 1/1
  bind interface interface_name local
  exit
port ethernet 1/1
  no shutdown
  exit
snmp engine-id local 800007e580ed826c191ded2d3d
end
```

**ステップ7** 次のコマンドを入力して、IP ルートの設定を確認します。

```
[local]cf_host_name show ip route
```

CLI 出力は、次の出力例のようになります。

```
""" indicates the Best or Used route.
  Destination      Nexthop          Protocol  Prec Cost Interface
*0.0.0.0/0         ipaddress        static    1    0   vnic1
*network           0.0.0.0          connected 0    0   vnic1
```

**ステップ8** 次のコマンドを入力して、インターフェイス バインディングを確認します。

```
[local]cf_host_name show ip interface name interface_name
```

*interface\_name* は、手順 7b で設定したインターフェイスの名前です。CLI 出力は、次の出力例のようになります。

```
Intf Name:          vnic1

Description:
IP State:           UP (Bound to 1/1 untagged, ifIndex 83951617)
IP Address:         ipaddress          Subnet Mask:      subnetmask
Bcast Address:     bcastaddress        MTU:              1500
Resoln Type:       ARP              ARP timeout:      3600 secsL3 monitor LC-port
switchover:        DiasabledNumber of Secondary Addresses: 0
```

**ステップ9** 「設定の確認と保存」の章の説明に従って、設定を保存します。

# SSH オプションの設定

## SSH ホストキー

SSH キーベースの認証では、誰に対しても表示が許可されている「公開」キーと、所有者のみが表示を許可されている別の「秘密」キーの、2つのキーを使用します。キーペアを作成し、ログインするデバイスに秘密キーを安全に保存して、ログインするシステム（VPC-DI）に公開キーを保存します。

SSH ホストキーは、指定された StarOS コンテキスト内で生成されます。コンテキストは、ユーザーインターフェイスに関連付けられています。

コンテキストに関連付けられている sshd サーバーにアクセスするための承認されたキーを持つ管理ユーザー名を設定または削除します。

## SSH キーのサイズ設定

グローバル コンフィギュレーション モードの **ssh key-size** CLI コマンドは、すべてのコンテキストの SSH キー生成のキーサイズを設定します（RSA ホストキーのみ）。

**ステップ 1** グローバル コンフィギュレーション モードを開始します。

```
[local]host_name# configure
[local]host_name(config)#
```

**ステップ 2** SSH キーのビットサイズを指定します。

```
[local]host_name(config)# ssh key-size { 2048 | 3072 | 4096 | 5120 | 6144 | 7168 |
9216 }
```

SSH キーのデフォルトのビットサイズは 2048 ビットです。

## SSH キー生成の待機時間の設定

SSH キーは、最後のキー生成以降に設定可能な時間間隔が経過した後にのみ生成できます。**ssh key-gen wait-time** コマンドは、この待機時間を秒単位で指定します。デフォルトの間隔は 300 秒（5 分）です。

**ステップ 1** コンテキスト コンフィギュレーション モードを開始します。

```
[local]host_name(config)# context context_name
[local]host_name(config-ctx)#
```

**ステップ 2** 待機時間間隔を指定します。

```
[local]host_name(config-ctx)# ssh key-gen wait-time seconds
```



```
[local]host_name(config-ctx)#
```

注：

- *seconds* を 0 ~ 86400 の整数で指定します。デフォルト = 300

---

## SSH 暗号化暗号方式の指定

SSH コンフィギュレーションモードの暗号 CLI コマンドは、SSH 対称暗号化のために、`sshd` の暗号優先順位リストを設定します。そのコンテキストの暗号オプションが変更されます。

**ステップ 1** SSH コンフィギュレーションモードを開始します。

```
[local]host_name(config-ctx)# server sshd
```

**ステップ 2** 必要な暗号化アルゴリズムを指定します。

```
[local]host_name(config-sshd)# ciphers algorithms
```

注：

- アルゴリズムは 1 ~ 511 文字の英数字の文字列で、次に示すように、優先順位（左から右）でカンマ区切りの変数（スペースなし）の単一の文字列として使用するアルゴリズムを指定します。
  - **blowfish-cbc** : 対称キープロック暗号、暗号ブロック連鎖（CBC）
  - **3des-cbc** : トリプルデータ暗号化規格、CBC
  - **aes128-cbc** : Advanced Encryption Standard（AES; 高度暗号化規格）、128 ビットキーサイズ、CBC
  - **aes128-ctr** : AES、128 ビットキーサイズ、カウンタモード暗号化（CTR）
  - **aes192-ctr** : AES、192 ビットキーサイズ、CTR
  - **aes256-ctr** : AES、256 ビットキーサイズ、CTR
  - **aes128-gcm@openssh.com** : AES、128 ビットキーサイズ、Galois Counter モード [GCM]、OpenSSH
  - **aes256-gcm@openssh.com** : AES、256 ビットキーサイズ、GCM、OpenSSH
  - **chacha20-poly1305@openssh.com** : ChaCha20 対称暗号、Poly1305 暗号化メッセージ認証コード [MAC]、OpenSSH

通常のビルドにおけるアルゴリズムのデフォルトの文字列は次のとおりです。

```
blowfish-cbc,3des-cbc,aes128-cbc,aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-gcm@openssh.com,
chacha20-poly1305@openssh.com
```

信頼できるビルドにおけるアルゴリズムのデフォルトの文字列は次のとおりです。

```
aes256-ctr,aes192-ctr,aes128-ctr
```

**ステップ 3** SSH コンフィギュレーションモードを終了します。

```
[local]host_name(config-sshd)# end
[local]host_name#
```

## MAC アルゴリズムの設定

### 機能の概要と変更履歴

#### 要約データ

該当製品または機能エリア	すべて
該当プラットフォーム	<ul style="list-style-type: none"> <li>• ASR 5500</li> <li>• VPC-DI</li> <li>• VPC-SI</li> </ul>
機能のデフォルト	無効：設定が必要
このリリースでの関連する変更点	N/A
関連資料	<ul style="list-style-type: none"> <li>• <i>ASR 5500 System Administration Guide</i></li> <li>• <i>Command Line Interface Reference</i></li> <li>• VPC-DI システム管理ガイド</li> <li>• VPC-SI System アドミニストレーションガイド</li> </ul>

#### マニュアルの変更履歴



**重要** リリース 21.2 および N5.1 よりも前に導入された機能の改訂履歴の詳細は示していません。

改訂の詳細	リリース
最初の導入。	21.13

#### 機能説明

MAC アルゴリズム設定機能を使用すると、内部 SSHD サーバーの MAC アルゴリズムの優先順位を設定または変更することができます。

この機能をサポートする、新しい CLI **MACs** CLI コマンドが SSH モード設定に導入されました。

#### MAC アルゴリズムの設定

ここでは、MAC アルゴリズムの設定方法を説明します。

MAC アルゴリズムの優先順位を指定するには、次の設定を使用します。

```

configure
  context context_name
    server sshd
      macs algorithms
    end

```

```

default macs

```

注：

- *algorithms* : 1 ~ 511 文字の英数字文字列を参照します。この文字列は、次のリストで示す優先順位（左から右）のコンマ区切りの変数（スペースなし）の1つの文字列として使用するアルゴリズムを指定します。

- HMAC = ハッシュベースのメッセージ認証コード
- SHA2 = セキュア ハッシュ アルゴリズム 2
- SHA1 = セキュア ハッシュ アルゴリズム 1
- ETM = Encrypt-Then-MAC
- UMAC = ユニバーサルハッシュに基づくメッセージ認証コード

- 次に、通常のビルドのヘルプ文字列とアルゴリズムのリストを示します。

```

hmac-sha2-512-etm@openssh.com, hmac-sha2-256-etm@openssh.com, hmac-sha1-etm@openssh.com, hmac-sha2-512,
hmac-sha2-256, hmac-sha1, umac-128-etm@openssh.com, umac-128@openssh.com, umac-64-etm@openssh.com, umac-64@openssh.com

```

- 次に、信頼できるビルドのヘルプ文字列とアルゴリズムのリストを示します。

```

hmac-sha2-512-etm@openssh.com, hmac-sha2-256-etm@openssh.com, hmac-sha1-etm@openssh.com, hmac-sha2-512,
hmac-sha2-256, hmac-sha1

```

- デフォルト値の文字列は次のとおりです。

```

hmac-sha2-512-etm@openssh.com, hmac-sha2-256-etm@openssh.com, hmac-sha1-etm@openssh.com, hmac-sha2-512,
hmac-sha2-256, hmac-sha1

```

## MAC アルゴリズムの指定

MAC アルゴリズムの優先順位を設定するには、次の CLI コマンドを使用します。このコマンドは、SSH コンフィギュレーションモードで設定します。

```

configure
  context context_name
    server sshd
      macs algorithms
    end

```

```

default macs

```

注：

- *algorithms* : 1 ~ 511 文字の英数字文字列を参照します。この文字列は、次のリストで示す優先順位（左から右）のコンマ区切りの変数（スペースなし）の1つの文字列として使用するアルゴリズムを指定します。

- HMAC = ハッシュベースのメッセージ認証コード
  - SHA2 = セキュア ハッシュ アルゴリズム 2
  - SHA1 = セキュア ハッシュ アルゴリズム 1
  - ETM = Encrypt-Then-MAC
  - UMAC = ユニバーサルハッシュに基づくメッセージ認証コード
- 次に、通常のビルドのヘルプ文字列とアルゴリズムのリストを示します。
- ```
hmac-sha2-512-etm@openssh.com, hmac-sha2-256-etm@openssh.com, hmac-sha1-etm@openssh.com, hmac-sha2-512,
hmac-sha2-256, hmac-sha1, urac-128-etm@openssh.com, urac-128@openssh.com, urac-64-etm@openssh.com, urac-64@openssh.com
```
- 次に、信頼できるビルドのヘルプ文字列とアルゴリズムのリストを示します。
- ```
hmac-sha2-512-etm@openssh.com, hmac-sha2-256-etm@openssh.com, hmac-sha1-etm@openssh.com, hmac-sha2-512,
hmac-sha2-256, hmac-sha1
```
- デフォルト値の文字列は次のとおりです。
- ```
hmac-sha2-512-etm@openssh.com, hmac-sha2-256-etm@openssh.com, hmac-sha1-etm@openssh.com, hmac-sha2-512,
hmac-sha2-256, hmac-sha1
```

## SSH キーの生成

**ssh generate** コマンドは、SSH サーバーによって使用される公開キーまたは秘密キーのペアを生成します。**ssh generate** CLI コマンド内に隠されている **v2-dsa** キーワード。SSH キーを生成するために使用できる唯一のキーワードは、**v2-rsa** です。



**重要** 生成されたキーペアは、コマンドが再度発行されるまで使用中のままになります。

**ステップ 1** コンテキスト コンフィギュレーション モードを開始します。

```
[local]host_name(config)# context context_name
[local]host_name(config-ctx)#
```

**ステップ 2** SSH キーペアを生成します。

```
[local]host_name(config-ctx)# ssh generate key type v2-rsa
[local]host_name(config-ctx)#
```

## SSH キーペアの設定

**ssh key** コマンドは、システムで使用される公開キーと秘密キーのペアを設定します。**v2-dsa** キーワードは、**ssh key** コマンドでは隠されています。

SSH キーペアのパラメータを指定します。

```
[local]host_name(config-ctx)# ssh key data length octets type v2-rsa
```

注：

- **data** は 1 ～ 1023 文字の英数字の文字列で表される暗号化キーです。
- **length octets** は 0 ～ 65535 の整数で表される、暗号化されたキーのオクテット単位の長さです。
- **type** はキータイプを指定します。**v2-rsa** はサポートされている唯一のタイプです。

**重要** 最大 200 の設定可能な承認済みの SSH キーが StarOS でサポートされています。

## 承認済み SSH ユーザーアクセス

ユーザーが、SSH 認証キーペアを持つ特定のホストから StarOS コンテキストにアクセスすることを許可する必要があります。

### SSH ユーザーアクセスの認可

SSH コンフィギュレーションモードの **authorized-key** コマンドは、指定されたホストからのコンテキストへのユーザーアクセスを許可します。

**ステップ 1** SSH コンフィギュレーションモードに移動します。

```
[local]host_name(config-ctx)# server sshd
[local]host_name(config-sshd)#
```

**ステップ 2** **authorized-key** コマンドを使用して管理ユーザーアクセスを指定します。

```
[local]host_name(config-sshd)# authorized-key username user_name host host_ip [ type {
v2-dsa | v2-rsa } ]
```

注：

- **username user\_name** は、sshd サーバーへのアクセスに許可されたキーを持つ既存の StarOS 管理者ユーザー名を指定します。**user\_name** は、1 ～ 255 文字の英数字文字列で表されます。sshd キーをバイパスしないようにするには、**nopassword** オプションを使用してコンテキストコンフィギュレーションモードの **administrator** コマンドを使用して、ユーザー名を事前に作成しておく必要があります。管理者の作成の詳細については、「システム設定」の章を参照してください。
- **host host\_ip** は、このユーザー名の認証キーを持つ SSH ホストの IP アドレスを指定します。この IP アドレスは、IPv4 ドット付き 10 進表記または IPv6 コロン区切り 16 進表記である必要があります。
- **type** はキータイプを指定します。**v2-rsa** はサポートされている唯一のタイプです。

## SSH ユーザーログインの制限事項

管理者は、StarOS CLI への SSH アクセスを、許可されたユーザーの「ホワイトリスト」に制限できます。サービスへのアクセスは、正当なニーズを持つユーザーにのみ制限される場合があります。明示的に許可されたユーザーのみが、SSH を介してホストに接続できます。ユーザー名には、必要に応じて特定の送信元 IP アドレスを含めることができます。

AllowUsers リストは、スペースで区切られたユーザー名パターンで構成されます。パターンで「USER」という形式を使用すると、そのユーザーに対してログインが制限されます。パターンが「USER@IP\_ADDRESS」形式の場合、ユーザーと IP アドレスは個別にチェックされ、指定した IP アドレスからのユーザーへのログインを制限します。

デフォルトでは、任意のユーザーによる無制限のアクセスを許可します。

### 許可済みユーザーリストの作成

**allowusers add** コマンドを使用すると、管理者は StarOS CLI にログインできるユーザーのリストを作成できます。

**ステップ 1** コンテキスト コンフィギュレーション モードを開始します。

```
[local]host_name(config)# context context_name
[local]host_name(config-ctx)#
```

**ステップ 2** SSH コンフィギュレーション モードに移動します。

```
[local]host_name(config-ctx)# server sshd
```

**ステップ 3** SSH ユーザーリストを設定します。

```
[local]host_name(config-sshd)# allowusers add user_list
```

*user\_list* は、スペースで区切られたユーザー名のパターンのリストを、1 ~ 999 文字の英数字の文字列として指定します。パターンで「USER」という形式を使用すると、そのユーザーに対してログインが制限されます。

パターンが「USER@IP\_ADDRESS」形式の場合は、ユーザー名と IP アドレスが個別にチェックされ、その特定の IP アドレスからユーザーへのログインが制限されます。

パターンが「USER@<context>@IP\_ADDRESS」形式の場合は、ユーザー名、StarOS コンテキスト、および IP アドレスが個別にチェックされ、その特定の IP アドレスから特定のコンテキストに関連付けられているユーザーへのログインを制限します。

*user\_list* には次の制限が適用されます。

- この文字列の最大長は 3000 バイト（スペースを含む）です。
- スペースでカウントされる AllowUsers の最大数は 256 で、これは OpenSSH からの制限と一致します。

**重要** 上記の制限のいずれかを超えると、エラーメッセージが表示されます。このメッセージでは、正規表現のパターンを使用して文字列を短くするか、または **no allowusers add** や **default allowusers add** を使用してすべての allowusers を削除するか、または再設定するように求められます。

詳細については、『*Command Line Interface Reference*』の「*SSH Configuration Mode Commands*」の章を参照してください。

**ステップ 4** SSH コンフィギュレーション モードを終了します。

```
[local]host_name(config-sshd)# end
[local]host_name#
```

## SSH ユーザーログイン認証

StarOS は、次のシナリオの場合、許可済みキーとユーザーアカウントの組み合わせを使用して SSH によるユーザーログインの試行を認証します。

- ユーザーは、ローカルコンテキスト (VPN) インターフェイスを介してローカルコンテキストのユーザー名と、ローカルコンテキストで設定されている許可済みのキーを使用してログインしようとします。
- ユーザーは、ローカル以外のコンテキストインターフェイスを介してローカル以外のコンテキストのユーザー名と、ローカル以外のコンテキストで設定されている許可済みのキーを使用してログインしようとします。
- ユーザーは、ローカル以外のコンテキストインターフェイスを介してローカルコンテキストのユーザー名と、ローカルコンテキストで設定されている許可済みのキーを使用してログインしようとします。
- ユーザーは、ローカル コンテキスト インターフェイスを介してローカル以外のコンテキストのユーザー名と、ローカル以外のコンテキストで設定されている許可済みのキーを使用してログインしようとします。

現在のシステム設定に基づいて認証が失敗すると、ログインが阻止され、エラーメッセージが生成されます。

StarOS では、ユーザー ID が異なるユーザーが同じ公開 SSH キーを使用して、許可されていないコンテキストへログインすることは許可されていません。ユーザーの認証では、許可済みキーとユーザーアカウントの組み合わせが考慮されます。



**重要** StarOS リリース 21.0 以降では、ユーザーがローカル以外のコンテキストからログインした場合、そのユーザーは `/flash` ディレクトリにアクセスできません。

## セキュアなセッションログアウト

StarOS が SSH クライアントから切断されると、デフォルトの動作によって CLI または SFTP セッションは約 45 秒 (デフォルトのパラメータを使用) で終了します。SSH コンフィギュレー

セッションモードの CLI コマンドを使用すると、このデフォルトの SSHD 切断動作を無効にしたり、変更したりできます。



**重要** セキュリティを強化するため、シスコでは、少なくとも `lient-alive-countmax` を 2、`client-alive-interval` を 5 にすることを推奨します。セッションのログアウト値が小さいと、ssh セッションのログアウトが不定期にログアウトする可能性があります。セキュリティとユーザーの使いやすさとのバランスが取れるように値を調整します。

**client-active-countmax** コマンドは、sshd なしで送信される `client-alive` メッセージの数を、SSH クライアントからのメッセージを受信しないように設定します（デフォルトは 3）。`client-alive` メッセージの送信中にこのしきい値に達すると、sshd は SSH クライアントを切断してセッションを終了します。

**client-alive-interval** コマンドは、タイムアウト間隔を秒単位で設定します（デフォルトは 15）。その後、SSH クライアントからデータを受信しなかった場合、sshd は暗号化されたチャネルを介してメッセージを送信し、クライアントからの応答を要求します。メッセージが送信される回数は、`client-alive-countmax` パラメータによって決定されます。sshd が SSH クライアントの切断を解除するまでのおおよその時間は、`client-alive-countmax X client-alive-interval` となります。

クライアントまたはサーバーがいつ接続が非アクティブになったかを認識しているかどうかにかかわらず、依存している場合、`client-alive` メカニズムは重要です。



**重要** `client-alive` メッセージは暗号化チャネルを介して送信されるため、スプーフィングできません。



**重要** これらのパラメータは、SSH プロトコルバージョン 2 のみに適用されます。

## デフォルトの sshd セキュア セッション ログアウト パラメータの変更

次のコマンドシーケンスは、クライアントの `ClientAliveCountmax`（デフォルトは 3）および `ClientAliveInterval`（デフォルトは 15 秒）のパラメータのデフォルト設定を変更します。

**ステップ 1** コンテキスト コンフィギュレーション モードを開始します。

```
[local]host_name# configure
```

**ステップ 2** SSH コンフィギュレーション モードに移動します。

```
[local]host_name(config)# context context_name
```

**ステップ 3** `ClientAliveCountmax` パラメータを 2 に設定します。

```
[local]host_name(config-sshd)# client-alive-countmax 2
```



ステップ4 ClientAliveInterval パラメータを 5 秒に設定します。

```
[local]host_name(config-sshd)# client-alive-interval 5
```

ステップ5 SSH コンフィギュレーションモードを終了します。

```
[local]host_name(config-sshd)# end  
[local]host_name#
```

---

## 外部サーバーへの SSH クライアントログイン

StarOS は、StarOS ゲートウェイから外部サーバーへの SSH/SFTP アクセスの公開キーの認証をサポートしています。この機能を設定するには、SSH クライアントキーのペアを生成し、クライアント公開キーを外部サーバーにプッシュします。



(注) デフォルトでは、StarOS は外部サーバーへの `username-password` の認証のみをサポートしています。

---

## SSH クライアント暗号の設定

SSH クライアント コンフィギュレーションモードの `cipher` CLI コマンドは、外部サーバーにログインするときに暗号優先順位リストを設定します。

ステップ1 SSH クライアント コンフィギュレーションモードを開始します。

```
[local]host_name(config)# client ssh
```

ステップ2 必要な暗号化アルゴリズムを指定します。

```
[local]host_name(config-ssh)# ciphers algorithms
```

注：

- アルゴリズムは 1 ~ 511 文字の英数字の文字列で、次に示すように、優先順位（左から右）でカンマ区切りの変数（スペースなし）の単一の文字列として使用するアルゴリズムを指定します。
  - **blowfish-cbc** : 対称キープブロック暗号、暗号ブロック連鎖（CBC）
  - **3des-cbc** : トリプルデータ暗号化規格、CBC
  - **aes128-cbc** : Advanced Encryption Standard（AES; 高度暗号化規格）、128 ビットキーサイズ、CBC
  - **aes128-ctr** : AES、128 ビットキーサイズ、カウンタモード暗号化（CTR）
  - **aes192-ctr** : AES、192 ビットキーサイズ、CTR
  - **aes256-ctr** : AES、256 ビットキーサイズ、CTR
  - **aes128-gcm@openssh.com** : AES、128 ビットキーサイズ、Galois Counter モード [GCM]、OpenSSH
  - **aes256-gcm@openssh.com** : AES、256 ビットキーサイズ、GCM、OpenSSH

- **chacha20-poly1305@openssh.com** : ChaCha20 対称暗号、Poly1305 暗号化メッセージ認証コード [MAC]、OpenSSH

通常のビルドにおけるアルゴリズムのデフォルトの文字列は次のとおりです。

```
aes256-ctr,aes192-ctr,aes128-ctr,aes256-gcm@openssh.com,aes128-gcm@openssh.com,chacha20-poly1305@openssh.com,blowfish-cbc,3des-cbc,aes128-cbc
```

信頼できるビルドにおけるアルゴリズムのデフォルトの文字列は次のとおりです。

```
aes256-ctr,aes192-ctr,aes128-ctr
```

**ステップ3** SSH クライアント コンフィギュレーション モードを終了します。

```
[local]host_name(config-ssh)# end
[local]host_name#
```

## 優先認証方式の設定

SSH クライアント コンフィギュレーション モードの **preferredauthentications** CLI コマンドは、適切な認証方式を設定します。

**ステップ1** SSH クライアント コンフィギュレーション モードを開始します。

```
[local]host_name(config)# client ssh
```

**ステップ2** 優先認証方式の指定

```
[local]host_name(config-ssh)# preferredauthentications methods
```

注：

- **方式** : 次に示すように、優先順位順 (左から右) に、カンマ区切りの変数 (スペースなし) の単一の文字列として使用される認証方式を指定します。
  - **publickey** : SSH v2-RSA プロトコルを使用した認証
  - **keyboard-interactive** : 任意の数の情報を要求します。各情報について、サーバーはプロンプトのラベルを送信します。
  - **password** : 単一のパスワードの単純な要求
- **デフォルト** : 方式の値を [publickey,password] にリセットします。

**ステップ3** SSH クライアント コンフィギュレーション モードを終了します。

```
[local]host_name(config-ssh)# exit
[local]host_name(config)#
```

## SSH クライアントキーペアの生成

SSH クライアント コンフィギュレーション モードでコマンドを使用し、秘密キーを指定して、SSH クライアントキーペアを生成します。

**ステップ 1** SSH クライアント コンフィギュレーション モードを開始します。

```
[local]host_name(config)# client ssh
[local]host_name(config-ssh)#
```

**ステップ 2** SSH クライアントキーのペアを生成します。

```
[local]host_name(config-ssh)# ssh generate key [ type v2-rsa ] [ key-size ]
[local]host_name(config-ssh)#
```

**type v2-rsa** は SSH クライアントキーのタイプを指定します。サポートされている SSH クライアントキーのタイプは、**v2-rsa** のみです。

**key-size** は SSH クライアントのキーサイズを指定します。サポートされているキーサイズは、2048、3072、4096、5120、6144、7168、および 9216 です。

**ステップ 3** SSH クライアントキーが生成されていることを確認します。

```
[local]host_name(config-ssh)# do show ssh client key
```

**ステップ 4** SSH クライアント コンフィギュレーション モードを終了します。

```
[local]host_name(config-ssh)# exit
[local]host_name(config)#
```

## 外部サーバーへの SSH クライアント公開キーのプッシュ

このサーバーへの SSH/SFTP アクセスをサポートするには、SSH クライアント公開キーを外部サーバーにプッシュする必要があります。

**ステップ 1** Exec モードで、**push ssh-key** コマンドを実行します。

```
[local]host_name# push ssh-key { host_name | host_ip_address } user username [ context context_name ]
[local]host_name#
```

**host\_name** は、DNS ルックアップを介して解決される必要がある論理ホスト名を使用してリモートサーバーを指定します。これは、1 ~ 127 文字の英数字文字列で表されます。

**host\_ip\_address** は、IPv4 ドット付き 10 進表記または IPv6 コロン区切り 16 進表記で表されます。

**user username** は、外部サーバーで有効なユーザー名を 1 ~ 79 文字の英数字の文字列として指定します。

**context context\_name** は、有効なコンテキスト名を指定します。コンテキスト名はオプションです。指定されていない場合は、現在のコンテキストが処理に使用されます。

ステップ2 他の外部サーバーでの SSH/SFTP アクセスをサポートするには、ステップ1 を繰り返します。

ステップ3 外部サーバーへの SSH クライアントのログインをテストします。

```
local]host_name# ssh { hostname | ip_address } user username port port_number
```

## NETCONF の有効化

SSH キーは、NETCONF プロトコルと ConfD エンジンが Cisco Network Service Orchestrator (NSO) をサポートするために有効になる前に必要になります。

NETCONF を有効にする方法の詳細については、このガイドの付録の「NETCONF と ConfD」を参照してください。

## 2 番目の IP アドレスを使用した管理インターフェイスの設定

必要に応じて、vNIC 管理インターフェイスに 2 番目の IP アドレスを設定できます。

### 手順

|       | コマンドまたはアクション                              | 目的                                                                                                                                                                                                                              |
|-------|-------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ステップ1 | プロンプトで次のコマンドを入力して、コンフィギュレーションモードを開始します。   | [local]host_name <b>configure</b><br>[local]host_name(config)                                                                                                                                                                   |
| ステップ2 | コンテキスト コンフィギュレーションモードを開始するには、次のように入力します。  | [local]host_name(config) <b>context local</b><br>[local]host-name(config-ctx)                                                                                                                                                   |
| ステップ3 | 次のコマンドを使用して、インターフェイスのシロット番号とポート番号を入力します。  | [local]host_name(config-ctx) 1/1<br>[local]host_name(config-if-eth)                                                                                                                                                             |
| ステップ4 | 次のコマンドを入力して、セカンダリ IP アドレスとサブネットマスクを入力します。 | [local]host_name(config-if-eth) { <b>ip   ipv</b> }<br><b>address</b> ipaddress subnet_mask secondary                                                                                                                           |
| ステップ5 | 次のコマンドを入力して、コンフィギュレーションモードを終了します。         | [local]host_name(config-if-eth) <b>end</b>                                                                                                                                                                                      |
| ステップ6 | 次のコマンドを入力して、インターフェイスの IP アドレスを確認します。      | [local]host_name <b>show config context local</b><br><br>CLI 出力は次の例のようになります。<br><br>config<br>context local<br>interface interface_name<br>ip address ipaddress subnetmask<br>ip address ipaddress subnetmask secondary<br>exit |

|        | コマンドまたはアクション                  | 目的 |
|--------|-------------------------------|----|
| ステップ 7 | 「インターフェイスとポートの設定の確認と保存」に進みます。 |    |

## Open SSH から Cisco SSH へのアップグレードと移行

### 機能の概要と変更履歴

#### 要約データ

|                 |                                                                                                                                                                                                                        |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 該当製品または機能エリア    | すべて                                                                                                                                                                                                                    |
| 該当プラットフォーム      | <ul style="list-style-type: none"> <li>• ASR 5500</li> <li>• VPC-DI</li> <li>• VPC-SI</li> </ul>                                                                                                                       |
| 機能のデフォルト        | 有効、常時オン                                                                                                                                                                                                                |
| このリリースでの関連する変更点 | N/A                                                                                                                                                                                                                    |
| 関連資料            | <ul style="list-style-type: none"> <li>• <i>ASR 5500 System Administration Guide</i></li> <li>• <i>Command Line Interface Reference</i></li> <li>• VPC-DI システム管理ガイド</li> <li>• VPC-SI System アドミニストレーションガイド</li> </ul> |

#### マニュアルの変更履歴



**重要** リリース 21.2 および N5.1 よりも前に導入された機能の改訂履歴の詳細は示していません。

| 改訂の詳細                                                                     | リリース      |
|---------------------------------------------------------------------------|-----------|
| このリリースでは、暗号と MAC のアルゴリズム値は、OpenSSH から CiscoSSH へのアップグレードと移行に基づいて変更されています。 | 21.16     |
| 最初の導入。                                                                    | 21.2 よりも前 |

## 変更された機能

Cisco ASR 5500 および VPC 製品のセキュリティ対策として、暗号および MAC アルゴリズム値は、Cisco SSH バージョンへの Open SSH のアップグレードと移行をサポートするように変更されています。

以前の動作：21.16 よりも前のリリースでは、**cipher** コマンドと **macs** コマンドの **default** アルゴリズム値は次のようになっていました。

- 暗号化方式

### 21.15（通常ビルドのみ）

通常のビルドのアルゴリズムの値を次のようにリセットします。

```
blowfish-cbc,3des-cbc,aes128-cbc,aes128-ctr,aes192-ctr,aes256-ctr,aes128-gcm@openssh.com,aes256-gcm@openssh.com,chacha20-poly1305@openssh.com
```

- MAC

### 21.15（信頼できるビルドのみ）

信頼できるビルドのアルゴリズムの値を次のようにリセットします。

```
hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sha1-etm@openssh.com,hmac-sha2-512,hmac-sha2-256,hmac-sha1
```

- KEX アルゴリズム

### 21.15

通常のビルドと信頼できるビルドで使用可能なアルゴリズム：

```
diffie-hellman-group1-sha1,diffie-hellman-group14-sha1
```

新しい動作：このリリースでは、**default** コマンドと **cipher** コマンドの **macs** アルゴリズム値は次のとおりです。

- 暗号化方式

リリース 21.16 以降：Post OpenSSH から CiscoSSH へのアップグレードと移行

通常のビルドのデフォルトのアルゴリズムは次のとおりです。

```
aes256-ctr,aes192-ctr,aes128-ctr,aes256-gcm@openssh.com,aes128-gcm@openssh.com,chacha20-poly1305@openssh.com
```

通常のビルドで使用可能なアルゴリズムは次のとおりです。

```
aes256-ctr,aes192-ctr,aes128-ctr,aes256-gcm@openssh.com,aes128-gcm@openssh.com,chacha20-poly1305@openssh.com,aes128-cbc
```

信頼できるビルドでデフォルトのアルゴリズムと使用可能なアルゴリズム：

```
aes256-ctr,aes192-ctr,aes128-ctr
```



(注) 信頼できるビルドのデフォルトの暗号と設定可能な暗号に変更はありません。

- MAC

リリース 21.16 以降：Post OpenSSH から CiscoSSH へのアップグレードと移行

通常のビルドでデフォルトのアルゴリズムと使用可能なアルゴリズム：

```
hmac-sha2-512-etm@openssh.com, hmac-sha2-256-etm@openssh.com, hmac-sha1-etm@openssh.com, hmac-sha2-512,
hmac-sha2-256, hmac-sha1
```

信頼できるビルドでデフォルトのアルゴリズム：

```
hmac-sha2-512, hmac-sha2-256, hmac-sha1
```

信頼できるビルドで使用可能なアルゴリズム：

```
hmac-sha2-512, hmac-sha2-256, hmac-sha1
```



(注) hmac-sha2-512-etm@openssh.com、hmac-sha2-256-etm@openssh.com、hmac-sha1-etm@openssh.com は信頼できるビルドから削除されます。

#### • KEX アルゴリズム

リリース 21.16 以降：Post OpenSSH から CiscoSSH へのアップグレードと移行

通常のビルドと信頼できるビルドで使用可能なアルゴリズム：

```
diffie-hellman-group14-sha1
```



(注) KEXアルゴリズムは、StarOS では設定できません。したがって、CLI の変更はありません。

## VM ハードウェアの検証

リソース割り当ての問題を回避するには、システム内で使用されるすべての VM が同じサイズの CPU と同じサイズのメモリを持つことが重要です。すべてのインターフェイスでパフォーマンスのバランスを取るために、サービスポートと DI ポートが同じスループット能力を備えていることを確認してください。

すべてのカードまたは特定のカードのハードウェア設定を確認するには、**show cloud hardware**`[card_number]` コマンドを使用します。次に、カード 1 (CF) でのこのコマンドの出力例を示します。

```
[local]s1# show cloud hardware 1

Card 1:
CPU Nodes           : 1
CPU Cores/Threads   : 8
Memory              : 16384M (qvpc-di-medium)
Hugepage size       : 2048kB
cpeth0              :
  Driver             : virtio_net
loeth0              :
```

```
Driver : virtio_net
```

次に、カード3 (SF) でのこのコマンドの出力例を示します。

```
[local]s1# show cloud hardware 1

Card 3:
CPU Nodes           : 1
CPU Cores/Threads   : 8
Memory              : 16384M (qvmc-di-medium)
Hugepage size       : 2048kB
cpeth0              :
  Driver            : vmxnet3
port3_10            :
  Driver            : vmxnet3
port3_11            :
  Driver            : vmxnet3
```

基本となる VM ハードウェアの最適な設定を表示するには、**show hardware optimum** を使用します。現在の VM 設定を最適な設定と比較するには、**show cloud hardware test** コマンドを使用します。最適に設定されていないパラメータは、次の出力例に示すように、アスタリスク付きでフラグが立てられます。この例では、CPU コア/スレッドおよびメモリが最適に設定されていません。

```
[local]s1# show cloud hardware test 1

Card 1:
CPU Nodes           : 1
* CPU Cores/Threads : 8                               Optimum value is 4
* Memory            : 8192M (qvmc-di-medium)         Optimum value is 16384
Hugepage size       : 2048kB
cpeth0              :
  Driver            : virtio_net
loeth0              :
  Driver            : virtio_net
```

設定ディスクまたはローカルフラッシュ上の設定ファイルを表示するには、**show cloud configuration card\_number** コマンドを使用します。フラッシュメモリ上のロケーションパラメータファイルは、インストール時に定義されます。また、ディスク構成は通常、オーケストレーションによって作成され、カードに接続されます。次に、カード1でのこのコマンドの出力例を示します。

```
[local]s1# show cloud configuration 1

Card 1:
Config Disk Params:
-----
  No config disk available

Local Params:
-----
CARDSLOT=1
CARDTYPE=0x40010100
CPUID=0
```



すべてのカードまたは特定のカードの IFTASK 設定を表示するには、**show cloud hardware iftask** コマンドを使用します。デフォルトでは、コアは PMD と VNPU の両方に使用されるように設定されています。次に、カード 4 でのこのコマンドの出力例を示します。

```
[local]mySystem# show cloud hardware iftask 4
Card 4:
  Total number of cores on VM:      24
  Number of cores for PMD only:     0
  Number of cores for VNPU only:    0
  Number of cores for PMD and VNPU: 3
  Number of cores for MCDMA:        4
  Hugepage size:                    2048 kB
  Total hugepages:                   16480256 kB
  NPUSHM hugepages:                  0 kB
  CPU flags: avx sse sse2 ssse3 sse4_1 sse4_2
  Poll CPU's: 1 2 3 4 5 6 7
  KNI reschedule interval: 5 us
```



## 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。