Detector モジュールの設定

この章では、Cisco Traffic Anomaly Detector Module（Detector モジュール）のサービスの設定方法について説明します。

この章には、次の主要な項があります。

- Detector モジュールのサービスのアクティブ化
- アクセス コントロールの設定：認証、認可、アカウンティングの使用
- Cisco Anomaly Guard Module との通信のイネーブル化
- 日付と時刻の設定
- SSH 鍵の管理
- SFTP 接続用の鍵の設定
- ホスト名の変更
- SNMP トラップのイネーブル化
- SNMP コミュニティ ストリングの設定

Detector モジュールのサービスのアクティブ化

Detector モジュールでアクティブにするサービスを定義することができます。正しい機能をイネーブルにするためには、サービスをイネーブルにして、そのサービスへのアクセスを許可する必要があります。Detector モジュールのサービスのアクティベーションを制御できます。また、特定の IP アドレスに対してアクセス権を付与または拒否することにより、Detector モジュールにアクセスし、制御する IP アドレスを制限できます。

表 4-1 で、Detector モジュールのサービスについて説明します。

表 4-1　Detector モジュールのサービス

サービス	説明
snmp-server	Simple Network Management Protocol（SNMP; 簡易ネットワーク管理プロトコル）サーバ サービス。SNMP を使用して Detector モジュールにアクセスすることにより、Riverhead 管理情報ベース（Riverhead の専用 MIB、MIB2、および UCDavis の MIB）で定義された情報を取得することができます。 MIB 定義の詳細については、このバージョンに付属の MIB ファイルを参照してください。 （注）　Riverhead MIB には、64 ビットのカウンタが含まれています。MIB を読み取るには、SNMP バージョン 2 をサポートするブラウザを使用する必要があります。
snmp-trap	SNMP トラップ サービス。snmp-trap サービスをアクティブにすると、Detector モジュールは SNMP トラップを生成します。詳細については、P.4-39 の「SNMP トラップのイネーブル化」を参照してください。
ssh	Secured Shell サービス（詳細については、P.4-35 の「SSH 鍵の管理」を参照）。
wbm	Web-Based Management（WBM）サービス。Web ブラウザを使用して、Web から Detector モジュールを制御できます。

Detector モジュールのサービスをアクティブにするには、次の手順を実行します。

ステップ 1　Detector モジュールのサービスをイネーブルにします。次のコマンドを入力します。

```
service {snmp-server | snmp-trap | wbm}
```

Detector モジュールのサービスについては、表 4-1 を参照してください。デフォルトでは、SSH 以外、Detector モジュールのすべてのサービスはディセーブルになっています。

ステップ 2　Detector モジュールのサービスに対するアクセス権を付与し、接続をイネーブルにします。次のコマンドを入力します。

```
permit service ip-address-general [ip-mask]
```

表 4-2 に、**permit** コマンドの引数を示します。

表 4-2　permit コマンドの引数

パラメータ	説明
service	アクセスと操作の対象となるサービス。Detector モジュールのサービスについては、表 4-1 を参照してください。
ip-address-general	アクセスを許可する IP アドレス。IP アドレスをドット区切り 10 進表記で入力します（たとえば 192.168.10.2）。すべての IP アドレスからのアクセスを許可するには、アスタリスク（*）を使用します。
ip-mask	（オプション）IP サブネット マスク。サブネット マスクをドット区切り 10 進表記で入力します（たとえば 255.255.255.0）。デフォルトのサブネット マスクは、255.255.255.255 です。

注意　セキュリティ上の理由から、すべての IP アドレスからのサービスへのアクセスを許可すること（*）は推奨しません。

次の例を参考にしてください。

```
user@DETECTOR-conf# service wbm
user@DETECTOR-conf# permit wbm 192.168.10.35
```

アクセス コントロールの設定：認証、認可、アカウンティングの使用

アクセス コントロールとは、Detector モジュールにアクセスできるユーザ、およびアクセス権を持ったユーザが使用できるサービスを制御することです。認証、認可、アカウンティング（AAA）のセキュリティ サービスは、アクセスを設定するための基本的なフレームワークとなります。

- 認証:ユーザにシステムおよびシステム サービスへのアクセスを許可する前に、そのユーザを識別する方法。
- 認可：システムへのアクセスを取得した後で、ユーザが実行することのできる内容を決定するプロセス。通常、このプロセスは、ユーザが認証され、システムの操作を開始した後で実行されます。
- アカウンティング：ユーザが実行中または実行済みの内容を記録する処理。アカウンティングにより、ユーザがアクセスしているサービスを追跡することができます。

Detector モジュールには、次のシステム ユーザ アカウントがあらかじめ設定されています。

- admin：admin ユーザ アカウントには、Detector モジュールの CLI およびすべての機能にアクセスできる管理アクセス権が設定されています。Detector モジュールの CLI に初めて接続すると、このアカウントに対するパスワードを設定するように要求されます。新しいユーザ アカウントを設定するには admin ユーザ アカウントを使用します。
- riverhead：riverhead ユーザ アカウントには、ダイナミック（dynamic）のアクセス権が設定されています。Detector モジュールは、このユーザ アカウントを使用して、Cisco Anomaly Guard Module と通信します。Detector モジュールの CLI に初めて接続すると、このアカウントに対するパスワードを設定するように要求されます。

  Detector モジュールは、riverhead ユーザ名を使用して、Guard をリモートでアクティブにします。

システム ユーザ アカウントは削除できません。

ユーザを定義すると、Detector モジュールのユーザ コミュニティをドメインに分割して、必要に応じてパスワードを割り当てることができるため、セキュリティで保護され、管理されたアクセスを確立できます。初期設定が完了した後は、ユーザのアクションを監視できるように新しいアカウントを作成し、システム ユーザ アカウントは使用しないことをお勧めします。

次の各項では、アクセス コントロールの設定方法について説明します。

認証の設定

ユーザが Detector モジュールにログインしようとするとき、または（**enable** コマンドを使用して）上位の特権レベルを要求するときに、Detector モジュールで使用する認証方式を設定することができます。Detector モジュールは、次の認証オプションを提供します。

- ローカル認証：ローカル認証では、ローカルに設定されたログイン名およびイネーブル パスワードが認証に使用されます。これはデフォルトの認証方式です。詳細については、P.4-8 の「ローカル認証の設定」を参照してください。
- TACACS+ 認証 ：TACACS+ 認証では、1 つの TACACS+ サーバまたは複数の TACACS+ サーバのリストを使用してユーザが認証されます。

  ユーザを 1 つの TACACS+ サーバにしか設定しない場合、その TACACS+ サーバで当該ユーザに対して認可も設定する必要があります。設定しないと、そのユーザは show コマンドにしかアクセスできません。

シーケンシャルな認証リストを設定することができます。認証リストでは、ユーザの認証に使用する認証方式を定義します。この定義により、認証に使用する 1 つ以上の方式を指定することができます。したがって、最初の方式が失敗した場合は、認証のバックアップ システムが提供されます。

Detector モジュールは、最初にリストされた方式を使用してユーザを認証します。その方式が応答しない場合、Detector モジュールは 2 番目の認証方式を選択します。両方の認証方式を試してもうまくいかない場合、認証は失敗します。

分散認証方式を設定することもできます。Detector モジュールは、最初の TACACS+ サーバを使用してユーザを認証します。認証で拒否が返された場合、Detector モジュールは TACACS+ サーバ リスト、および、存在する場合は代替の認証方式（ローカル）をスキャンします。リストをすべて試してもうまくいかない場合、認証は失敗します。このオプションは、*first-hit* オプションを設定していない場合にのみ有効です。

（注）　ユーザ データベースが、複数の TACACS+ サーバ、または TACACS+ サーバとローカル ユーザ データベースに分散している場合は、**no tacacs-server first-hit** コマンドを使用してください。

認証方式の設定

Detector モジュールで使用する認証方式を設定するには、次の手順を実行します。

ステップ 1　TACACS+ 認証が必要な場合は、TACACS+ サーバ接続を設定します。詳細については、P.4-18 の「TACACS+ サーバ アトリビュートの設定」を参照してください。

ステップ 2　認証方式を定義します。次のコマンドを入力します。

```
aaa authentication {enable | login} {local | tacacs+}
[tacacs+ | local]
```

表 4-3 に、**aaa authentication** コマンドの引数を示します。

表 4-3　aaa authentication コマンドのキーワード

パラメータ	説明
enable	Detector モジュールは、上位の特権レベルに入るときに認証を行います。
login	Detector モジュールへのログイン時に認証が行われます。
local	Detector モジュールは、ローカル データベースを使用してユーザを認証します。
tacacs+	TACACS+ サーバによってユーザが認証されます。
tacacs+ | local	（オプション）設定された方式が失敗した場合の代替認証方式を設定します。

コンソール セッションから Detector モジュールにアクセスする場合、Detector モジュールは、定義されている認証方式に関係なく、認証にローカル ユーザ データベースを使用します。

認証方式を変更するには、このコマンドを再入力します。

次の例は、上位の特権レベルに入る際に認証を行うように設定する方法を示しています。最初の認証方式は TACACS+ に設定され、2 番目の認証方式はローカル ユーザ データベースに設定されています。

```
user@DETECTOR-conf# aaa authentication enable tacacs+ local
```

ローカル認証の設定

Detector モジュールには、管理者特権を持つユーザ名があらかじめ設定されています。このユーザ名を使用して新しいユーザを作成できます。ユーザを定義すると、Detector モジュールのユーザ コミュニティをドメインに分割して、必要に応じてパスワードを割り当てることができるため、セキュリティで保護され、管理されたアクセスを確立できます。

1 つの TACACS+ サーバを使用した CLI ユーザの認証をイネーブルにするには、P.4-6 の「認証の設定」を参照してください。

ユーザの追加

Detector モジュールのローカル データベースにユーザを追加するには、次のコマンドを入力します。

username *username* {**admin** | **config** | **dynamic** | **show**} [*password*]

表 4-4 に、**username** コマンドの引数とキーワードを示します。

表 4-4　username コマンドの引数とキーワード

パラメータ	説明
username	ユーザ名。1 ～ 63 文字の英数字の文字列です。大文字と小文字が区別され、先頭は英字である必要があります。この文字列にはスペースを含めることはできませんが、アンダースコアを含めることはできます。
admin | **config** | **dynamic** | **show**	ユーザの特権レベル。詳細については、表 3-1 を参照してください。
password	（オプション）パスワード。6 ～ 24 文字の文字列を入力します。スペースは使用できず、大文字と小文字が区別されます。パスワードを入力しない場合、入力するよう要求されます。

次の例を参考にしてください。

```
user@DETECTOR-conf# username Robbin config 1234
```

ユーザはクリア テキストでパスワードを入力しますが、Detector モジュールの設定ファイルでは、パスワードが暗号化されて表示されます。次に、Detector モジュールの設定ファイル（running-config）の表示例を示します。

```
username Richard config encrypted 840xdMk3
```

上の例の encrypted オプションは、パスワードが暗号化されていることを示しています。

Detector モジュールに設定されているユーザのリストを表示するには、**show running-config** コマンドまたは **show guard** コマンドを使用します。現在 CLI にログインしているユーザのリストを表示するには、**show users** コマンドを使用します。

Detector モジュールのユーザ リストからユーザを削除するには、**no username** *username* コマンドを使用します。

パスワードの変更

ユーザは、自分自身のパスワードを変更することができます。管理者は、自分自身のパスワードと、他のすべてのユーザのパスワードを変更できます。

自分自身のパスワードを変更するには、次の手順を実行します。

ステップ 1　次のコマンドを入力します。

password

ステップ 2　現在のパスワードを入力します。新しいパスワードの入力を求めるプロンプトが表示されます。

ステップ 3　新しいパスワードを入力します。パスワードは、スペースを含まない、6 ～ 24 文字の英数字の文字列である必要があります。パスワードでは大文字と小文字が区別されます。新しいパスワードをもう一度入力し、確認するように求めるプロンプトが表示されます。

次の例を参考にしてください。

```
user@DETECTOR# password
Old Password: <old-password>
New Password: <new-password>
Retype New Password: <new-password>
```

管理者は、他のユーザのパスワードを変更できます。

特定のユーザのパスワードを変更するには、次の手順を実行します。

ステップ 1　グローバル モードで次のコマンドを入力します。

password *username-password*

username-password 引数は、変更対象のパスワードを持つユーザです。

ステップ 2　新しいパスワードを入力します。パスワードは、スペースを含まない、6 ～ 24 文字の英数字の文字列である必要があります。パスワードでは大文字と小文字が区別されます。新しいパスワードをもう一度入力し、確認するように求めるプロンプトが表示されます。

次の例では、管理者はユーザ *John* のパスワードを変更しています。

```
user@DETECTOR# password Jose
New Password: <new-password>
Retype New Password: <new-password>
```

認可の設定

ユーザが使用できるサービスを制限することができます。認可をイネーブルにすると、Detector モジュールはユーザのプロファイルでそのユーザのアクセス権を確認します。プロファイルは、ローカル ユーザ データベースまたは TACACS+ セキュリティ サーバにあります。ユーザは、そのユーザのプロファイル内の情報で許可されている場合にのみ、要求したサービスへのアクセス権を付与されます。

ユーザがコマンドを実行しようとするときに Detector モジュールで使用する認可方式を設定することができます。Detector モジュールでは、次の認可オプションが提供されています。

- TACACS+ 認可：TACACS+ 認可では、TACACS+ サーバを使用してユーザが認可されます。後続のサーバが定義されている場合は、1 つのサーバとの通信が失敗した場合にのみ、そのサーバへのアクセスが開始されます。

  TACACS+ 認可では、2 種類がサポートされています。実行認可では、ユーザが Detector モジュールにログインする場合の認証時に一度ユーザの特権レベルが決定されます。コマンド認可では、ユーザがコマンドに入ると、各コマンドの認可を取得するために TACACS+ サーバに対して確認が行われます。

  TACACS+ 認可では、コマンドごとにアクセス権を指定することができます。

注意　**copy running-config** コマンドへの認可の付与には注意を払うことをお勧めします。
copy running-config コマンドの実行を認可すると、設定ファイル内ですべてのコマンドにそれぞれ認可を設定しているかどうかに関係なく、すべての設定コマンドに対して認可が与えられます。

- ローカル認可：ローカル認可では、コマンド グループのアクセス コントロールにローカルで設定されたユーザ プロファイルが使用されます。認可は、指定された特権レベルのすべてのコマンドに対して定義されます。これはデフォルトの認可方式です。

シーケンシャルな認可リストを設定することができます。認可リストでは、ユーザの認可に使用する認可方式を定義します。この定義により、認可に使用する 1 つ以上の方式を指定することができます。したがって、最初の方式に対する通信が失敗した場合は、認可のバックアップ システムが提供されます。

Detector モジュールは、最初にリストされた方式を使用してユーザを認可します。その方式が応答しない場合、Detector モジュールは 2 番目の認可方式を選択します。両方の認可方式が成功しなかった場合、認可は失敗します。

Detector モジュールのローカル認可は、TACACS+ サーバへの通信に失敗した場合に実行することができます。

ローカル認可の設定

Detector のサービスにアクセスできるかどうかは、ユーザの特権レベルによって決まります。システム管理者は、ユーザが使用できるサービスを制限することができます。Detector モジュールは、ユーザのプロファイルをチェックして、ユーザのアクセス権を確認します。認可されると、ユーザは、そのユーザのプロファイル内の情報で許可されている場合にのみ、要求したサービスへのアクセス権を付与されます。ユーザの特権レベルについては、表 3-1 を参照してください。

パスワードを使用した特権レベルの割り当て

管理者は、ユーザの特権レベルへのアクセスを制限するパスワードを設定できます。

ローカル パスワードを設定して特権レベルへのアクセスを制御するには、次のコマンドを入力します。

enable password [**level** *level*] [*password*]

表 4-5 に、**enable password** コマンドの引数を示します。

表 4-5　enable password コマンドの引数

パラメータ	説明
level	（オプション）目的の特権レベル。このレベルには、admin、config、dynamic、show のいずれかを指定できます。詳細については、表 3-1 を参照してください。デフォルトのレベルは admin です。
password	（オプション）特権レベルのパスワード。パスワードは、スペースを含まない、6 ～ 24 文字の英数字の文字列である必要があります。パスワードでは大文字と小文字が区別されます。パスワードを入力しない場合、入力するよう要求されます。

次の例を参考にしてください。

```
user@DETECTOR-conf# enable password level admin <password>
```

ユーザ特権レベル間の移動

認可されたユーザは、ユーザ特権レベル間を移動することができます。

ユーザ特権レベル間を移動するには、次の手順を実行します。

ステップ 1　次のコマンドを入力します。

```
enable [level]
```

level 引数には、目的の特権レベルを指定します。このレベルには、admin、config、dynamic のいずれかを指定できます。デフォルトのレベルは **admin** です。詳細については、表 3-1 を参照してください。

ステップ 2　特権レベルのパスワードを入力します。

次の例を参考にしてください。

```
user@DETECTOR> enable admin
Enter enable admin Password: <password>
```

下位の特権レベル（show）に戻る場合は、**disable** コマンドを使用します。

認可方式の設定

認可方式を設定するには、次の手順を実行します。

ステップ 1　TACACS+ 認可が必要な場合は、TACACS+ サーバ接続を設定します。詳細については、P.4-18 の「TACACS+ サーバ アトリビュートの設定」を参照してください。

ステップ 2　認可方式を定義します。次のいずれかのコマンドを入力します。

- **aaa authorization exec tacacs+**
- **aaa authorization commands** *level* {**local** | **tacacs+**} [**local**]

認可方式のシーケンシャルなリストを設定できます。各方式について、**aaa authorization** コマンドを入力します。認可方式を削除するには、このコマンドの no 形を使用します。

表 4-6 に、**aaa authorization** コマンドの引数とキーワードを示します。

表 4-6　aaa authorization コマンドの引数とキーワード

パラメータ	説明
exec	ユーザが EXEC シェルの実行を許可されているかどうかを判別するために認可が実行されます。Detector モジュールは、TACACS+ サーバに確認して、認証されたユーザの特権レベルを判別します。 **注意**　認可を設定する前に、TACACS+ サーバにそのユーザを設定しておく必要があります。設定しておかないと、Detector モジュールにアクセスできない可能性があります。
commands	指定された特権レベルのすべてのコマンドに対して認可が実行されます。複数の特権レベルの認可を設定するには、認可が必要な特権レベルごとにこのコマンドを発行します。
level	指定された特権レベルの認可を定義します。有効なエントリは、show、dynamic、config、および admin です。ユーザの特権レベルについては、表 3-1 を参照してください。
local	Detector モジュールのローカル データベースでユーザのアクセス権を確認します。
tacacs+	TACACS+ サーバでユーザのアクセス権を確認します。
local	（オプション）設定された認可方式が失敗した場合の代替の認可方式を設定します。

パフォーマンスへの影響を考慮し、show 特権レベルのコマンドには認可を設定しないことをお勧めします。

（注）　コンソール セッションから入力されたコマンドには、TACACS+ 認可は実行されません。

次の例は、*config* 特権レベルを必要とするコマンドの認可を設定する方法を示しています。最初の認可方式は TACACS+ に設定され、2 番目の認可方式はローカル ユーザ データベースに設定されています。

```
user@DETECTOR-conf# aaa authorization commands config tacacs+ local
```

注意　設定コマンド モードにアクセスできるようにするには、dynamic ユーザ特権レベルに対するアクセス権を付与するか、**configure** コマンドへのアクセス権を指定する必要があります。

TACACS+ サーバの設定例

TACACS+ サーバのデータベースで、各コマンドの認可を指定することができます。

次の例を参考にしてください。

```
user=Zoe {
    cmd = detect {
        permit .*
    }
    cmd = "no detect" {
        permit .*
    }
    cmd = learning {
        deny policy*
    }
    cmd = "no learning" {
        deny .*
    }
    cmd = dynamic-filter {
        permit .*
    }
    cmd = "no dynamic-filter" {
        permit .*
    }
}
```

アカウンティングの設定

アカウンティング管理により、ユーザがアクセスしているサービスを追跡し、TACACS+ サーバにアカウンティング情報を保存することができます。課金、レポート、またはセキュリティの目的のため、要求されたサービスのアカウンティングをイネーブルにします。デフォルトでは、Detector モジュールでアカウンティング管理はディセーブルに設定されています。

アカウンティングを設定するには、次の手順を実行します。

ステップ 1　TACACS+ サーバ接続を設定します。詳細については、P.4-18 の「TACACS+ サーバ アトリビュートの設定」を参照してください。

ステップ 2　複数の特権レベルのアカウンティングを設定するには、アカウンティングが必要な特権レベルごとにこのコマンドを発行します。次のコマンドを入力します。

```
aaa accounting commands {show | dynamic | config | admin} stop-only
{local | tacacs+}
```

表 4-7 に、**aaa accounting** コマンドのキーワードを示します。

表 4-7　aaa accounting コマンドのキーワード

パラメータ	説明
show | dynamic | config | admin	指定された特権レベルのアカウンティングを定義します（ユーザの特権レベルの詳細については、表 3-1 を参照してください）。
stop-only	コマンドの実行が終了したときにアクションを記録します。
tacacs+	アカウンティング情報の記録に TACACS+ サーバのデータベースを使用します。
local	アカウンティング情報を保存しません。

パフォーマンスへの影響を考慮し、アカウンティング管理は config ユーザ特権レベルに対してのみイネーブルにすることをお勧めします。

アカウンティング管理を削除するには、このコマンドの no 形を使用します。

次の例は、TACACS+ サーバ上で *config* 特権レベルを必要とするコマンドのアカウンティングを設定する方法を示しています。

```
user@DETECTOR-conf# aaa accounting commands config stop-only tacacs+
```

TACACS+ サーバ アトリビュートの設定

TACACS+ サーバのアトリビュートを設定するには、次の手順を実行します。

注意　TACACS+ 認証方式を適用する前に、TACACS+ サーバのアトリビュートを設定しておく必要があります。設定しておかないと、Detector モジュールにアクセスできない可能性があります。

ステップ 1　TACACS+ サーバの IP アドレスを設定します。詳細については、P.4-19 の「TACACS+ サーバの IP アドレスの設定」を参照してください。

ステップ 2　Detector モジュールが TACACS+ サーバへのアクセスに使用する暗号鍵を設定します。詳細については、P.4-20 の「TACACS+ サーバの暗号鍵の設定」を参照してください。

ステップ 3　（オプション）Detector モジュールが認証に使用する検索方式を設定します。詳細については、P.4-20 の「TACACS+ の検索の設定」を参照してください。

ステップ 4　（オプション）TACACS+ サーバの接続タイムアウトを設定します。詳細については、P.4-21 の「TACACS+ サーバの接続タイムアウトの設定」を参照してください。

ステップ 5　TACACS+ サーバ接続の統計情報を表示します。詳細については、P.4-22 の「TACACS+ サーバの統計情報の表示」を参照してください。

Detector モジュールのユーザ特権レベルは、TACACS+ の特権番号に次のように対応しています。

- admin = 15
- config = 10
- dynamic = 5
- show = 0

TACACS+ サーバの IP アドレスの設定

認証、認可、アカウンティングに TACACS+ サーバのシーケンシャルなリストを使用するように、Detector モジュールを設定できます。Detector モジュールは、リストされた TACACS+ サーバを使用してユーザを認証または認可するか、あるいはアカウンティング イベントを送信します。そのサーバが応答しない場合、Detector モジュールは 2 番目のサーバを選択します。リストされたすべてのサーバを試してもうまくいかない場合、認証または認可は失敗します。

または、Detector モジュールがリストの最初の TACACS+ サーバだけを使用してユーザを認証するように設定することもできます（詳細については、P.4-20 の「TACACS+ の検索の設定」を参照）。

リストには、各 TACACS+ サーバの IP アドレスを定義する必要があります。最大 9 つの TACACS+ サーバを定義できます。

リストに TACACS+ サーバを追加し、IP アドレスを割り当てるには、設定モードで次のコマンドを入力します。

tacacs-server host *ip-address*

ip-address 引数には、TACACS+ サーバの IP アドレスを指定します。

TACACS+ サーバは、入力した順序でリストに追加されます。リストには、最大 9 つのサーバを追加できます。

次の例を参考にしてください。

```
user@DETECTOR-conf# tacacs-server host 192.168.33.45
```

TACACS+ サーバの暗号鍵の設定

TACACS+ サーバにアクセスするには、暗号鍵を設定する必要があります。コマンドの一部として入力される鍵は、TACACS+ サーバ上の鍵と一致している必要があります。鍵にスペースを含めることはできません。

サーバの暗号アクセス鍵を設定するには、設定モードで次のコマンドを入力します。

tacacs-server key *tacacs-key*

引数 *tacacs-key* は、英数字の文字列です。

（注）　定義できる暗号鍵は 1 つだけです。複数の TACACS+ サーバを使用する場合、Detector モジュールは、同じ鍵を使用して、すべての TACACS+ サーバとの通信を暗号化します。

次の例は、TACACS+ サーバの暗号鍵を *TacacsKey* に設定する方法を示しています。

```
user@DETECTOR-conf# tacacs-server key <TacacsKey>
```

TACACS+ の検索の設定

Detector モジュールが、1 つの認証拒否を最終的なものと見なし、他の TACACS+ サーバやローカル認証方式を使用したそれ以上の検索を中止するように設定することができます。**tacacs-server first-hit** コマンドを使用します。Detector モジュールは、サーバ リストの、最初に応答する TACACS+ サーバだけを使用してユーザ認証を実行します。最初の TACACS+ サーバが応答しない場合、Guard はリストにある次のサーバを選択します。Guard は、ユーザ認証に対して最初に受け取る承認または拒否を最終的なものと見なし、他の TACACS+ サーバまたはローカル ユーザ データベースを使用したそのユーザ認証の試行を停止します。

TACACS+ の検索方式を first-hit に設定しない場合、Detector モジュールはデフォルトでリスト内のすべての TACACS+ サーバでユーザを認証しようとします。ユーザ認証用に first-hit 検索方式をイネーブルにする（**no tacacs-server first-hit** コマンドを使用する）と、Guard は、リスト内の最初の TACACS+ サーバを使用してユーザを認証します。最初のサーバが応答しなかった、またはユーザの認証に失敗した場合は、Guard はリストにある次のサーバを選択します。リストにあるすべての TACACS+ サーバが応答しなかった、またはユーザ認証に失敗した場合、ローカルの認証方式が設定されていないと、そのユーザ認証は失敗します。

（注）　TACACS+ の検索方式は、認証にのみ適用されます。

Detector モジュールがリストの最初の TACACS+ サーバだけを使用してユーザを認証するように設定するには、設定モードで次のコマンドを入力します。

tacacs-server first-hit

次の例を参考にしてください。

```
user@DETECTOR-conf# tacacs-server first-hit
```

TACACS+ サーバの接続タイムアウトの設定

Detector モジュールが TACACS+ サーバからの応答を待つ時間を設定できます。タイムアウトが終了すると、Detector モジュールは次の TACACS+ サーバ（そのようなサーバが設定されている場合）との接続を確立しようとするか、ローカルの AAA にフォールバックします（そのようなフォールバックが設定されている場合）。フォールバックの方式が設定されていない場合、認証と認可は失敗します。

（注）　すべての TACACS+ サーバとの通信に同じサーバ タイムアウトが使用されます。

TACACS+ サーバの接続タイムアウトを設定するには、設定モードで次のコマンドを入力します。

tacacs-server timeout *timeout*

timeout 引数には、Detector モジュールが TACACS+ サーバからの応答を待つ時間（秒）を指定します。デフォルトのタイムアウトは 0 です。

次の例を参考にしてください。

```
user@DETECTOR-conf# tacacs-server timeout 600
```

ヒント　ネットワークに問題がある場合や、小さいタイムアウト値を使用していて、TACACS+ サーバの応答が遅いためにタイムアウトが継続的に発生する場合には、タイムアウトの値を大きくすることができます。

TACACS+ サーバの統計情報の表示

TACACS+ サーバに関連する統計情報を表示することができます。統計データは、各サーバに対して提供されます。

TACACS+ 関連の統計を表示するには、**show tacacs statistics** コマンドを使用します。

TACACS+ の統計をクリアするには、**clear tacacs statistics** コマンドを使用します。

表 4-8 に、**show tacacs statistics** コマンド出力のフィールドを示します。

表 4-8　show tacacs statistics コマンド出力のフィールドの説明

フィールド	説明
PASS	サービスが TACACS+ サーバに正常にアクセスし、アクセス権を付与された回数。
FAIL	サービスが TACACS+ サーバに正常にアクセスし、アクセス権を拒否された回数。
ERROR	サービスが TACACS+ サーバにアクセスできなかった回数。

Cisco Anomaly Guard Module との通信のイネーブル化

Detector モジュールと Cisco Anomaly Guard Module（Guard モジュール）との間に安全な通信チャネルを確立して、次のタスクをイネーブルにすることができます。

- Remote activation of zone protection：Detector モジュールは、ゾーン トラフィックの異常を検出すると、通信チャネルを使用して Guard モジュールをアクティブにし、ゾーンを保護します。
- Synchronization of zone configuration information：Detector モジュールと Guard モジュールは、通信チャネルを介してゾーンの設定情報を交換します。

Detector モジュールは、次の 2 つのタイプの通信チャネルをサポートしています。

- Secure Sockets Layer（SSL）：Remote activation of zone protection および Synchronization of zone configuration information をイネーブルにします。
- セキュア シェル 2（SSH2）：Remote activation of zone protection のみをイネーブルにします。

Detector モジュールは、ゾーンを保護し、ゾーンの情報を同期させるためにアクティブにする Guard モジュールのリストを保持しています。このリストは、リモート Guard リストと呼ばれます。Detector モジュールは、リモート Guard リストに設定されている各 Guard モジュールと通信チャネルを確立します。

通信チャネルを確立する前に、必要なリモート Guard リストに Guard モジュールを追加しておく必要があります。詳細については、P.5-45 の「リモート Guard のアクティブ化」を参照してください。

Detector モジュールは、各 Guard モジュールとの SSH 通信チャネルを確立してから、SSL 通信チャネルを確立します。したがって、両方のタイプのリストを設定済みである場合、SSH 通信チャネルの設定を行う必要はありません。SSH 通信チャネルは SSL 通信チャネルによって設定されるためです。

この項では、次のトピックについて取り上げます。

SSL 通信チャネルの設定

Guard モジュールと Detector モジュールは、通信チャネルに SSL 接続を使用します。SSL は、プライバシー、認証、およびデータ整合性を組み合せることにより、接続のセキュリティを確保します。SSL の高度なセキュリティは、デジタル証明書、秘密と公開の鍵交換ペア、および Diffie-Hellman 鍵合意パラメータによって実現されます。

SSL では、デバイス認証とデータ暗号化により、安全なネットワーク通信チャネルが提供されます。それぞれの Guard モジュールと Detector モジュールは、その通信チャネルを介してデバイスと通信を試みることで、デバイスを認証します。デバイス認証は、デジタル証明書とデバイス固有の情報（IP アドレスなど）を使用して実行されます。SSL は、Guard モジュールと Detector モジュールが交換するデータを暗号化します。データを解読できるのは、指定された受信者だけです。

安全な接続を確保するために、Detector モジュールは秘密鍵と公開鍵のペアを生成し、公開鍵をリモート Guard リスト内の Guard モジュールに配布します。

Guard モジュールで通信チャネル サービスをイネーブルにした後で、Detector モジュールから通信チャネルを確立します。Detector モジュールは、最初に、Guard モジュール上のユーザ *riverhead* に対して SSH2 接続を確立します。次に Detector モジュールは、安全な SSH2 接続を使用して、SSL 接続鍵を交換します。

この項では、次のトピックについて取り上げます。

- SSL 通信チャネルの確立
- SSL 証明書の再生成

SSL 通信チャネルの確立

Guard モジュールと Detector モジュールとの間に SSL 通信チャネルを確立するには、次のタスクを実行する必要があります。

1. Guard モジュールと Detector モジュールの両方で、通信チャネル サービスをイネーブルにします。
2. Guard モジュールと Detector モジュールの両方で、その通信チャネル サービスへのアクセスを許可します。
3. Detector モジュールから通信チャネルを確立します。

Detector モジュールは、最初に、Guard モジュール上のユーザ *riverhead* に対して SSH2 接続を確立します。次に Detector モジュールは、安全な SSH2 接続を使用して、SSL 接続鍵を交換します。

注意　Guard　モジュールで　TACACS+　認証を使用してユーザを認証している場合、Detector モジュールで SSH2 接続を確立するためには、TACACS+ サーバにユーザ *riverhead* を定義する必要があります。

Detector モジュールで通信チャネルをイネーブルにするには、Detector モジュールで次の手順を実行します。

（注）　Guard モジュールで通信チャネルをイネーブルにする場合は、Guard モジュールで同じコマンドを使用してください。

ステップ 1　設定モードで **permit ssh** *ip-address-general* [*ip-mask*] と入力して、Guard モジュールの IP アドレスから Detector モジュール上の SSH サービスへのアクセスを許可します。

引数 *ip-address-general* および *ip-mask* には、Detector モジュールへのアクセス権を付与する Guard モジュールの IP アドレスを定義します。

（注）　SSH サービスはすでにイネーブルになっているので、ここでイネーブルにする必要はありません。

ステップ 2　設定モードで **service internode-comm**　コマンドを入力して、通信チャネル サービスをイネーブルにします。

ステップ 3　設定モードで **permit internode-comm** *ip-address-general* [*ip-mask*] コマンドを入力して、Guard モジュールの IP アドレスから通信チャネル サービスへのアクセスを許可します。

引数 *ip-address-general* および *ip-mask* には、Detector モジュールへのアクセス権を付与する Guard モジュールの IP アドレスを定義します。

（注）　SSL 証明書内の Guard モジュールと Detector モジュールの ID は、IP アドレスに関連付けられています。通信チャネルのそれぞれの端にある Guard モジュールまたは Detector モジュールのいずれかの IP アドレスを変更する場合、SSL 証明書を再生成する必要があります。P.4-28 の「SSL 証明書の再生成」を参照してください。

Detector モジュールから通信チャネルを確立するには、Detector モジュール上で次の手順を実行します。

ステップ 1　SSH2 秘密および公開鍵ペアを生成します。

設定モードで次のコマンドを入力します。

```
key generate
```

ステップ 2　SSH2 の鍵ペアがすでに存在している場合は、次のメッセージが表示されます。

```
/root/.ssh/id_rsa already exists.
Overwrite (y/n)?
```

目的のオプションを入力します。

次のいずれかのオプションを入力します。

- y：Detector モジュールは新しい SSH2 鍵ペアを生成してから、リモート Guard リスト内の各 Guard モジュールと公開鍵を交換し、SSL 証明書を生成して、SSL 証明書を交換します。
- n：Detector モジュールは新しい SSH2 鍵ペアを生成しません。Detector モジュールは、リモート Guard リスト内の各 Guard モジュールと現在の公開鍵を交換し、SSL 証明書を生成して、SSL 証明書を交換します。

ステップ 3　man-in-the-middle アタックを防ぐために、SSH2 ではホスト鍵を使用してリモート ホストの確実性が確認されます。リモート ホストに対して初めて SSH 接続を開始すると、リモート ホストはそのホストの公開鍵を送信します。これが Detector モジュールから Guard モジュールに対して開始される最初の接続である場合は、次のメッセージが表示されます。

```
The authenticity of host '<remote-host name> (<remote-host IP
address>)' can't be established.
RSA key fingerprint is <RSA key fingerprint>
Are you sure you want to continue connecting (yes/no)?
```

yes と入力します。

次のプロンプト行が表示されます。

```
riverhead@remote-Guard-IP-address's password:
```

ステップ 4　Guard モジュールでユーザ **riverhead** 用に設定されたパスワードを入力します。

Detector モジュールは、SSL リモート Guard リスト内の各 Guard モジュールと SSL 通信チャネルを確立します。各リモート Guard について、ステップ 3 とステップ 4 を繰り返します。

次の例は、Detector モジュールと Guard モジュールの間で SSL 通信チャネルを確立する方法を示しています。

```
user@DETECTOR-conf# key generate
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
The authenticity of host '192.168.100.33 (192.168.100.33)' can't be
established.
RSA key fingerprint is
de:cb:ac:36:9a:fe:33:f9:6a:d8:7b:98:a9:51:75:54.
Are you sure you want to continue connecting (yes/no)? yes
riverhead@192.168.100.33's password: <password>
user@DETECTOR-conf#
```

Detector モジュールの SSH 公開鍵を表示するには、**show public-key** コマンドを使用します。

Detector モジュールにリストされているホスト鍵を表示するには、**show host-keys** コマンドを使用します。

SSL 証明書の再生成

SSL 証明書内の Guard モジュールと Detector モジュールを識別する鍵は、IP アドレスに関連付けられています。

次のいずれかの変更を行う場合、通信チャネルの両端にある Guard モジュールと Detector モジュール用に新しい SSL 証明書を生成する必要があります。

- いずれか一方のデバイスの IP アドレスを変更する。
- いずれか一方のデバイスを交換（スワップアウト）する。

新しい SSL 証明書を生成するには、まず、現在使用している証明書を両方のデバイスで削除する必要があります。

現在使用している SSL 証明書を消去するには、次の手順を実行します。

ステップ 1　Guard モジュールの SSL 証明書を Detector モジュールから削除します。

設定モードで次のコマンドを入力します。

```
cert remove cert-host-ip
```

cert-host-ip 引数には、Guard モジュールの IP アドレスを指定します。すべての Guard モジュールの SSL 証明書を削除するには、アスタリスク (*) を入力します。

ステップ 2　Detector モジュールの SSL 証明書を Guard モジュールから削除します。

ステップ 3　Guard モジュール デバイスを物理的に交換した場合は、Guard モジュールの SSH ホスト鍵を Detector モジュールから削除する必要があります。

Detector モジュールにリストされている SSH ホスト鍵を削除するには、設定モードで次のコマンドを入力します。

```
no host-keys ip-address-general
```

ip-address-general 引数には、リモート デバイスの IP アドレスを指定します。

ステップ 4　Guard モジュールと Detector モジュールの間に新しい SSL 通信チャネルを確立します。詳細については、P.4-24 の「SSL 通信チャネルの確立」を参照してください。

次の例は、SSL 証明書を削除する方法を示しています。

```
user@DETECTOR-conf# cert remove 10.56.36.4
```

SSH 通信チャネルの設定

トラフィックの異常を検出すると、Detector モジュールはそのイベントをログに記録するか、SSH 接続を使用して Guard モジュールをアクティブにしてゾーンを保護します。SSH 通信チャネルを使用している場合、Detector モジュールは次のタスクを実行できません。

- ゾーン設定を同期化する。
- Guard モジュールを監視して、ゾーンへの攻撃が終了したことを確認する。したがって、Detector モジュールを検出およびラーニングの動作状態でアクティブにしても、Detector モジュールはゾーンへの攻撃が終了したことを確認できないため、リモート Guard をアクティブにした後でゾーン トラフィックのラーニングを続行しません。

安全な SSH2 接続を確保するために、Detector モジュールは秘密 SSH 鍵と公開 SSH 鍵のペアを生成し、公開 SSH 鍵を Detector モジュールのリモート Guard リスト内の Guard モジュールに配布します。

この項では、次のトピックについて取り上げます。

SSH 通信チャネルの確立

Guard モジュールと Detector モジュールとの間に通信チャネルを確立するには、次のタスクを実行する必要があります。

1. **permit ssh** コマンドを入力して、Detector モジュールの IP アドレスから Guard モジュール上の SSH サービスへのアクセスを許可します。
2. Detector モジュールから SSH 通信チャネルを確立します。

Guard モジュール デバイスを交換（スワップアウト）する場合は、SSH 通信チャネルを再生成する必要があります。P.4-33 の「SSH 通信チャネル鍵の再生成」を参照してください。

Detector モジュールから通信チャネルを確立するには、Detector モジュール上で次の手順を実行します。

注意　Guard で TACACS+ 認証を使用してユーザを認証している場合、**key generate** コマンドを機能させるためには、TACACS+ サーバにユーザ *riverhead* を定義する必要があります。

ステップ 1　SSH2 秘密および公開鍵ペアを生成します。

設定モードで次のコマンドを入力します。

```
key generate
```

ステップ 2　SSH2 の鍵ペアがすでに存在している場合は、次のメッセージが表示されます。

```
/root/.ssh/id_rsa already exists.
Overwrite (y/n)?
```

目的のオプションを入力します。

次のいずれかのオプションを入力します。

- y：Detector モジュールは新しい SSH2 鍵ペアを生成してから、リモート Guard リスト内の各 Guard モジュールと公開鍵を交換します。
- n：Detector モジュールは新しい SSH2 鍵ペアを生成しません。Detector モジュールは、リモート Guard リスト内の各 Guard モジュールと現在の公開鍵を交換します。

ステップ 3　man-in-the-middle アタックを防ぐために、SSH2 ではホスト鍵を使用してリモート ホストの確実性が確認されます。リモート ホストに対して初めて SSH2 接続を開始すると、リモート ホストはそのホストの公開鍵を送信します。これが Detector モジュールから Guard モジュールに対して開始される最初の接続である場合は、次のメッセージが表示されます。

```
The authenticity of host '<remote-host name> (<remote-host IP
address>)' can't be established.
RSA key fingerprint is <RSA key fingerprint>
Are you sure you want to continue connecting (yes/no)?
```

yes と入力します。

次のプロンプト行が表示されます。

```
riverhead@remote-Guard-IP-address's password:
```

ステップ 4　Guard モジュールでユーザ **riverhead** 用に設定されたパスワードを入力します。

Detector モジュールは、SSH リモート Guard リスト内の各 Guard モジュールと SSH 通信チャネルを確立します。各リモート Guard について、ステップ 3 とステップ 4 を繰り返します。

次の例は、Detector モジュールと Guard モジュールの間で SSH 通信チャネルを確立する方法を示しています。

```
user@DETECTOR-conf# key generate
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
The authenticity of host '192.168.100.33 (192.168.100.33)' can't be
established.
RSA key fingerprint is
de:cb:ac:36:9a:fe:33:f9:6a:d8:7b:98:a9:51:75:54.
Are you sure you want to continue connecting (yes/no)? yes
riverhead@192.168.100.33's password: <password>
user@DETECTOR-conf#
```

リモート Guard リストにリモートの Guard が設定されていない場合や、Detector モジュールとリモートの Guard の間にその時点でネットワーク接続がない場合は、リモートの Guard に手動で SSH 公開鍵を追加してもかまいません。

Detector モジュールの SSH 公開鍵を表示するには、**show public-key** コマンドを使用します。

Detector モジュールにリストされているホスト鍵を表示するには、**show host-keys** コマンドを使用します。

SSH 通信チャネル鍵の再生成

Guard モジュールのデバイスを物理的に交換する場合、次の手順を実行して新しい SSH 通信チャネルを確立します。

ステップ 1　SSH2 ホスト鍵を Detector モジュールから削除します。設定モードで Detector モジュールから **no host-keys** *ip-address-general* コマンドを入力します。

ip-address-general 引数には、リモート デバイスの IP アドレスを指定します。

Detector モジュールにリストされているホスト鍵を表示するには、**show host-keys** コマンドを使用します。

ステップ 2　次のいずれかのアクションを実行します。

- Detector モジュールから新しい SSH 通信チャネルを確立する（P.4-30 の「SSH 通信チャネルの確立」を参照）。

  または

- Detector モジュールの公開鍵をリモート Guard に手動で追加する。

  Detector モジュールの公開 SSH 鍵を表示するには、Detector モジュールで **show public-key** コマンドを使用します。

  Detector モジュールの公開 SSH 鍵を設定するには、Guard モジュールで **key add** コマンドを使用します。

日付と時刻の設定

時刻と日付を設定するには、設定モードで次のコマンドを入力します。

date *MMDDhhmm*[[*CC*]*YY*][*.ss*]

表 4-9 に、**date** コマンドの引数とキーワードを示します。

表 4-9　date コマンドの引数

パラメータ	説明
MM	数字で表した月。
DD	月の日付。
hh	24 時間表記の時間。
mm	分。
CC	（オプション）年の最初の 2 桁（たとえば、**20**05）。
YY	（オプション）年の最後の 2 桁（たとえば、20**05**）。
.ss	（オプション）秒（小数点が必要）。

次の例は、日付を *2003* 年 *10* 月 *8* 日、時刻を午後 *5* 時 *10* 分（1710）*17* 秒に設定する方法を示しています。

```
user@DETECTOR-conf# date 1008171003.17
Wed Oct  8 17:10:17 EDT 2003
```

SSH 鍵の管理

Detector モジュールは、安全なリモート ログインのために SSH をサポートしています。SSH 鍵のリストを追加して、ログイン名とパスワードを入力せずにリモート デバイスから Detector モジュールへの安全な通信をイネーブルにすることができます。

次の各項では、Detector モジュールの SSH 鍵リストの操作方法について説明します。

SSH 鍵の追加

ログイン名とパスワードを入力せずに SSH 接続をイネーブルにするには、Detector モジュールの SSH 鍵リストにリモート接続の SSH 公開鍵を追加します。

設定モードで次のコマンドを入力します。

key add [user-*name*] {**ssh-dsa** | **ssh-rsa**} *key-string comment*

表 4-10 に、**key add** コマンドの引数とキーワードを示します。

表 4-10　key add コマンドの引数とキーワード

パラメータ	説明
user-name	（オプション）指定されたユーザの SSH 鍵を追加します。他のユーザの SSH 鍵を追加できるのは管理者だけです。 デフォルトは、現行ユーザの SSH 鍵の追加です。
ssh-dsa	SSH2-DSA 鍵のタイプ。
ssh-rsa	SSH2-RSA 鍵のタイプ。
key-string	Detector またはリモート端末で作成された公開 SSH 鍵。鍵ストリングは、8,192 ビットまでに制限されています。 鍵タイプの識別（ssh-rsa または ssh-dsa）を除いた完全な鍵をコピーする必要があります。

表 4-10　key add コマンドの引数とキーワード（続き）

パラメータ	説明
comment	デバイスの説明。コメントの形式は、通常、鍵の生成に使用されるユーザとマシンを表す user@hostname になります。たとえば、Detector で生成される SSH 公開鍵に使用されるデフォルトのコメントは、**root@DETECTOR** です。

次の例を参考にしてください。

```
user@DETECTOR-conf# key add ssh-rsa 14513797528175730. .
.user@Detector module.com
```

SSH 鍵の削除

リストから SSH 鍵を削除できます。SSH 鍵を削除すると、次に Detector モジュールと SSH セッションを確立するときには認証を受ける必要があります。

Detector モジュールから SSH 鍵を削除するには、設定モードで次のコマンドを入力します。

key remove [*user-name*] *key-string*

表 4-11 に、**key remove** コマンドの引数を示します。

表 4-11　key remove コマンドの引数

パラメータ	説明
user-name	（オプション）指定のユーザの SSH 鍵を削除します。 他のユーザの SSH 鍵を削除できるのは管理者だけです。デフォルトは、現行ユーザの SSH 鍵の削除です。
key-string	削除する公開 SSH 鍵。 プロンプトに SSH 公開鍵をペーストします。識別フィールド（ssh-rsa または ssh-dsa）は除き、鍵だけをペーストしてください。

次の例は、key remove コマンド用にカット アンド ペーストを行えるように、ユーザ鍵を表示する方法を示しています。

```
user@DETECTOR-conf# show keys Lilac
ssh-rsa 2352345234523456... user@Detector module.com
user@DETECTOR-conf# key remove Lilac 2352345234523456...
```

SFTP 接続用の鍵の設定

Detector モジュールは、SSH2 の上の層の Secure FTP（SFTP）をサポートしています。SFTP では、公開鍵による認証と強力なデータ暗号化を使用しています。これにより、ログイン、データ、およびセッションの情報が送信中に傍受されたり変更されたりすることを防止できます。

SFTP サーバに公開鍵を設定するには、Detector モジュールで次の操作を実行します。

ステップ 1　Detector モジュールの公開鍵を表示します。設定モードで **show public-key** コマンドを使用します。

鍵が存在する場合は、ステップ 2 を省略してステップ 3 に進みます。

鍵が存在しない場合は、ステップ 2 に進みます。

ステップ 2　秘密および公開鍵ペアを生成します。設定モードで次のコマンドを入力します。

```
key generate
```

注意　秘密および公開鍵ペアが存在する場合は、再生成しないことをお勧めします。Detector モジュールは、Detector モジュールのデフォルトのリモート Guard リストまたはゾーンのリモート Guard リストに設定されているすべてのリモート Guard との接続を確立して、公開鍵を交換します。不必要に鍵ペアを再生成すると、現在オンラインになっていないリモート Guard と後で通信するときに問題が発生する可能性があります。

SSH の鍵ペアがすでに存在している場合は、次のメッセージが表示されます。

```
/root/.ssh/id_rsa already exists.
Overwrite (y/n)?
```

目的のオプションを入力します。

Detector モジュールが公開および秘密鍵ペアを作成します。Detector モジュールの公開鍵を表示するには、設定モードで **show public-key** コマンドを使用します。

ステップ 3　公開鍵をコピーし、SFTP サーバ上の鍵ファイル内にペーストします。

たとえば、ユーザ root で Linux オペレーティング システムにインストールされている SFTP サーバに接続する場合は、/root/.ssh/authorized_keys2 ファイルに Detector モジュールの公開鍵を追加します。

鍵は 1 行に収まるようにコピーしてください。鍵が 2 行としてコピーされた場合は、1 行目の末尾にある改行文字を削除します。

ホスト名の変更

Detector モジュールのホスト名を変更できます。この変更はすぐに反映され、新しいホスト名は自動的に CLI プロンプト ストリングに組み込まれます。

Detector モジュールのホスト名を変更するには、設定モードで次のコマンドを入力します。

hostname *name*

name 引数には、新しいホスト名を指定します。

次の例を参考にしてください。

```
user@DETECTOR-conf# hostname CiscoDetector
admin@CiscoDetector-conf#
```

SNMP トラップのイネーブル化

Detector モジュールが SNMP トラップを送信し、Detector モジュールで発生する重大なイベントを管理者に通知するように設定することができます。Detector モジュールの SNMP トラップ ジェネレータを設定し、トラップ情報のスコープを定義できます。

トラップのログは、Detector モジュールのイベント ログに記録され、トラップ条件が発生すると、SNMP エージェントがトラップを送信するかどうかに関係なく、イベント モニタに表示されます。

Detector モジュールが SNMP トラップを送信するように設定するには、次の手順を実行します。

ステップ 1　SNMP トラップ ジェネレータ サービスをイネーブルにします。設定モードで次のコマンドを入力します。

```
service snmp-trap
```

ステップ 2　SNMP トラップ ジェネレータのパラメータ（トラップの宛先 IP アドレスとトラップ情報のスコープ）を設定します。次のコマンドを入力します。

```
snmp trap-dest ip-address [community-string [min-severity]]
```

表 4-12 に、**snmp trap-dest** コマンドの引数を示します。

表 4-12　snmp trap-dest コマンドの引数

パラメータ	説明
ip-address	宛先ホストの IP アドレス。
community-string	（オプション）トラップとともに送信されるコミュニティ ストリング。このストリングは、宛先ホスト用に定義されたコミュニティ ストリングと一致する必要があります。デフォルトのコミュニティ ストリングは、*public* です。1 ～ 15 文字の英数字の文字列を入力します。この文字列にスペースを含めることはできません。

表 4-12　snmp trap-dest コマンドの引数（続き）

パラメータ	説明
min-severity	（オプション）トラップ情報のスコープ。重大度レベルの範囲の下限を指定してスコープを定義します。この定義により、トラップは指定された重大度レベル以上のすべてのイベントを表示します。たとえば、重大度レベル *Warnings* を指定すると、トラップは *Warnings* から *Emergencies* までのすべての重大度レベルのイベントを表示します。重大度レベルのオプションを次に示します。 • **Emergencies**：システムは使用不能（重大度 = 0） • **Alerts**：即時のアクションが必要（重大度 = 1） • **Critical**：危険な状態（重大度 = 2） • **Errors**：エラー状態（重大度 = 3） • **Warnings**：警告状態（重大度 = 4） • **Notifications**：正常ではあるが、重要な状態（重大度 =5） • **Informational**：情報通知のためのメッセージ（重大度 = 6） • **Debugging**：デバッグ メッセージ（重大度 = 7） デフォルトでは、レポートにはすべての重大度レベルのイベントが表示されます。

SNMP トラップ ジェネレータ パラメータを削除するには、**no snmp trap-dest** コマンドを使用します。すべての SNMP トラップ宛先パラメータを削除するには * を使用します。

次に、*errors* 以上の重大度レベルのトラップが、SNMP コミュニティ ストリング *tempo* とともに宛先 IP アドレス *192.168.100.52* に送信される例を示します。

```
user@DETECTOR-conf# snmp trap-dest 192.168.100.52 tempo errors
```

SNMP コミュニティ ストリングの設定

Detector モジュールの SNMP サーバにアクセスして、管理情報ベース 2（MIB2）および Riverhead の専用 MIB によって定義されている情報を取得できます。コミュニティ ストリングはパスワードのように機能し、Detector モジュールの SNMP エージェントからの読み取りアクセスを許可します。Detector モジュールの SNMP コミュニティ ストリングを設定して、異なる組織のクライアントが異なるコミュニティ ストリングを使用して SNMP エージェントにアクセスできるようにすることができます。

SNMP コミュニティ ストリングを追加するには、設定モードで次のコマンドを入力します。

snmp community *community-string*

引数 *community-string* には、目的の Detector モジュールのコミュニティ ストリングを指定します。1 ～ 15 文字の英数字の文字列を入力します。この文字列にスペースを含めることはできません。Detector モジュールのデフォルトのコミュニティ ストリングは *riverhead* です。コミュニティ名はいくつでも指定できます。コミュニティ ストリングを削除するには、**no community string** コマンドを使用します。すべての SNMP コミュニティ ストリングを削除するには * を使用します。

次の例を参考にしてください。

```
user@DETECTOR-conf# snmp community tempo
```