オブジェクト トラッキングの設定

ここでは、Cisco IOS XR ネットワークでのオブジェクト トラッキングの設定について説明します。このモジュール内に記載されているコマンドの詳細については、「**その他の関連資料**」の項を参照してください。設定タスクを実行する手順の中で出現する可能性のあるその他のコマンドについて記載されたマニュアルを検索するには、トピック「その他の関連資料」の「**テクニカル ドキュメント**」の項を参照してください。

オブジェクト トラッキングの設定

ここでは、Cisco IOS XR ネットワークでのオブジェクト トラッキングの設定について説明します。このモジュール内に記載されているコマンドの詳細については、「**その他の関連資料**」の項を参照してください。設定タスクを実行する手順の中で出現する可能性のあるその他のコマンドについて記載されたマニュアルを検索するには、トピック「その他の関連資料」の「**テクニカル ドキュメント**」の項を参照してください。

オブジェクト トラッキングの実装の前提条件

適切なタスク ID を含むタスク グループに関連付けられているユーザ グループに属している必要があります。このコマンド リファレンスには、各コマンドに必要なタスク ID が含まれます。ユーザ グループの割り当てが原因でコマンドを使用できないと考えられる場合、AAA 管理者に連絡してください。

オブジェクト トラッキングについて

オブジェクト トラッキング とは、オブジェクトを追跡して、そのプロパティの変化に基づいて、トラッキング対象オブジェクトとは関係のない別のオブジェクトに対してアクションを実行する仕組みです。

各トラッキング対象オブジェクトは、トラッキング コマンドライン インターフェイス（CLI）で指定された一意の名前で識別されます。Cisco IOS XR が処理し、この名前を使用して特定のオブジェクトを追跡します。

トラッキング プロセスでは、定期的にトラッキング対象オブジェクトをポーリングして、ステートのアップ、ダウンなどの変化をユーザの指定により即時または時間をおいてレポートします。

リストを使った方法で複数のオブジェクトを追跡することもできます。リストはオブジェクトの組み合わせにブール論理式を使った柔軟なメソッドです。リストでは次の演算を使用します。

- **ブール AND 関数**：トラッキング対象リストにブール AND 関数を指定した場合、サブセット内に定義された各オブジェクトはアップ ステートでなければならないため、トラッキング対象オブジェクトもアップ ステートになります。
- **ブール OR 関数**：トラッキング対象リストにブール OR 関数を指定した場合、サブセット内に定義されたオブジェクトのうち少なくとも 1 つがアップ ステートでなければならないため、トラッキング対象オブジェクトもアップ ステートであることを意味します。

オブジェクト トラッキングの実装方法

ここでは、さまざまなオブジェクト トラッキングの手順を説明します。

インターフェイスのライン プロトコル ステートのトラッキング

インターフェイスのラインプロトコル ステートをトラッキングするには、グローバル コンフィギュレーション モードで次の作業を実行します。

インターフェイスのラインプロトコルがアップしている場合は、トラッキング対象オブジェクトはアップ状態と見なされます。

トラッキング対象オブジェクトの設定後、そのステートがトラッキング対象になっているインターフェイスを関連付けたり、トラッキング オブジェクトがインターフェイスをポーリングしてステートを取得するまで待機する秒数を指定したりすることができます。

手順

	コマンドまたはアクション	目的
ステップ 1	**configure**	

	コマンドまたはアクション	目的
ステップ 2	**track** *track-name* 例： `RP/0/RP0/CPU0:router(config)# track track1`	トラック コンフィギュレーション モードを開始します。 • *track-name*：トラッキングの対象となるオブジェクト名を指定します。
ステップ 3	**type line-protocol state** 例： `RP/0/RP0/CPU0:router(config-track)# type line-protocol state`	インターフェイスのライン プロトコルに基づいてトラッキングを作成します。
ステップ 4	**interface** *type interface-path-id* 例： `RP/0/RP0/CPU0:router(config-track-line-prot)# interface atm 0/2/0/0.1`	プロトコル ステートをトラッキングするインターフェイスを指定します。 • *type*：インターフェイス タイプを指定します。詳細については、疑問符（?）オンライン ヘルプ機能を使用します。 • *interface-path-id*：物理インターフェイスまたは仮想インターフェイスを識別します。 （注） ルータに現在設定されている可能性があるすべてのインターフェイスのリストを表示するには、 **show interfaces** コマンドを使用します。 （注） ループバック インターフェイスおよびヌル インターフェイスは、常にアップ ステートであり、そのためトラッキングできません。
ステップ 5	**exit** 例： `RP/0/RP0/CPU0:router(config-track-line-prot)# exit`	トラック ライン プロトコル コンフィギュレーション モードを終了します。
ステップ 6	（任意） **delay** {**up** *seconds* | **down** *seconds*} 例： `RP/0/RP0/CPU0:router(config-track)# delay up 10`	オブジェクトがアップかダウンかのトラッキング間に発生可能な遅延をスケジューリングします。

	コマンドまたはアクション	目的

ステップ 7	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	例：
	`RP/0/RP0/CPU0:router(config-track)# end`
	または
	`RP/0/RP0/CPU0:router(config-track)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、次に示す変更のコミットを求めるプロンプトが表示されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		• **yes** と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータがEXECモードに戻ります。
		• **no** と入力すると、コンフィギュレーション セッションが終了して、ルータがEXECモードに戻ります。変更はコミットされません。
		• **cancel** と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 実行コンフィギュレーション ファイルに設定変更を保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

IP ルートの到達可能性のトラッキング

ホストまたはネットワークがリモート サイトでダウン状態になった場合、ルーティング プロトコルはルータに通知し、ルーティング テーブルはそれに応じて更新されます。ルーティング プロセスは、ルーティング アップデートによってルートの状態が変わった場合にトラッキング プロセスに通知するように設定されます。

ルーティング テーブル エントリがルートに存在し、そのルートがアクセス可能であると、トラッキング対象オブジェクトはアップ状態にあると見なされます。

手順

	コマンドまたはアクション	目的
ステップ 1	**configure**	
ステップ 2	**track** *track-name* 例： `RP/0/RP0/CPU0:router(config)# track track1`	トラック コンフィギュレーション モードを開始します。 • *track-name*：トラッキングの対象となるオブジェクト名を指定します。
ステップ 3	**type route reachability** 例： `RP/0/RP0/CPU0:router(config-track)# type route reachability vrf internet`	ルーティングアップデートによってルートの状態が変わった場合にトラッキング プロセスに通知するようにルーティング プロセスを設定します。

ステップ 4	次のいずれかのコマンドを使用します。
	• **vrf** *vrf-table-name*
	• **route ipv4** *IP-prefix/mask*
	例：
	`RP/0/RP0/CPU0:router(config-track-route)# vrf vrf-table-4`
	または
	`RP/0/RP0/CPU0:router(config-track-route)# route ipv4 10.56.8.10/16`
		トラッキングする IP ルートのタイプを設定します。これは、ルータのタイプによって次のいずれかで構成可能です。
		• *vrf-table-name*：VRF テーブル名。
		• *IP-prefix/mask*：ネットワークとサブネット マスクからなる IP プレフィックス（例：10.56.8.10/16）。

ステップ 5	**exit** 例： `RP/0/RP0/CPU0:router(config-track-line-prot)# exit`	トラック ライン プロトコル コンフィギュレーション モードを終了します。
ステップ 6	（任意） **delay** {**up** *seconds* | **down** *seconds*} 例： `RP/0/RP0/CPU0:router(config-track)# delay up 10`	オブジェクトがアップかダウンかのトラッキング間に発生可能な遅延をスケジューリングします。
ステップ 7	**commit**	

オブジェクト リストに基づくトラッキングの設定

グローバル コンフィギュレーション モードでこのタスクを実行し、ブール式を使用してリストの状態を判断して、トラッキング対象オブジェクト リスト（ここではインターフェイスまたはプレフィックスのリスト）を作成します。

トラッキング対象リストには 1 つまたは複数のオブジェクトが含まれます。ブール式では、AND または OR 演算子を使用して 2 種類の演算を実行できます。たとえば、AND 演算子を使用して 2 つのインターフェイスをトラッキングする場合、アップは両方のインターフェイスがアップ状態であることを意味し、ダウンはいずれか一方のインターフェイスがダウン状態であることを意味します。

（注） トラッキング対象リストにオブジェクトを追加するには、そのオブジェクトが存在している必要があります。

NOT 演算子は、1 つまたは複数のオブジェクトに指定し、そのオブジェクトの状態を否定します。

トラッキング対象オブジェクトを設定したら、状態をトラッキングするインターフェイスを関連付ける必要があります。オプションとして、トラッキング オブジェクトがインターフェイスをポーリングしてその状態を取得するまでの待機時間を秒数で指定できます。

手順

	コマンドまたはアクション	目的
ステップ 1	**configure**	
ステップ 2	**track** *track-name* **例**： `RP/0/RP0/CPU0:router(config)# track track1`	トラック コンフィギュレーション モードを開始します。 • *track-name*：トラッキングの対象となるオブジェクト名を指定します。
ステップ 3	**type list boolean** { **and** | **or** } **例**： `RP/0/RP0/CPU0:router(config-track-list)# type list boolean and`	ブール リスト オブジェクトを設定し、トラッキング リスト コンフィギュレーション モードを開始します。 • **boolean**：トラッキング対象リストのステートがブール式に基づいて決まることを指定します。 • **and**：リストについて、すべてのオブジェクトがアップの場合はアップ、ダウンのオブジェクトが 1 つ以上ある場合はダウンになるように指定します。たとえば 2 つのインターフェイスをトラッキングする場合、

	コマンドまたはアクション	目的
		アップは両方のインターフェイスがアップ状態であることを意味し、ダウンはいずれか一方のインターフェイスがダウン状態であることを意味します。 • **or**：少なくとも1つのオブジェクトがアップであればリストがアップになるように指定します。たとえば2つのインターフェイスをトラッキングする場合、アップはいずれか一方のインターフェイスがアップ状態であることを意味し、ダウンは両方のインターフェイスがダウン状態であることを意味します。
ステップ4	**object** *object-name* ［ **not** ］ **例**： `RP/0/RP0/CPU0:router(config-track-list)# object 3 not`	リストによるトラッキングの対象となるオブジェクトを指定します。 • *obect-name*：トラッキングするオブジェクトの名前。 • **not**：オブジェクトの状態を否定します。
ステップ5	**exit** **例**： `RP/0/RP0/CPU0:router(config-track-line-prot)# exit`	トラック ライン プロトコル コンフィギュレーション モードを終了します。
ステップ6	（任意） **delay** {**up** *seconds* | **down** *seconds*} **例**： `RP/0/RP0/CPU0:router(config-track)# delay up 10`	オブジェクトがアップかダウンかのトラッキング間に発生可能な遅延をスケジューリングします。
ステップ7	次のいずれかのコマンドを使用します。 • **end** • **commit** **例**： `RP/0/RP0/CPU0:router(config-track)# end` または	設定変更を保存します。 • **end** コマンドを実行すると、次に示す変更のコミットを求めるプロンプトが表示されます。 `Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`

	コマンドまたはアクション	目的
	`RP/0/RP0/CPU0:router(config-track)# commit`	• **yes** と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータがEXECモードに戻ります。 • **no** と入力すると、コンフィギュレーション セッションが終了して、ルータがEXECモードに戻ります。変更はコミットされません。 • **cancel** と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。 • 実行コンフィギュレーション ファイルに設定変更を保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

オブジェクト リストに基づくトラッキングの設定：しきい値の割合

グローバル コンフィギュレーション モードでこのタスクを実行し、しきい値の割合を使用してリストの状態を判断して、トラッキング対象オブジェクト リスト（ここではインターフェイスまたはプレフィックスのリスト）を作成します。

手順

	コマンドまたはアクション	目的
ステップ 1	**configure**	
ステップ 2	**track** *track-name* 例： `RP/0/RP0/CPU0:router(config)# track track1`	トラック コンフィギュレーション モードを開始します。 • *track-name*：トラッキングの対象となるオブジェクト名を指定します。

	コマンドまたはアクション	目的
ステップ 3	**type list threshold percentage** 例： `RP/0/RP0/CPU0:router(config-track-list)# type list threshold percentage`	トラッキングのタイプにしきい値の割合リストを設定します。

ステップ 4	**object** *object-name*
	例：
	`RP/0/RP0/CPU0:router(config-track-list-threshold)# object 1`
	`RP/0/RP0/CPU0:router(config-track-list-threshold)# object 2`
	`RP/0/RP0/CPU0:router(config-track-list-threshold)# object 3`
	`RP/0/RP0/CPU0:router(config-track-list-threshold)# object 4`
		トラック タイプ track1 のメンバーに object 1、object 2、object 3 および object 4 を設定します。

ステップ 5	**threshold percentage up** *percentage* **down** *percentage* 例： `RP/0/RP0/CPU0:router(config-track-list-threshold)# threshold percentage up 50 down 33`	リストがそれぞれアップ状態またはダウン状態であると見なされるために、アップ状態またはダウン状態である必要があるオブジェクトの割合を設定します。 たとえば、object 1、object 2、および object 3 がアップ状態にあり、object 4 がダウン状態にある場合、リストはアップ状態にあると見なされます。

ステップ 6	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	例：
	`RP/0/RP0/CPU0:router(config-track)# end`
	または
	`RP/0/RP0/CPU0:router(config-track)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、次に示す変更のコミットを求めるプロンプトが表示されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		• **yes** と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		• **no** と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モー

	コマンドまたはアクション	目的
		ドに戻ります。変更はコミットされません。 • **cancel** と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。 • 実行コンフィギュレーション ファイルに設定変更を保存し、コンフィギュレーション セッションを継続するには、**commit** コマンドを使用します。

オブジェクト リストに基づくトラッキングの設定：しきい値の重み

グローバル コンフィギュレーション モードでこのタスクを実行し、しきい値の重みを使用してリストの状態を判断して、トラッキング対象オブジェクト リスト（ここではインターフェイスまたはプレフィックスのリスト）を作成します。

手順

	コマンドまたはアクション	目的
ステップ 1	**configure**	
ステップ 2	**track** *track-name* **例**： `RP/0/RP0/CPU0:router(config)# track track1`	トラック コンフィギュレーション モードを開始します。 • *track-name*：トラッキングの対象となるオブジェクト名を指定します。
ステップ 3	**type list threshold weight** **例**： `RP/0/RP0/CPU0:router(config-track-list)# type list threshold weight`	トラッキングのタイプにしきい値の重みリストを設定します。

ステップ 4	**object** *object-name* **weight** *weight*
	例：
	`RP/0/RP0/CPU0:router(config-track-list-threshold)# object 1 weight 10`
	`RP/0/RP0/CPU0:router(config-track-list-threshold)#`
		track t1 のメンバーに object 1、object 2 および object 3 を設定し、それぞれに重み 10、5 および 3 を設定します。

	コマンドまたはアクション	目的
	`object 2 weight 5` `RP/0/RP0/CPU0:router(config-track-list-threshold)# object 3 weight 3`	
ステップ 5	**threshold weight up** *weight* **down** *weight* **例**： `RP/0/RP0/CPU0:router(config-track-list-threshold)# threshold weight up 10 down 5`	リストがそれぞれアップ状態またはダウン状態であると見なされるために、アップ状態またはダウン状態である必要があるオブジェクトの重みの範囲を設定します。この例では、object 1 および 2 がアップ状態にあり、累積の重みは 15 である（10 ～ 5 の範囲内ではない）ため、リストはダウン状態と見なされます。

ステップ 6	次のいずれかのコマンドを使用します。
	• **end**
	• **commit**
	例：
	`RP/0/RP0/CPU0:router(config-track)# end`
	または
	`RP/0/RP0/CPU0:router(config-track)# commit`
		設定変更を保存します。
		• **end** コマンドを実行すると、次に示す変更のコミットを求めるプロンプトが表示されます。
		`Uncommitted changes found, commit them before exiting(yes/no/cancel)? [cancel]:`
		• **yes** と入力すると、実行コンフィギュレーション ファイルに変更が保存され、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。
		• **no** と入力すると、コンフィギュレーション セッションが終了して、ルータが EXEC モードに戻ります。変更はコミットされません。
		• **cancel** と入力すると、現在のコンフィギュレーションセッションが継続します。コンフィギュレーション セッションは終了せず、設定変更もコミットされません。
		• 実行コンフィギュレーション ファイルに設定変更を保存し、コンフィギュレーション セッションを継続

	コマンドまたはアクション	目的
		するには、**commit** コマンドを使用します。

IPSLA の到達可能性のトラッキング

IP サービス レベル契約（SLA）動作の戻りコードのトラッキングをイネーブルにするには、このタスクを使用します。

手順

	コマンドまたはアクション	目的
ステップ 1	**configure** **例：** `RP/0/RP0/CPU0:router# configure`	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。
ステップ 2	**track** *track-name* **例：** `RP/0/RP0/CPU0:router(config)# track t1`	トラック コンフィギュレーション モードを開始します。
ステップ 3	**type rtr** *ipsla-no* **reachability** **例：** `RP/0/RP0/CPU0:router(config-track)# type rtr 100 reachability`	到達可能性をトラッキングする IP SLA 動作 ID を指定します。*ipsla-no* の有効値は、1 ～ 2048 の範囲です。
ステップ 4	**commit**	

IPSLA トラッキングの設定：例

次に、IPSLA のトラッキング設定の例を示します。

```
RP/0/RP0/CPU0:router(config)# track track1
RP/0/RP0/CPU0:router(config-track)# type rtr 1 reachability
RP/0/RP0/CPU0:router(config-track)# delay up 5
RP/0/RP0/CPU0:router(config-track)# delay down 10
```

オブジェクト トラッキングの設定例

インターフェイスがアップ状態かダウン状態かのトラッキング：実行コンフィギュレーションの例

```
track connection100
  type list boolean and
    object object3 not
    delay up 10
    !
interface service-ipsec 23
  line-protocol track connection100
  !
```

インターフェイスのライン プロトコル ステートのトラッキング：実行コンフィギュレーションの例

この例では、トラフィックはインターフェイス service-ipsec1 から到着し、インターフェイス TenGigE0/11/0/3 を経由して終了します。

```
track IPSec1
  type line-protocol state
    interface TenGigE0/11/0/3
    !
interface service-ipsec 1
  ipv4 address 70.0.0.1 255.255.255.0
  profile vrf1_profile_ipsec
  line-protocol track IPSec1
  tunnel source 80.0.0.1
  tunnel destination 80.0.0.2
  service-location preferred-active 0/0/1
  !
```

次に、前述の例を実行した後の **show track** コマンドの出力例を示します。

```
RP/0/RP0/CPU0:router# show run track

  Track IPSec1
  Interface GigabitEthernet0_0_0_3 line-protocol
  !
    Line protocol is UP
    1 change, last change 10:37:32 UTC Thu Sep 20 2007
    Tracked by:
    service-ipsec1
    !
```

IP ルートの到達可能性のトラッキング：実行コンフィギュレーションの例

この例では、インターフェイス service-ipsec1 から到着したトラフィックの宛先がネットワーク 7.0.0.0/24 にあります。このトラッキング手順は、ルーティング プロトコル プレフィックスの状態に従い、ルーティング テーブルに変更があったときに信号を送ります。

```
track PREFIX1
  type route reachability
    route ipv4 7.0.0.0/24
    !
  interface service-ipsec 1
  vrf 1
  ipv4 address 70.0.0.2 255.255.255.0
  profile vrf_1_ipsec
  line-protocol track PREFIX1
  tunnel source 80.0.0.2
  tunnel destination 80.0.0.1
  service-location preferred-active 0/2/0
```

オブジェクトのリストに基づいたトラックの構築：実行コンフィギュレーションの例

この例では、インターフェイス service-ipsec1 から到着するトラフィックが、インターフェイス TenGigE0/11/0/3 およびインターフェイス ATM 0/2/0/0.1 を介して終了します。トラフィックの宛先はネットワーク 7.0.0.0/24 です。

いずれかのインターフェイスまたはリモート ネットワークがダウンした場合は、トラフィック フローが停止される必要があります。これを行うには、ブール AND 式を使用します。

```
track C1
 type route reachability
  route ipv4 3.3.3.3/32
 !
!
track C2
 type route reachability
  route ipv4 1.2.3.4/32
 !
!
track C3
 type route reachability
  route ipv4 10.0.20.2/32
 !
!
track C4
 type route reachability
  route ipv4 10.0.20.0/24
 !
!
track OBJ
 type list boolean and
  object C1
  object C2
 !
```

```
!
track OBJ2
 type list boolean or
  object C1
  object C2
 !
```

IPSLA ベースのオブジェクト トラッキングの設定：コンフィギュレーションの例

次に、ACL と IPSLA 設定を含む IPSLA ベースのオブジェクト トラッキングの設定例を示します。

ACL の設定：

```
RP/0/RP0/CPU0:router(config)# ipv4 access-list abf-track
RP/0/RP0/CPU0:router(config-ipv4-acl)# 10 permit any nexthop track track1 1.2.3.4
```

オブジェクト トラッキングの設定：

```
RP/0/RP0/CPU0:router(config)# track track1
RP/0/RP0/CPU0:router(config-track)# type rtr 1 reachability
RP/0/RP0/CPU0:router(config-track)# delay up 5
RP/0/RP0/CPU0:router(config-track)# delay down 10
```

IPSLA の設定：

```
RP/0/RP0/CPU0:router(config)# ipsla
RP/0/RP0/CPU0:router(config-ipsla)# operation 1
RP/0/RP0/CPU0:router(config-ipsla-op)# type icmp echo
RP/0/RP0/CPU0:router(config-ipsla-icmp-echo)# source address 2.3.4.5
RP/0/RP0/CPU0:router(config-ipsla-icmp-echo)# destination address 1.2.3.4
RP/0/RP0/CPU0:router(config-ipsla-icmp-echo)# frequency 60
RP/0/RP0/CPU0:router(config-ipsla-icmp-echo)# exit
RP/0/RP0/CPU0:router(config-ipsla-op)# exit
RP/0/RP0/CPU0:router(config-ipsla)# schedule operation 1
RP/0/RP0/CPU0:router(config-ipsla-sched)# start-time now
RP/0/RP0/CPU0:router(config-ipsla-sched)# life forever
```