Cisco Meeting Server

Cisco Meeting Server リリース 3.7

イベントガイド

2023 年 3 月 16 日

目次

変更履歴

日付	変更点
2023 年 3 月 16 日	3.7 の新しいバージョン。
2022 年 8 月 23 日	3.6 に対応した新バージョン。
2022 年 4 月 20 日	3.5 に対応した新バージョン。
2021 年 8 月 24 日	3.3 に対応した新バージョン。
2021 年 4 月 9 日	3.2 に対応した新バージョン。

1 はじめに

Meeting Server は、Meeting Server 上で発生した変更をリアルタイムで「イベントクライアント」に通知できます。Meeting Server はイベントのサーバとして機能し、イベント クライアントは Web ベースの管理アプリケーションなどになります。Cisco Meeting Management は、イベントクライアントとして機能します。

注：ユーザーは、API クライアントの構築に似た方法で、独自のイベントクライアントを構築できます。イベント クライアントは、HTTP および WebSocket ライブラリをサポートする必要があります。これらは、Python のような一般的なスクリプト言語で使用できます。
Meeting Server のイベント ポートは、Web 管理用に設定したのと同じポートです。これは通常、インターフェイス A の TCP ポート 443 になります。

Meeting Server の API リソースを継続的にポーリングするのではなく、イベント クライアントは、イベント リソースにサブスクライブして更新を受信します。たとえば、イベントクライアントと Meeting Server の間の WebSocket 接続を確立した後に、イベントクライアントはイベントリソース **`callRoster`**に登録し、アクティブな会議の参加者リストの最新情報を受け取り、新しい参加者が参加したり、既存の参加者がレイアウトを変更したりするのを確認できます。

Meeting Server では、登録可能な次の 3 つのイベントリソースをサポートしています。

- **`callInfo`** は、特定の会議に関する情報を提供します。
- **`callRoster`** は、電話会議中の各参加者に関する情報を提供します。
- **`calls`** は、アクティブな会議に関する情報を提供します。

サブスクリプション可能な各イベントリソースにはサブスクリプション可能な要素があります（セクション 2 を参照）。

1.1 サブスクリプションの概要

1.1.1 サブスクリプションの概要

イベントクライアントが Meeting Server へのサブスクリプションを作成すると、サブスクリプションには、イベントクライアントが登録するリソースのセットが一覧になります。これは、アクティブなサブスクリプションを希望するリソースの完全なリストである必要があります。そのた

め、イベントクライアントが既存のサブスクリプションセットに新しいリソースを追加する必要がある場合は、既存の登録先のすべてのリソースを新しいリクエストに含める必要があります。すべてのアクティブなサブスクリプションを停止するには、イベントクライアントによって空のリクエストを提供する必要があります（これは、WebSocket 接続が切断された際にも発生します）。

サブスクリプション リクエスト内で登録される各リソースには、イベントクライアントによって一意の番号が割り当てられるため、Meeting Server が更新を提供する際に、クライアントではどのサブスクリプションが参照されているかを認識します。イベントクライアントが以前と同じリソースに登録しているが、数値識別子が異なる場合、これは Meeting Server によって新しいサブスクリプションとして扱われ、古いサブスクリプション リクエストは事実上破棄されます。

Meeting Server が新しいサブスクリプション リクエストを受信すると、最初に、そのサブスクリプション リクエストを受信して処理中であることを示す単純な「ack」で応答します。この段階での肯定確認応答は、Meeting Server がリクエストを受信したことを意味するだけです。サブスクリプション リクエストの各要素の更新ステータスを与えるフォローアップがあり、その後、そのリソースの更新が実際に行われます。

たとえば、イベントクライアントがアクティブな会議に関する情報と特定の会議の参加者リストの登録を要求した場合、最初に「ack」を返し、次に両方のサブスクリプション「保留中」であることを伝える「subscriptionUpdate」メッセージを返します。これは Meeting Server がまだサブスクリプションを設定中であることを意味します。しばらくして、イベントクライアントでは、アクティブな会議のリストへのサブスクリプションが「アクティブ」であり、参加者リストのサブスクリプションがまだ「保留中」であることを示す更新を受け取る場合があります。Meeting Server は、この段階でアクティブな会議リストの更新の提供も開始します。登録者にサブスクリプションが「アクティブ」であることが通知されるまで、登録先のリソースに関連する実際の更新は発生しません。イベントクライアントが登録した参加者リストを持つ会議が存在する場合（つまり、提供された GUID がまだアクティブな会議に対応している場合）、イベントクライアントは、アクティブな会議サブスクリプションと参加者リストのサブスクリプションの両方が「アクティブ」であることを示すサブスクリプションの更新を受け取り、またこの時点で参加者リストの更新の受信も開始します。参加者リストのサブスクリプションの会議 GUID がアクティブな会議に正常に解決されなかった場合、サブスクリプションの状態は「非アクティブ化」になります。これは、会議が終了したとき、またはイベントクライアントが登録解除したときに、後続の「subscriptionUpdate」メッセージで登録が変更される状態でもあります。

イベントリソースを登録する際のメッセージフローの詳細については、セクション 5 を参照してください。

2 イベントリソースとサブスクリプション可能な要素

2.0.1 イベントリソースとサブスクリプション可能な要素

バージョン 2.4 以降、Meeting Server はイベントを使用して、次のイベントリソースで「イベントクライアント」にリアルタイム情報を提供することをサポートしています。

- **callInfo** は、特定の会議に関する情報を提供します。
- **callRoster** は、電話会議中の各参加者に関する情報を提供します。
- **calls** は、アクティブな会議に関する情報を提供します。

2.1 callInfo

表 1: 登録可能なイベントリソースを使用して利用可能な特定の会議（コール）に関する情報

callInfo

名前	値	説明（Description）
リクエストパラメータ		
call	ID	更新された情報を受け取る会議の ID です。
登録可能な要素		
name	文字列	会議の名前です。
participants	数値	現在の会議の参加者数です。
distributedInstances	数値	Call Bridge クラスタ全体に存在するこの会議の分散インスタンスの数（クラスタ化されていない会議の場合は 0）。
recording	active| inactive	次のいずれか： *active* - この会議は現在録音されています *inactive* - この会議は現在録音されていません。
endpointRecording	active| inactive	次のいずれか： *active* - この会議は現在、エンドポイント（Lync クライアント）によって外部で録音されています。 *inactive* - この会議は現在、エンドポイント（Lync クライアント）よって外部で録音されていません。

名前	値	説明（Description）
streaming	active| inactive	次のいずれか： *active* - この会議は現在ストリーミング中です *inactive* - この会議は現在ストリーミングされていません。
lockState	locked| unlocked	次のいずれか： *locked* - この会議は現在ロックされています *unlocked* - この会議は現在ロック解除されています。
callType	coSpace| adHoc| lyncConferencing| 転送	次のいずれか： *coSpace* - このコールは coSpace のインスタンス化です *adHoc* - これはアドホック マルチパーティ コールです *lyncConferencing* - このコールは、Lync がホストする会議への Meeting Server 接続です。 *forwarding* - これは転送または「ゲートウェイ」コールです。
callCorrelator	ID	この値は、複数の Call Bridge に分散している可能性がありますが、同じ coSpace またはアドホックコールのいずれかにあるコールレッグを識別するために使用できます。 注：coSpace 内のコールの場合、callCorrelator 値は coSpace の有効期間中は同じになります。アドホックコールごとに、値が動的に生成されます。
joinAudioMuteOverride	true| false	次のいずれか： *true* - 会議に参加すると、新しい参加者はミュートされます *false* - 会議に参加する際に、新しい参加者はミュートされません。設定されていない場合は、これがデフォルト設定です。

2.2 callRoster

表 2：登録可能なイベントリソース `callRoster` で利用可能な電話会議中（コール）の各参加者に関する情報

名前	値	説明（Description）
リクエストパラメータ		
call	ID	参加者の更新を受信する会議の ID です。
登録可能な要素		
name	文字列	参加者の表示名です。
uri	URI ユーザ パート	この参加者に関連付けられた URI です。
state	initial| ringing| connected| onHold	この参加者の現在のシグナリング状態を反映します。
direction	incoming| outgoing	次のいずれか： *incoming* – この参加者は、会議にダイヤルしました（リモート SIP デバイスが Meeting Server への接続を開始しました） *outgoing* – この参加者は、会議に参加するためにダイヤルアウトされました（コールレッグは、Meeting Server からリモート SIP デバイスに確立しました）。
audioMuted	true| false	次のいずれか： *true* – Meeting Server がこの参加者の音声をミュートしました *false* – Meeting Server はこの参加者の音声をミュートしていません。
videoMuted	true| false	次のいずれか： *true* – Meeting Server がこの参加者のビデオをミュートしました *false* – Meeting Server はこの参加者のビデオをミュートしていません。
importance	数値	この参加者の重要度の値です。重要度が設定されていない場合、または設定解除に変更された場合は NULL になります。

名前	値	説明（Description）
layout	allEqual| speakerOnly| telepresence| stacked| allEqualQuarters| allEqualNinths| allEqualSixteenths| allEqualTwentyFifths | onePlusFive| onePlusSeven| onePlusNine| automatic| onePlusN	この参加者が現在使用しているレイアウトです。
activeSpeaker	true| false	次のいずれか： *true* - この参加者は現在、この会議のアクティブなスピーカーと見なされます *false* - この参加者は現在、この会議のアクティブなスピーカーとは見なされていません
presenter	true| false	次のいずれか： *true* - この参加者は現在、この会議でプレゼンテーションを行っています（画面を共有しています） *false* - この参加者は現在、この会議でプレゼンテーションを行っていません。
endpointRecording	active| inactive	次のいずれか： *active* - この参加者は現在、会議を録音しています *inactive* - この参加者は現在、会議を録音していません。
canMove	true|false	この参加者が movedParticipant API コマンドを使用して移動できるかどうかを示します。（バージョン 2.6 から）
canMoveToLobby	true|false	この参加者をロビーに移動できるかどうかを示します。（バージョン 3.5 より）
movedParticipant	ID	この参加者が参加者の移動の一環として作成された場合、ID は、この参加者が移動した参加者の GUID になります。（バージョン 2.6 から）
movedParticipantCallBridge	ID	この参加者が参加者の移動の一環として作成された場合、ID は、この参加者が移動した conference をホストしている Call Bridge の GUID です。（バージョン 2.6 から）

2.3 calls

表 3：登録可能なイベントリソース calls で利用可能な会議（コール）に関する情報

名前	値	説明（Description）
必須の応答要素		
call	ID	要素が更新された会議の ID です。
登録可能な要素		
name	文字列	会議の名前です。
participants	数値	現在の会議の参加者数です。
distributedInstances	数値	Call Bridge クラスタ全体に存在するこの会議の分散インスタンスの数（クラスタ化されていない会議の場合は 0）。
recording	active| inactive	次のいずれか： active - この会議は現在録音されています inactive - この会議は現在録音されていません。
endpointRecording	active| inactive	次のいずれか： *active* - この会議は現在、エンドポイント（Lync クライアント）によって外部で録音されています。 *inactive* - この会議は現在、エンドポイント（Lync クライアント）よって外部で録音されていません。
streaming	active| inactive	次のいずれか： *active* - この会議は現在ストリーミング中です *inactive* - この会議は現在ストリーミングされていません。
lockState	locked| notLocked	次のいずれか： *locked* - この会議は現在ロックされています *notLocked* - この会議は現在ロックされていません。
callType	coSpace| forwarding| adHoc| lyncConferencing	次のいずれか： *coSpace* - この会議は coSpace のインスタンス化です。 *forwarding* - これは転送または「ゲートウェイ」コールです *adHoc* - これはアドホック マルチパーティ コールです *lyncConferencing* - このコールレッグは Lync 会議に参加しています。
callCorrelator	ID	コールのすべての分散インスタンスで同じ相関関係がある GUID です。

3 Call Bridge グループとクラスタ

3.0.1 Call Bridge グループとクラスタ

会議が Call Bridge グループ（またはクラスタ）全体でホストされている場合、グループ内の Call Bridge 間でメッセージが渡され、各 Meeting Server がアクティブな会議と、1 人または複数の参加者がいる会議の名簿リスト情報を確実に認識できるようにします。

サブスクリプションによって会議へのローカルおよびリモートの参加者の更新を受信しますが、リモートの参加者が会議から退席した場合、パラメータ「reason」は提供されません。会議をホストしている Call Bridge へのサブスクリプションのみが、参加者が会議から退席した理由を受信できます。Call Bridge グループまたはクラスタ全体のすべてのアクティブな会議に関する情報を受信するには、イベントクライアントはすべての Meeting Server のイベントリソース **`calls`** に登録する必要がありますが、イベントリソース **`callInfo`** または **`callRoster`** を表示するには、イベントクライアントは会議をレポートする Meeting Server の 1 つを登録するだけで済みます。

Call Bridge でホストされている参加者が会議から退席したために Meeting Server が会議に参加しなくなった場合、Call Bridge はサブスクリプションを非アクティブ化し、イベントクライアントは別の Meeting Server を選択する必要があります。

4 認証フロー例

4.0.1 認証フロー例

Meeting Server でイベントの登録が許可される前に、イベントクライアントは Meeting Server によって認証され、イベントクライアントと Meeting Server の間で WebSocket プロトコルを使用した接続が確立される必要があります。この接続により、イベントクライアントは、Meeting Server の API を常時ポーリングする必要なく、イベント情報を受信できます。

図 1 は、WebSocket を確立するために必要なイベントクライアントと Meeting Server 間のコールフローを示しています。HTTP POST および HTTP GET を Meeting Server に送信するためには、イベントクライアントに Python などのコーディング言語が必要になります。

図 1：イベントクライアントと Meeting Server 間で WebSocket を確立するコールフローの概要

1. イベントクライアントは、新しい認証トークンをプロビジョニングするために /api/v1/authTokens に POST します。たとえば、リクエストは次のようになります。

```
POST /api/v1/authTokens HTTP/1.1\r\n
Origin: http://xx.xxx.xxx.xxx:8080\r\n
Content-length: 0\r\n
Host: xx.xxx.xxx.xxx:8080\r\n
Accept: */*\r\n
Connection: keep-alive\r\n
Authorization: Basic Ym9iOmJ1aWxkZXI=\r\n
\r\n
```

2. 「承認」が十分な権限を持つユーザーアカウントに関連していると仮定すると、Meeting Server からの正常な応答は次のような形式になります。

```
HTTP/1.1 200 OK\r\n
X-Cisco-CMS-Auth-Token: 7174c102-61b3-47a6-8ff2-b86256cca958\r\n
Connection: close\r\n
\r\n
```

返された「X-Cisco-CMS-Auth-Token」は、次の段階である WebSocket の接続そのもので使用されます。

3. クライアントは、WebSocket 接続を設定するために、次のような形式で別の HTTP リクエストを行います。

```
GET /events/v1?authToken=7174c102-61b3-47a6-8ff2-b86256cca958 HTTP/1.1\r\n/
Host: xx.xxx.xxx.xxx\r\n/
Connection: Upgrade\r\n/
Pragma: no-cache\r\n/
Cache-Control: no-cache\r\n/
Upgrade: websocket\r\n/
Origin: http://xx.xxx.xxx.xxx:8080\r\n/
Sec-WebSocket-Version: 13\r\n/
Accept-Encoding: gzip, deflate\r\n
Accept-Language: en-GB,en-US;q=0.8,en;q=0.6\r\n
Sec-WebSocket-Key: lGaahHe/KdA9lPdPxAIZfw==\r\n
Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits\r\n
\r\n
```

注：X-Cisco-CMS-Auth-Token の値は、「authToken」URI パラメータとして Meeting Server に送信する必要があります。Sec-WebSocket-Key の値（およびフォローアップ Sec-WebSocket-Accept）の値は、RFC6455 に準拠しています。

---

4. Meeting Server が WebSocket 接続を受け入れると仮定すると（たとえば、authToken が有効な場合）、成功時の応答は次のようになります。

```
HTTP/1.1 101 Switching Protocols\r\n
Upgrade: websocket\r\n
Connection: upgrade\r\n
Sec-WebSocket-Accept: ZISmDfOsp675RM7TQKa0LbQKCqk=\r\n
\r\n
```

5. 双方向の WebSocket 接続が「準備完了」となり、イベントメッセージのフローを開始できます。

5 メッセージフローの例

5.0.1 メッセージフローの例

イベントクライアントと Meeting Server 間の WebSocket が確立されると、イベントクライアントは Meeting Server にイベントタイプの更新を登録できるようになります。

図 2 は、Web ソケット接続を介したイベントクライアントと Meeting Server 間のコールフローを示しています。イベントクライアントからのサブスクリプションは、JSON ファイルの形式である必要があります。コールフローの詳細については、図 2 に続く説明と例を参照してください。

図 2：イベントクライアントと Meeting Server 間をフローするイベントメッセージの概要

1. イベントクライアントから Meeting Server への最初のサブスクリプションで、コールリスト（アクティブな会議リスト）などのイベントタイプへの登録をリクエストします。

```
{
     "type": "message",
     "message": {
          "messageId": 8,
          "type": "subscribeRequest",
          "subscriptions": [
               {
                    "index": 3,
                    "type": "calls",
                    "elements": [
                    "name",
                    "participants"
                    ]
               }
          ]
     }
}
```

上記の例では、クライアントはインデックス 8 のメッセージとして登録リクエストを送信し、「subscriptions」と呼ばれるセット（配列）に登録する単一のリソースを提供しています。このサブスクリプションにはインデックス（タグ）「3」を使用しており、そのタイプは「calls」であり、アクティブコールリストを参照しています。このサブスクリプションの中では「name」（会議名）と「participants」（アクティブな参加者数）という関心のある要素のセットが指定されています。この要素のセットには、主に 2 つの意味があります。

- どの変更が Meeting Server → クライアントの更新をトリガーするかを決定します（この場合、名前または参加者数が変更された場合、更新が送信されます）。
- Meeting Server → クライアントの更新に含まれる要素を決定します（この場合、クライアントに送信されるレコードには、名前と参加者数のみが含まれます）。

2. Meeting Server は、登録リクエストに対して ACK で応答します。

```
{
     "type": "messageAck",
     "messageAck": {
          "messageId": 8,
          "status": "success"
     }
}
```

この ACK には、どのメッセージが確認応答されているかをクライアントに示すための 「messageId」の 8 と、メッセージが正常に解釈され、処理されたかどうかを示すための「status」コードがあります。

3. また、Meeting Server は、アクティブなサブスクリプションの状態に関して、リクエスト元のクライアントを最新の状態に保ちます。たとえば、これは次のようにクライアントに送信される subscriptionUpdate の形式になっています。

```
{
     "type": "message",
     "message":
          "messageId": 1,
          "type": "subscriptionUpdate",
          "subscriptions": [
               {
                    "index": 3,
                    "state": "pending"
               }
          ]
     }
}
```

これは、インデックス「3」のサブスクリプションが保留中であることをクライアントに通知します。設定中ですが、まだアクティブではありません。Meeting Server はこのメッセージに messageId が 1 のタグを付けています。クライアントでは、Meeting Server が後で更新を送信できるようにこのメッセージに ACK を送信する必要があります。

4. これに対応するクライアントから Meeting Server への messageAck は次のようになります。

```
{
     "type": "messageAck",
     "messageAck": {
          "messageId": 1,
          "status": "success"
     }
}
```

Meeting Server とクライアントは、messageId 値にそれぞれ個別の番号領域を使用しますが、これらは区別される必要はなく、連続的である必要もありません。

5. Meeting Server がリクエストされたサブスクリプションを正常に設定すると、さらに次のような subscriptionUpdate を送信する場合があります。

```
{
     "type": "message",
     "message": {
          "messageId": 2,
          "type": "subscriptionUpdate",
          "subscriptions": [
               {
                    "index": 3,
                    "state": "active"
               }
          ]
     }
}
```

これにより、サブスクリプション（インデックス 3）が現在アクティブであることをクライアントに通知します。クライアントでは、そのサブスクリプションに固有の更新を受信できるようになります。以前と同じように、Meeting Server からのこのメッセージは、クライアントによって承認される必要があります。

6. クライアントでは、messageId が 2 のサブスクリプションの更新を承認します。クライアントは、subscriptionUpdate で Meeting Server によって使用される値として、ack の messageId 2 を使用します。

```
{
     "type": "messageAck",
          "messageAck":
          { "messageId": 2,
          "status": "success"
     }
}
```

7. クライアントによってコールリストに登録した後のある時点で、Meeting Server は、開始直後の会議に関する情報を送信します。

```
{
     "type": "message",
     "message": {
          "messageId": 3,
          "type": "callListUpdate",
          "subscriptionIndex": 3,
          "updates": [
               {
                    "call": "97c771ae-fc2e-4257-b129-30ee818e034b",
                    "updateType": "add",
                    "name": "Andy's coSpace",
                    "participants": 0
               }
          ]
```

```
    }
}
```

以前と同じように、このメッセージ messageId 3 は、クライアントによって承認される必要があります。

注：重複を避けるために、この例では messageAcks は表示されなくなりました。

この更新には subscriptionIndex 「3」のタグが付けられ、更新によって参照するサブスクリプションをクライアントに示します。更新には、データの解析を支援する「type」が含まれています。「updates」配列には、Meeting Server が提供する新しい情報が含まれています。この例では、「updateType」は「add」です。これは、この会議の最初の通知であることを意味し、「call」フィールドにその会議の GUID が格納されています。「call」および「updateType」の値はすべての更新に表示されますが、残りのフィールドは、クライアントからのサブスクリプション リクエストで提供される「elements」の値で決まります。登録メッセージに「elements」ノードが指定されていない場合（または空の場合）、追加のフィールドは含まれません。ただし、例では「name」と「participants」が含まれていたため、更新にはそれらが含まれます。

8. この会議の最初の「add」メッセージの後、参加者が会議に参加すると、Meeting Server からさらに更新を受信します。

```
{
    "type": "message",
    "message": {
        "messageId": 4,
        "type": "callListUpdate",
        "subscriptionIndex": 3,
        "updates": [
            {
                "call": "97c771ae-fc2e-4257-b129-30ee818e034b",
                "updateType": "update",
                "participants": 1
            }
        ]
    }
}
```

参加者数は現在「1」、updateType が「update」であり、これはそのコール（97c771ae-fc2e-4257-b129-30ee818e034b）に対する最初のメッセージではなく、以前の通知の更新であることを示しています。このコールの「name」の値は変更されていないため、この更新には含まれません。

9. クライアント アプリケーションが（イベントメカニズムまたは API クエリまたは 「callStart」CDR 経由）関心のあるアクティブなコールの存在を把握すると、クライアント アプリケーションではそのコールに固有のリソースを登録できます。たとえば、次のような新しいメッセージを使用して、Meeting Server とのサブスクリプションを再構成できます。

```
{
     "type": "message",
     "message": {
          "messageId": 9,
          "type": "subscribeRequest",
          "subscriptions": [
               {
                    "index": 1,
                    "type": "callRoster",
                    "call": "97c771ae-fc2e-4257-b129-30ee818e034b",
                    "elements": [
                         "name",
                         "uri",
                         "state",
                         "importance"
                    ]
               },
               {
                    "index": 2,
                    "type": "callInfo",
                    "call": "97c771ae-fc2e-4257-b129-30ee818e034b",
                    "elements": [
                         "name",
                         "participants",
                         "streaming"
                    ]
               },
               {
                    "index": 3,
                    "type": "calls",
                    "elements": [
                         "name",
                         "participants"
                    ]
               }
          ]
     }
}
```

クライアントは、その「subscriptions」のセットに同じ「index」3 を持つサブスクリプションを保持することにより、「calls」（アクティブな会議リスト）に登録されたままになります。ただし、現在、「callRoster」および「callInfo」へのサブスクリプションがセットに追加されています。これらのサブスクリプションでは、特定の「call」GUID を指定する必要があります。この場合は「97c771ae-fc2e-4257-b129-30ee818e034b となります。これはクライアントに Meeting Server からの以前の "callListUpdate" メッセージで通知されたものです。

6 WebSocket の仕様

このセクションでは、WebSocket を使用してイベントクライアントと Meeting Server 間の接続を確立する方法について詳しく説明します。以下に指定されている制限を超えないようにしてください。

- Meeting Server あたりの WebSocket 同時接続の最大数：5
- WebSocket 接続あたりの同時サブスクリプションの最大数：100
- フラグメント化した WebSocket フレームはサポートされていません。
- ping または pong コントロールフレーム、バイナリデータフレームはサポートされていません。

Cisco の法的情報

このマニュアルに記載されている仕様および製品に関する情報は、予告なしに変更されることがあります。このマニュアルに記載されている表現、情報、および推奨事項は、すべて正確であると考えていますが、明示的であれ黙示的であれ、一切の保証の責任を負わないものとします。このマニュアルに記載されている製品の使用は、すべてユーザー側の責任となります。

対象製品のソフトウェア ライセンスと限定保証は、製品に添付された『Information Packet』に記載されています。添付されていない場合には、代理店にご連絡ください。

Cisco が採用している TCP ヘッダー圧縮機能は、UNIX オペレーティング システムの UCB（University of California, Berkeley）のパブリック ドメイン バージョンとして、UCB が開発したプログラムを採用したものです。All rights reserved. Copyright © 1981, Regents of the University of California.
ここに記載されている他のいかなる保証にもよらず、各社のすべてのマニュアルおよびソフトウェアは、障害も含めて「現状のまま」として提供されます。シスコおよび上記代理店は、商品性、特定目的適合、および非侵害の保証、もしくは取り引き、使用、または商慣行から発生する保証を含み、これらに限定することなく、明示または暗黙のすべての保証を放棄します。

いかなる場合においても、Cisco およびその供給者は、このマニュアルの使用または使用できないことによって発生する利益の損失やデータの損傷をはじめとする、間接的、派生的、偶発的、あるいは特殊な損害について、あらゆる可能性がCisco またはその供給者に知らされていても、それらに対する責任を一切負わないものとします。

このマニュアルで使用している Internet Protocol（IP）アドレスおよび電話番号は、実際のアドレスおよび電話番号を示すものではありません。マニュアルの中の例、コマンド出力、ネットワーク トポロジー図、およびその他の図は、説明のみを目的として使用されています。説明の中に実際の IP アドレスおよび電話番号が使用されていたとしても、それは意図的なものではなく、偶然の一致によるものです。

この文書の印刷されたハード コピーおよび複製されたソフト コピーは、すべて管理対象外と見なされます。最新版については、現在のオンライン バージョンを参照してください。

Cisco は世界各国 200 箇所にオフィスを開設しています。各オフィスの住所と電話番号は、当社の Web サイト www.cisco.com/go/offices をご覧ください。

Cisco の商標または登録商標

Cisco および Cisco ロゴは、Cisco Systems, Inc. またはその関連会社の米国およびその他の国における登録商標または商標です。Cisco の商標の一覧については、https://www.cisco.com/c/ja_jp/about/legal/trademarks.html をご覧ください。本書に記載されているサードパーティの商標は、それぞれの所有者の財産です。「パートナー」という用語の使用は Cisco と他社との間のパートナーシップ関係を意味するものではありません。（1721R）