CHAPTER 1

概要

この章では、Cisco Unified Communications ソリューションの Java Telephony Application Programming Interface（JTAPI）アプリケーションを作成する前に理解しておく必要のある主な概念について説明します。次のような構成になっています。

- 「JTAPI の概要」（P.1-1）
- 「Cisco Unified JTAPI の概念」（P.1-4）
- 「コールバックのスレッド化」（P.1-11）
- 「アラーム サービス」（P.1-12）
- 「ソフトウェア要件」（P.1-13）

Cisco Unified Communications Manager の機能については、第 3 章「Cisco Unified JTAPI でサポートされる機能」を参照してください。詳細についてと、CTI のデバイスおよびサポートされる機能については、付録 E「CTI でサポートされるデバイス」と付録 D「リリースごとの Cisco Unified JTAPI オペレーション」を参照してください。

JTAPI の概要

Cisco Unified JTAPI は、Java ベースのコンピュータ テレフォニー アプリケーションとともに使用する目的で、Sun Microsystems によって開発された標準のプログラミング インターフェイスとして機能します。Cisco JTAPI では、Sun JTAPI 1.2 仕様が Cisco の追加の機能拡張として実装されています。Cisco JTAPI を使用して、次のアプリケーションを開発できます。

- Cisco Unified Communications Manager フォンの監視と制御。
- Computer–Telephony Integration（CTI）ポートとルート ポイント（仮想デバイス）を使用したコールのルーティング。

サポートされる基本的なテレフォニー API は、会議、転送、接続、応答、リダイレクト API で構成されます。

javax.telephony.* 階層にある JTAPI インターフェイスのパッケージは、テレフォニー リソースを Java アプリケーションで操作するためのプログラミング モデルを定義しています。インターフェイスの詳細については、付録 B「Cisco Unified JTAPI クラスおよびインターフェイス」を参照してください。

この章では、次のトピックについて取り上げます。

- 「Cisco Unified JTAPI と Contact Center」（P.1-2）
- 「Cisco Unified JTAPI とエンタープライズ」（P.1-2）
- 「Cisco Unified JTAPI アプリケーション」（P.1-3）
- 「Jtprefs アプリケーション」（P.1-3）

Cisco Unified JTAPI と Contact Center

Cisco Unified JTAPI は、Contact Center で使用し、正しい時間に正しい場所へコールを送信するためにデバイス ステータスの監視とルーティング指示の発行を行い、分析用にコールの統計を取得すると同時に録音指示の停止や開始を指示し、さらに、CRM アプリケーション、自動化スクリプト、およびリモート コール制御内で、コールを画面にポップアップさせます。

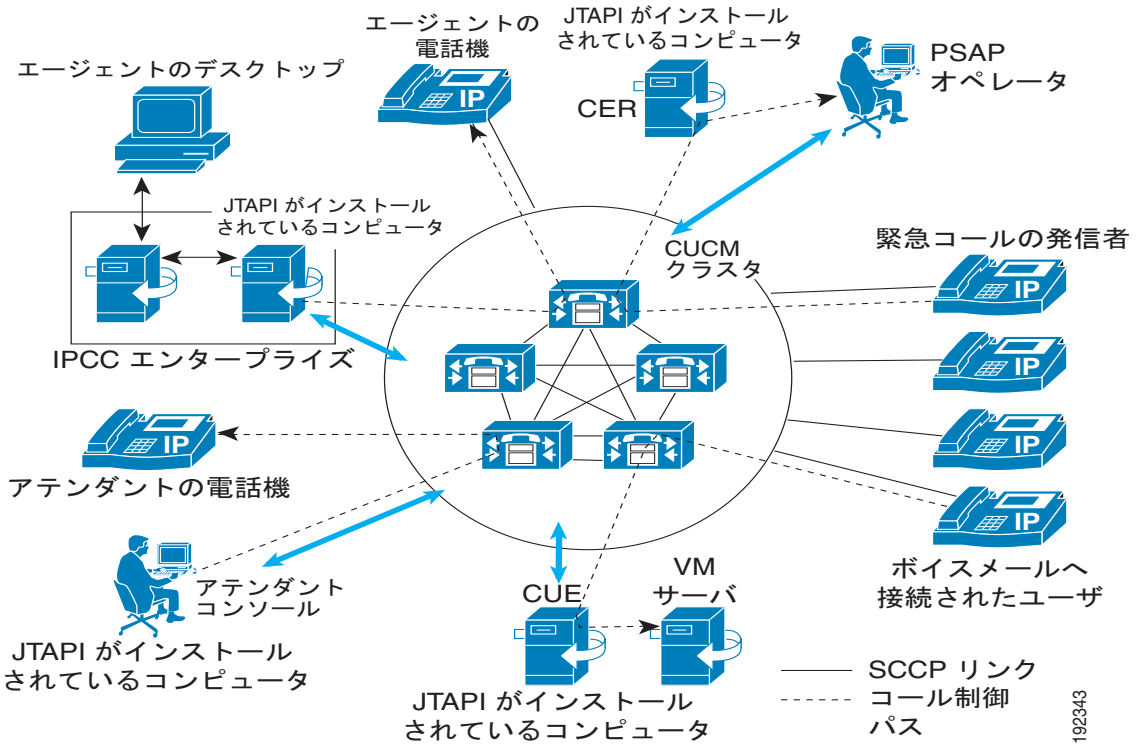
Cisco Unified JTAPI とエンタープライズ

Cisco Unified JTAPI はエンタープライズ環境で使用され、ユーザの応答可能、場所、プリファレンスを結合し、プレゼンス ベースのルーティングを行うための固有に調整された環境を実現します。たとえば、金融環境では、ブローカーや分析者が国際金融市場の急激な変化に対応できるように、市場データ、ビジネス ロジック、コール制御がブラウザ ベースのアプリケーションに組み込まれています。

健康管理環境では、コール制御、医者と患者の検索、救急隊の呼び出しが、ブラウザ ベースのコンソールに組み込まれています。さらに、サービス業環境では、発信者のデータが POS システムにリンクされ、部屋の予約またはレストランの予約、タクシーの手配、およびモーニング コールのスケジューリングなどの自動化が行われます。

図 1-1 は、エンタープライズ用に設定された典型的な Cisco Unified Communications Manager と Cisco Unified JTAPI を示しています。

図 1-1　Cisco Unified Communications Manager と Cisco Unified JTAPI



Cisco Unified JTAPI アプリケーション

Cisco Unified JTAPI アプリケーションのフローは次のようになります。

- JTAPIPeerFactory から JTAPIPeer オブジェクト インスタンスを取得する。
- JTAPIPeer で getProvider() API を使用し、プロバイダーを取得する。
- プロバイダーから、アプリケーションが使用する端末とアドレスを取得する。
- 関連するオブジェクトの機能を判別する。
- アプリケーションが監視および制御する必要がある、オブジェクトのオブザーバを追加する。
- アプリケーション フローを開始する（例：コールの開始）。

次の例に、基本的な JTAPI アプリケーションを示します。

```
   public void getProvider () {
          try {
                      JtapiPeer peer = JtapiPeerFactory.getJtapiPeer ( null );
                      System.out.println ("Got peer "+peer);
                      Provider provider =
peer.getProvider("cti-server;login=username;passwd=pass");
                      System.out.println ("Got provider "+provider);
                      MyProviderObserver providerObserver = new MyProviderObserver ();
                      provider.addObserver(providerObserver);
                      while (outOfService ) {
                             Thread.sleep(500);
                      }
                      System.out.println ("Provider is now in service");
                      Address[] addresses = provider.getAddresses();
                      System.out.println ("Found "+ addresses.length +" addresses");
                      for(int i=0; i< addresses.length; i++){
                             System.out.println(addresses[i]);
                      }
                      provider.shutdown();
          } catch (Exception e){
          }

   }
```

Jtprefs アプリケーション

Cisco Unified JTAPI の設定に必要なパラメータは、jtapi.ini ファイルにあります。Cisco Unified JTAPI は、このファイルを Java クラスパス内で検出します。パラメータは、Cisco Unified JTAPI でインストールされる Jtprefs アプリケーションを使用して変更できます。Jtprefs アプリケーションではこのアプリケーションが必要とするパラメータだけが設定されます。jtapi.ini に依存せずに、1 箇所からアプリケーションを管理できるため、大変有用です。

jtapi.ini ファイルにはデフォルト値が含まれていますが、クライアント アプリケーションでは jtapi.ini ファイルを特に修正することなくさまざまな値を修正できます。ただし、クライアント アプリケーションのさまざまなインスタンスでは、これらのパラメータに対して個別の設定を指定できます。CiscoJtapiProperties インターフェイスは、com.cisco.jtapi.extensions パッケージで定義されています。

アプリケーションでは、CiscoJtapiPeer から CiscoJtapiProperties オブジェクトを取得し、アクセス用メソッドと変更用メソッドを使用してパラメータに変更を加えます。これらのプロパティは、CiscoJtapiPeer から派生したすべてのプロバイダーに対して、CiscoJtapiPeer で最初の getProvider () が呼び出される前に、設定および適用する必要があります。

jtprefs.ini を起動できない非 GUI ベースのプラットフォームで実行されるアプリケーションでは、jtapi.ini ファイルを作成して jtapi.jar とともに配置できます。

詳細については、次のトピックを参照してください。

- 「jtapi.ini ファイルのフィールド」（P.4-21）
- 「デフォルト値を使用する場合の jtapi.ini ファイルの例」（P.4-27）

Cisco Unified JTAPI の概念

ここでは、次の概念について説明します。

- 「CiscoObjectContainer インターフェイス」（P.1-4）
- 「JtapiPeer と Provider」（P.1-4）
- 「Address と Terminal の関係」（P.1-6）
- 「Connection」（P.1-7）
- 「Terminal Connection」（P.1-8）
- 「端末とアドレスの制限」（P.1-8）
- 「CiscoConnectionID」（P.1-11）

CiscoObjectContainer インターフェイス

CiscoObjectContainer インターフェイスを使用すると、このインターフェイスの実装されたオブジェクトに、アプリケーションによって定義されたオブジェクトを関連付けることができます。Cisco Unified JTAPI では、次のインターフェイス上で CiscoObjectContainer インターフェイスが拡張されます。

- CiscoJTAPIPeer
- CiscoProvider
- CiscoCall
- CiscoAddress
- CiscoTerminal
- CiscoConnection
- CiscoTerminalConnection
- CiscoConnectionID
- CiscoCallID

JtapiPeer と Provider

JtapiPeer オブジェクトの実装によって作成される Provider オブジェクトは、アプリケーションと JTAPI 実装間の主な接点として機能します。Provider オブジェクトは、アプリケーションで常に制御可能な Address、Terminal、Call などのコール モデル オブジェクト全体の集合を格納しています。

JTAPI Preferences（JTPREFS）アプリケーションでは、サーバ名を返す JtapiPeer.getServices() を管理します。

Provider は 2 つの基本プロセス（初期化とシャットダウン）を伴います。

アプリケーションで CiscoProvider を取得するときには、JtapiPeer.getProvider() メソッドを使用して必ず次の情報を渡します。

- Cisco Unified Communications Manager サーバのホスト名または IP アドレス。
- ディレクトリで管理されるユーザのログイン。
- 指定したユーザのパスワード。
- （オプション）アプリケーション情報（このパラメータには任意の長さの文字列を指定できます）。

  appinfo がアラームにログされた場合に管理者がアラームの原因となったアプリケーションを認識できるように、アプリケーションには十分な情報を記述する必要があります。アプリケーションには、アプリケーションが常駐するホスト名または IP アドレス、およびアプリケーションが起動した時間を含めないでください。また、「=」または「;」の記号は、getProvider() 文字列の区切りに使用されるため、appinfo 文字列に入れることはできません。appinfo を指定しない場合、代わりに一般的な擬似固有名（JTAPI[XXXX]@hostname）を使用できます（XXXX は 4 桁の乱数を表します）。

パラメータは、次のように文字列として連結されたキー値のペアで渡します。

JtapiPeer.getProvider("CTIManagerHostname;login=user;passwd=userpassword;appinfo=Cisco Softphone")

初期化

JtapiPeer.getProvider() メソッドでは、TCP リンク、Cisco Unified Communications Manager との初期ハンドシェイク、およびデバイス リストの列挙が完了すると同時に Provider オブジェクトが返されます。このとき、プロバイダーは OUT_OF_SERVICE 状態にあります。Cisco Unified JTAPI アプリケーションは、制御対象のデバイス リストが有効になる前に、プロバイダーが IN_SERVICE 状態に移行するのを待つ必要があります。ProvInServiceEv イベントは、ProviderObserver インターフェイスの実装されたオブジェクトに通知されます。

（注） CiscoProviderObserver の実装だけでは不十分であり、provider.addObserver() を使用してプロバイダーにオブザーバを追加する必要があります。アプリケーションは、Provider がイン サービス状態にあることを示す通知を待つ必要があります。

JTAPI における QoS ベースライン化作業の一環として、ProviderOpenCompletedEv は「DSCP value for Applications」を JTAPI に提供します。JTAPI は CTI との接続に対してこの DSCP 値を設定し、Provider オブジェクトが存在する限り、CTI へのすべての JTAPI メッセージがこの DSCP 値を保持します。

シャットダウン

アプリケーションを使用して provider.shutdown() を呼び出すと、JTAPI では Cisco Unified Communications Manager との通信が恒久的に失われ、ProvShutdownEv イベントがアプリケーションに通知されます。Provider が再度起動されることはないと想定し、アプリケーションによってシャットダウンを完全に処理する必要があります。

Provider.getTerminals()

このメソッドは、ディレクトリ上にあるユーザ制御リストで管理されるデバイスに対して作成される端末の配列を返します。ユーザ制御リストの管理については、*Cisco Unified Communications Manager Administration Guide* を参照してください。

Provider.getAddresses()

このメソッドは、ディレクトリ上にあるユーザ制御リストで管理されるデバイスに対して割り当てられる回線から作成されるアドレスの配列を返します。

ディレクトリ上のユーザ制御リストの変更

JTAPI アプリケーションの起動後にユーザ制御リストにデバイスが追加された場合、CiscoTermCreatedEv と各 CiscoAddrCreatedEv が生成され、CiscoProviderObserver の実装されたオブザーバに送信されます。また、アプリケーション側から、制御対象デバイスの現在の登録状態を監視し、これらのデバイスのアベイラビリティを動的に追跡することができます。イン サービスの Address または Terminal に対するイベントは、CiscoAddressObserver と CiscoTerminalObserver を実装しているオブザーバに通知されます。

（注） オブザーバの実装だけでは不十分であり、アドレスの場合は address.addObserver() メソッドを使用して、端末の場合は terminal.addObserver() メソッドを使用してオブザーバをそれぞれ追加する必要があります。

（注） call.connect() メソッドを呼び出す前に、発呼側のアドレスまたは端末に CallObserver を追加します。追加しないと、このメソッドによって例外が返されます。

Address と Terminal の関係

Cisco Unified Communications システムのアーキテクチャには 3 種類の基本エンドポイントがあります。

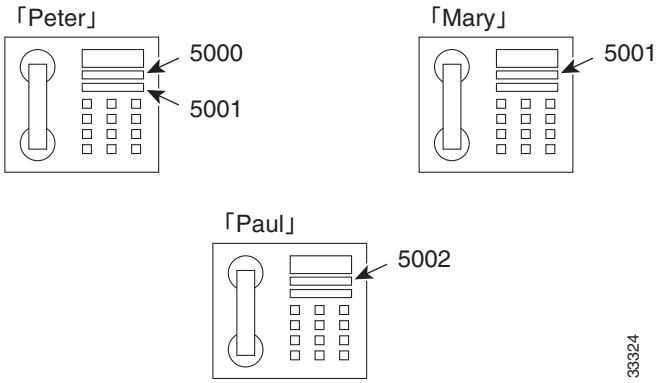
- 電話機
- 仮想デバイス（メディア終端点とルート ポイント）
- ゲートウェイ

これらのエンドポイントのうち、電話機とメディア終端点だけが Cisco Unified JTAPI を実装することにより使用されます。

Cisco Unified Communications Manager を使用するとユーザは、1 つ以上の回線やダイヤル可能な番号を電話機に設定して、その回線や番号を複数の電話機間で同時に共用したり、複数の回線を一度に 1 台の電話機だけで排他的に利用するように設定することができます。電話機には、回線あたり同時に複数のコールを終端させる機能（Maximum Number of Calls の設定に依存）があり、1 つのコール以外は保留になります。

これは、家庭用電話機の「コール ウェイティング」機能の動作に似ています。図 1-2 は、Peter と Mary が 1 本の電話回線 5001 を共有する設定と、Paul が専用の電話回線 5002 を使用する設定を示したものです。

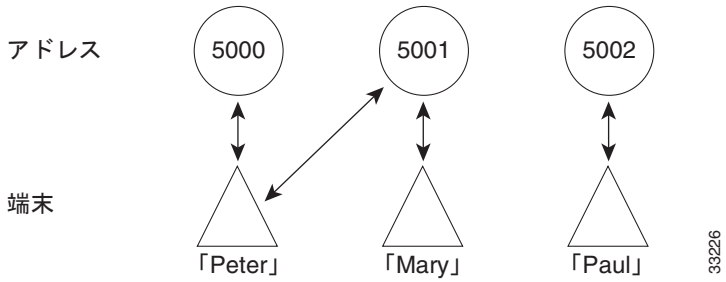
図 1-2　電話機の図



Cisco Unified Communications Manager では、あらゆる種類のエンドポイントが固有の名前によって識別されます。電話機の Media Access Control（MAC; メディア アクセス制御）アドレス（「SEP0010EB1014」など）で識別できますが、システム管理者は、区別可能であればどのような名前でもメディア終端点に割り当てることができます。

Cisco Unified JTAPI 実装では、プロバイダーによって制御される各エンドポイントごとに、管理者が割り当てた名前を使用して、対応する端末オブジェクトが構築されます。端末オブジェクトにもまた 1 つ以上のアドレス オブジェクトがあり、各アドレス オブジェクトはエンドポイント上の回線に対応します。図 1-3 の「Address と Terminal の関係」に Address と Terminal の関係を図示します。

図 1-3　Address と Terminal の関係



2 つ以上のエンドポイントで 1 本の回線（電話番号）を共用する場合は、対応するアドレス オブジェクトは 2 つ以上の端末オブジェクトに関連付けられます。

観察対象外の Address と Terminal

Cisco Unified JTAPI では、プロバイダーの端末とアドレスに CallObserver が付けられている場合にだけ、コールが認識されます。これは、Provider.getCalls() または Address.getConnections() などのメソッドでは、アドレスにコールがあってもこのアドレスに CallObserver が付けられていない場合は null が返されることを意味します。また、Call.connect() メソッドを使用して発呼するアドレスや端末にも、CallObserver を追加する必要があります。

Connection

Connection では、コールとアドレスへの恒久的な参照が保持されます。このため、コール イベントから取得した Connection の参照は、常に Connection のコール（getCall()）とアドレス（getAddress()）の取得に使用できます。

Terminal Connection

Terminal Connection では、常に端末と Connection への参照が保持されます。このため、コール イベントから取得した Terminal Connection の参照は、常に接続端末（getTerminal()）と Connection（getConnection()）の取得に使用できます。

端末とアドレスの制限

この端末とアドレスの制限では、管理者が Cisco Unified Communications Manager Administration の制限リストに特定の端末とアドレスのセットを追加した場合、それらの端末とアドレスをアプリケーションで制御または監視することが禁止されます。

管理者はデバイス上の特定の回線（特定の端末上のアドレス）を制限リストに追加できます。Cisco Unified Communications Manager Administration の制限リストに端末を追加した場合は、JTAPI でもその端末のすべてのアドレスが制限付きとしてマークされます。この設定が完了した後にアプリケーションを起動した場合は、CiscoTerminal.isRestricted() インターフェイスと CiscoAddress.isRestricted(Terminal) インターフェイスを確認することで、特定の端末またはアドレスが制限されているかどうかを認識することができます。共用回線の場合は、CiscoAddress.getRestrictedAddrTerminals() インターフェイスを照会することで、特定のアドレスがいずれかの端末で制限されているかどうかを確認できます。

アプリケーションの起動後に回線（端末上のアドレス）が制限リストに追加された場合は、CiscoAddrRestrictedEv がアプリケーションに通知されます。そのアドレスにオブザーバがある場合は CiscoAddrOutOfService がアプリケーションに通知されます。制限リストから回線が削除されると、CiscoAddrActivatedEv がアプリケーションに通知されます。そのアドレスにオブザーバがある場合は CiscoAddrInServiceEv がアプリケーションに送信されます。アプリケーションが制限リストに含まれるアドレスにオブザーバを追加しようとすると、PlatformException がスローされます。アドレスが制限される前に追加されたオブザーバはそのまま残りますが、そのアドレスが制限リストから削除されない限り、これらのオブザーバでイベントを取得できません。アプリケーションからアドレスのオブザーバを削除することもできます。

アプリケーションの起動後にデバイス（端末）が制限リストに追加された場合は、CiscoTermRestrictedEv がアプリケーションに送信されます。その端末にオブザーバがある場合は CiscoTermOutOfService がアプリケーションに送信されます。端末が制限リストに追加されると、JTAPI でもその端末に属するすべてのアドレスが制限され、CiscoAddrRestrictedEv がアプリケーションに通知されます。制限リストから端末が削除されると、CiscoTermActivatedEv と、対応するアドレスの CiscoAddrActivatedEv がアプリケーションに通知されます。アプリケーションが制限リストに含まれる端末にオブザーバを追加しようとすると、PlatformException がスローされます。端末が制限される前に追加されたオブザーバはそのまま残りますが、その端末が制限リストから削除されない限り、これらのオブザーバでイベントを取得できません。

アプリケーションの起動後に共用回線が制限リストに追加された場合は、CiscoAddrRestrictedOnTerminalEv がアプリケーションに通知されます。そのアドレスにアドレス オブザーバがある場合は、その端末の CiscoAddrOutOfServiceEv がアプリケーションに通知されます。すべての共用回線が制限リストに追加された場合は、最後の 1 つが追加された時点で、CiscoAddrRestrictedEv がアプリケーションに通知されます。アプリケーションの起動後に制限リストから共用回線が削除された場合は、CiscoAddrActivatedOnTerminalEv がアプリケーションに通知されます。そのアドレスにオブザーバがある場合は、その端末の CiscoAddrInServiceEv がアプリケーションに通知されます。制御リストに含まれるすべての共用回線が制限リストから削除された場合は、最後の 1 つが削除された時点で CiscoAddrActivatedEv がアプリケーションに通知され、端末上のすべてのアドレスが InService イベントを受信します。

制御リストに含まれるすべての共用回線が制限付きとしてマークされている場合、アプリケーションがオブザーバを追加しようとすると、PlatformException がスローされます。一部の共用回線のみが制限リストに追加されている場合、アプリケーションがそのアドレスにオブザーバを追加すると、制限されていない回線のみがイン サービスになります。

アドレスまたは端末が制限リストに追加されてリセットされたときにアクティブ コールが存在していた場合は、接続と TerminalConnections が接続解除として通知されます。

アドレスと端末が制限リストに 1 つも追加されていない場合、この機能は JTAPI の以前のバージョンと下位互換性をそのまま維持し、新しいイベントはアプリケーションに配信されません。

次のセクションでは、アドレスおよび端末の制限に関するインターフェイスの変更点について説明します。

CiscoTerminal

boolean **isRestricted()**

端末が制限されているかどうかを示します。端末が制限されている場合、その端末に関連付けられたすべてのアドレスも制限されます。端末が制限されている場合は true を返します。端末が制限されていない場合は false を返します。

CiscoAddress

javax.telephony.Terminal[] **getRestrictedAddrTerminals()**

このアドレスが制限されている端末の配列を返します。制限されている端末がない場合、このメソッドは null を返します。

共用回線の場合、端末上の少数の回線が制限されている可能性があります。このメソッドは、このアドレスが制限されているすべての端末を返します。アプリケーションは、制限された回線のコール イベントを認識できません。制限された回線が他の制御デバイスとのコールに関わっている場合、制限された回線用の外部接続が作成されます。

boolean **isRestricted**(javax.telephony.Terminal terminal)

この端末上のいずれかのアドレスが制限されている場合、true を返します。この端末上のアドレスが制限されていない場合、false を返します。

```
public interface CiscoRestrictedEv extends CiscoProvEv {
    public static final int ID = com.cisco.jtapi.CiscoEventID.CiscoRestrictedEv;

    /**
     * The following define the cause codes for restricted events
     */

    public final static int CAUSE_USER_RESTRICTED = 1;

    public final static int CAUSE_UNSUPPORTED_PROTOCOL = 2;

    }
```

これは制限されたイベントの基底クラスを表し、すべての制限されたイベントの原因コードを定義します。CAUSE_USER_RESTRICTED は、端末またはアドレスが制限とマークされていることを示します。CAUSE_UNSUPPORTED_PROTOCOL は、制御リスト内のデバイスが Cisco Unified JTAPI でサポートされていないプロトコルを使用していることを示します。SIP を実行している既存の Cisco Unified IP 7960 フォンおよび 7940 フォンはこれに該当します。

CiscoAddrRestrictedEv

public interface **CiscoAddrRestrictedEv** extends CiscoRestrictedEv。アプリケーションは、回線または関連するデバイスが Cisco Unified Communications Manager Administration から制限されたときに、このイベントを通知します。制限された回線では、アドレスがアウト オブ サービスになり、再び有効になるまでイン サービスに戻りません。アドレスが制限されている場合は、addCallObserver および addObserver によって例外がスローされます。共用回線では、いくつかの回線だけが制限されて残りは制限されなかった場合、例外はスローされませんが、制限された共用回線はイベントを受け取りません。すべての共用回線が制限された場合は、オブザーバを追加すると例外がスローされます。オブザーバを追加した後にアドレスが制限された場合、アプリケーションは CiscoAddrOutOfServiceEv を認識し、アドレスが有効にされるとイン サービスになります。

CiscoAddrActivatedEv

public interface **CiscoAddrActivatedEv** extends CiscoProvEv。アプリケーションは、回線または関連するデバイスが制御リストに含まれており、Cisco Unified Communications Manager Administration の制限リストから削除されたときに、このイベントを識別します。該当アドレスにオブザーバが存在する場合、アプリケーションは CiscoAddrInServiceEv を識別します。オブザーバが存在しない場合、アプリケーションはオブザーバの追加を試みることが可能で、アドレスはイン サービス状態になります。

CiscoAddrRestrictedOnTerminalEv

public interface **CiscoAddrRestrictedOnTerminalEv** extends CiscoRestrictedEv。ユーザが制御リストに共有アドレスを持っており、いずれかの回線を制限リストに追加した場合、このイベントが送信されます。getTerminal() インターフェイスは、アドレスが制限される端末を返します。getAddress() インターフェイスは、制限されるアドレスを返します。

javax.telephony.Address	**`getAddress()`**
javax.telephony.Terminal	**`getTerminal()`**

CiscoAddrActivatedOnTerminal

public interface **CiscoAddrActivatedOnTerminalEv** extends CiscoProvEv。共用回線または共用回線を持つデバイスを制限リストから削除すると、このイベントが送信されます。getTerminal() インターフェイスは、アドレスに追加される端末を返します。getAddress() インターフェイスは、新しい端末が追加されるアドレスを返します。

javax.telephony.Address	**`getAddress()`**
javax.telephony.Terminal	**`getTerminal()`**

CiscoTermRestrictedEv

public interface **CiscoTermRestrictedEv** extends CiscoRestrictedEv。アプリケーションは、アプリケーションの起動後にデバイスが Cisco Unified Communications Manager Administration から制限リストに追加されたときに、このイベントを認識します。アプリケーションは、制限された端末やその端末のアドレスに関するイベントを識別できません。InService 状態にある端末が制限された場合、アプリケーションはこのイベントを受信し、端末およびそれに対応するアドレスはアウト オブ サービス状態になります。

CiscoTermActivatedEv

public interface **CiscoTermActivatedEv** extends CiscoRestrictedEv。

javax.telephony.Terminal	**getTerminal()** 有効化されて制限リストから削除された端末を返します。

CiscoOutOfServiceEv

static int	**CAUSE_DEVICE_RESTRICTED** デバイスが制限されたためにイベントが送られたかどうかを示します。
static int	**CAUSE_LINE_RESTRICTED** 回線が制限されたためにイベントが送られたかどうかを示します。

CiscoCallEv

static int	**CAUSE_DEVICE_RESTRICTED** デバイスが制限されたためにイベントが送られたかどうかを示します。
static int	**CAUSE_LINE_RESTRICTED** 回線が制限されたためにイベントが送られたかどうかを示します。

CiscoConnectionID

CiscoConnectionID オブジェクトは、Cisco Unified JTAPI の各接続に関連付けられた固有のオブジェクトを表します。アプリケーションでは、オブジェクト自体またはオブジェクトの整数表現を使用できます。

コールバックのスレッド化

Cisco Unified JTAPI 実装の設計では、アプリケーションによって Call.connect() と TerminalConnection.answer() などのブロッキング用 JTAPI メソッドを、そのオブザーバのコールバック内から呼び出すことが可能です。これは、オブザーバのコールバック内から JTAPI メソッドを使用しないように警告する JTAPI 1.2 仕様の制限に、アプリケーションが制約されないことを意味します。

CiscoSynchronousObserver インターフェイス

Cisco Unified JTAPI 実装では、アプリケーションによって Call.connect() と TerminalConnection.answer() などのブロッキング用 JTAPI メソッドを、オブザーバのコールバック内から呼び出すことが可能です。これは、オブザーバのコールバック内部から JTAPI メソッドを使用しないように警告する JTAPI 1.2 仕様の制限に、アプリケーションが制約されないことを意味します。アプリケーションでは、そのオブザーバ オブジェクト上に CiscoSynchronousObserver インターフェイスを実装して、Cisco Unified JTAPI 実装のキューイング ロジックを選択的に無効にできます。

多くのアプリケーションは、この非同期動作の悪影響を受けることはありません。オブザーバ コールバック中にコヒーレント コール モデルの機能を使用するアプリケーションでは、Cisco Unified JTAPI 実装のキューイング ロジックを選択的に無効にできます。対象のオブザーバ オブジェクト上に CiscoSynchronousObserver インターフェイスを実装することで、アプリケーションからそのオブザーバに通知同期イベントが宣言されます。同期オブザーバに通知されるイベントは、オブザーバ コールバック内から照会されるコール モデル オブジェクトの状態と一致します。

（注） CiscoSynchronousObserver インターフェイスの実装されたオブジェクトでは、そのイベント コールバック内部からブロッキング用 JTAPI メソッドを呼び出さないでください。これを行った場合の影響は予測不能で、JTAPI 実装がデッドロックに陥る可能性があります。ただし、これらのオブジェクトでは、インスタンス Call.getConnections() または Connection.getState() のすべての JTAPI オブジェクトのアクセス用メソッドは安全に使用できます。

動的オブジェクトを照会する

コール オブジェクトのような動的オブジェクトの照会には注意が必要です。イベントを取得する時間までに、オブジェクト（コールなど）は示された状態とは別の状態で存在している場合があります。たとえば、CiscoTransferStartEV を取得する時間までに、転送コールによってその内部接続がすべて削除されている場合があります。

callChangeEvent()

callChangedEvent() メソッドが呼び出されときには、イベントに格納された参照の妥当性は引き続き保証されます。たとえば、イベントに getConnection() メソッドがある場合、アプリケーションでこのメソッドを呼び出して有効な接続の参照を取得できます。同様に、getCallingAddress() メソッドでは有効な Address オブジェクトを返すことが保証されます。

CiscoConsultCall

CiscoConsultCall インターフェイスの場合、コンサルティング端末接続への参照が恒久的に保持されます。たとえば、CiscoConsultCallActive イベントの処理時では、getConsultingTerminalConnection() は有効な端末接続の参照を返すことを保証します。また、端末接続は、コンサルティング接続とそのコンサルティング コールへのアクセスを保証します。

CiscoTransferStartEv

CiscoTransferStartEv の場合、callChangedEvent() が呼び出されると、イベント内にある転送コール、転送コントローラおよび最後のコールへの参照が有効になります。ただし、getConnections() を呼び出したときには、getConnections() から接続が返される場合と返されない場合があります。

アラーム サービス

Cisco Unified Communications アプリケーションに対する一般的なサービス フレームワークの一部として、サービスへのアラームの送信がサポートされています。com.cisco.services.alarm パッケージでは、アラーム コンポーネントが定義されています。

アラームのインターフェイスとフレームワークでは、Cisco Unified JTAPI アプリケーションのネットワーク上で使用可能なアラーム サービスへ、XML 形式のアラーム通知を TCP 経由で送信することがサポートされます。アラーム パッケージには、次の機能があります。

- アラーム サービスのカタログで解決される、アラームの XML 定義
- 送信側でアラームをバッファする、制限があり上書きされるキュー
- 送信アプリケーション側でブロッキングを回避する、別のスレッド上でのアラーム送信
- アラーム サービスへの TCP ベースの再接続スキーム

Cisco Unified JTAPI アラーム システムの全体的なフレームワークは、既存の JTAPI トレース パッケージに似た点が含まれています。アプリケーションでは、アラーム オブジェクトを作成可能な特定のファシリティ コード用の AlarmManager をインスタンスにする必要があります。実装の一部として、DefaultAlarm および DefaultAlarmWriter 実装クラスが含まれています。

ソフトウェア要件

次の表では、JTAPI アプリケーション、JTPREFS、およびサンプル コードのソフトウェア要件を示しています。

アプリケーション	ソフトウェア要件	例
JTAPI アプリケーション	いずれかの JDK 1.4.2 対応 Java 環境	• Internet Explorer 4.01 以降 • Sun JDK 1.4.2 または 1.5
JTPREFS	いずれかの JDK 1.4.2 対応環境	
コード例	Microsoft Internet Explorer 4.01 以降	