CHAPTER 3

Cisco Unified JTAPI でサポートされる機能

この章では、Cisco Unified JTAPI 仕様でサポートする機能について説明し、次のトピックについて説明します。

任意の通話者のドロップ（Drop Any Party）

この機能は、任意の参加者を電話会議から終了させる機能です。Cisco Unified JTAPI により、アプリケーションが接続のためにアドレスを監視していない場合でも、既存のインターフェイス Connection.disconnect() を使用して、参加者を会議から終了させることができます。これまで、アプリケーションでは、アドレスが監視対象のアドレスである接続のみ切断できました。

機能の動作は、Unified CM サービス パラメータ Advanced Ad Hoc Conference Enabled の設定に基づいて異なります。このサービス パラメータが False に設定されている場合、アプリケーションは会議コントローラのアドレスを監視している場合にのみ、電話会議の監視していないアドレスの接続を終了できます。このパラメータが True に設定されている場合、アプリケーションは無制限で接続を終了できます。

Cisco Unified JTAPI は、CiscoConnection で、接続のための CiscoPartyInfo オブジェクトの配列を取得するインターフェイスを提供します。CiscoPartyInfo は、CiscoConnection で提供されている新しいインターフェイス disconnect() を使用して、参加者を会議から切断させることができます。通常の回線では、その接続に CiscoPartyInfo が 1 つしかありませんが、共用回線には、共用回線の回線ごとに 1 つの CiscoPartyInfo があります。これにより、電話会議に複数の共用回線の参加者がいる場合に、アプリケーションは共用回線の参加者を選択して切断できます。共用回線の参加者の接続は 1 つしかないため、アプリケーションで既存の Connection.disconnect() API を使用した場合、すべての共用回線の参加者が切断されます。

Cisco Unified JTAPI では、CiscoJtapiProperties で、この機能を有効または無効にするインターフェイス setDropAnyPartyEnabled() を提供しています。デフォルトでこれは有効になっています。または、アプリケーションは、jtapi.ini ファイルの JTAPI ini パラメータ dropAnyPartyEnabled=0 を使用して、任意の通話者のドロップ（Drop Any Party）機能を無効にし、dropAnyPartyEnable=1 を使用して、この機能を有効にできます。jtapi.ini ファイルに dropAnyPartyEnable パラメータが存在しない場合、この機能はデフォルトで有効になります。

Cisco Unified JTAPI では、CiscoCall で、コールが電話会議であるかどうかを判断するためのインターフェイス isConferenceCall() も提供しています。この簡単なメソッドはブール値を返します。

インターフェイスの変更

「CiscoCall」（P.6-70）および「CiscoConnection」（P.6-123）を参照してください。

メッセージ シーケンス

「任意の通話者のドロップ（Drop Any Party）の使用例」（P.A-233）を参照してください。

下位互換性

この機能は下位互換性があります。

IPv6 のサポート

この機能は、IPv6 アドレスをサポートし、CiscoUnified JTAPI が CTIManager への IPv6 接続をサポートするように拡張されています。この機能により、Cisco Unified JTAPI アプリケーションは、IPv6 のサポート機能が有効にされ、発信側が IPv6 対応電話を使用している場合に、発信側アドレスとして、IPv6 アドレスを認識することができます。この機能では、次の関数をサポートしています。

- Cisco Unified JTAPI では、CiscoProviderCapabilities インターフェイスで、Cisco Unified Communications Manager の設定で IPv6 がサポートされているかどうかを示す新しい API canSupportIPv6() を公開しています。
- 以前に登録されている設定と現在の設定に不一致がある場合、Cisco Unified JTAPI はメディアまたはルート端末を閉じます。CiscoTermRegistrationFailedEv および新しい原因コード IP_ADDRESSING_MODE_ MISMATCH がこのシナリオに従って送信されます。
- 端末の IP アドレス機能は、CiscoTerminal インターフェイスの getIPAddressingMode() によって API に公開されています。IP アドレス機能は、CiscoTerminal/CiscoMediaTerminal および CiscoRouteTerminal で使用できます。
- IPv6 発信側 IP アドレスは、IPv4 対応デバイスの場合に IPv4 アドレスが提供されるのと同様に、CallCtlConnOfferedEv および RouteEvent の Cisco の拡張機能として InetAddress オブジェクトで提供されます。

監視対象のデバイスが IPv6 デバイスの場合に、CiscoRTPOutputStartedEv および CiscoRTPInputStartedEv の RTP アドレスにも IPv6 アドレスがあります。つまり、CiscoRTPInputProperties の API getLocalAddress() と CiscoRTPOutputProperties の API getRemoteAddress() で、IPv6 形式の IP アドレスを返せるようになりました。API は InetAddress オブジェクトを返し、アプリケーションでは、それが Inet4Address のインスタンスであるか、Inet6Address のインスタンスであるかを確認して、IPv4 形式の IP アドレスか、IPv6 形式の IP アドレスかを判断できます。

アプリケーションでは、IP アドレッシング モードを変更したら、デバイスをリセットする必要があります。そうしないと、予想した結果にならないことがあります。

リリース 7.1 より、Cisco Unified JTAPI では、CiscoTerminal で getIPAddressingMode() API を提供しています。CTI ポートおよびルートポイントの getIPAddressingMode() API もこのリリースからサポートされています。

Cisco Unified JTAPI では CiscoTerminal の同じ API を拡張し、この API は Unified CM の管理ページに、IP フォンの設定済みの IP アドレッシング モードを返します。デバイスの登録後に、ユーザが Unified CM の管理ページから、IP アドレッシング モードを変更した場合、デバイスをリセットする必要があります。更新された値は、IP フォンのリセット後にのみ Cisco Unified JTAPI で参照可能になります。設定済みの IP アドレッシング モードがデュアル スタックである場合、つまり、IPv4 アドレスと IPv6 アドレスがサポートされている場合、電話はこれらのいずれかまたは両方のアドレスで登録できます。これは、ネットワークのタイプと Unified CM の IPv6 のサポートなどの状況によって異なります。IP アドレッシング モードがデュアル スタックである場合、CiscoTerminal の getIPAddressingMode() は CiscoTerminal.IP_ADDRESSING_MODE_IPV4_V6 を返します。

インターフェイスの変更

「CiscoTerminal」（P.6-307）を参照してください。

メッセージ シーケンス

「IPv6 のサポート」（P.A-184）および「IPv6 のサポート」（P.A-296）を参照してください。

下位互換性

この機能は下位互換性があります。

回線をまたいで直接転送（Direct Transfer Across Lines）

回線をまたいで直接転送（Direct Transfer Across Lines）機能により、回線をまたいだ転送をサポートできます。この機能により、電話の [転送（Transfer）] ソフトキーを押すか、Cisco Unified JTAPI で提供されている transfer() API を使用して、同じ端末の異なるアドレスの 2 つのコールを転送できます。回線をまたいで転送を実行すると、アプリケーションは最後のコールとコンサルト コールで共通のコントローラを認識しないため、JTAPI アプリケーションの動作が変わります。API 自体に変更はなく、会議に参加するコールが同じアドレス（通常の転送）か異なるアドレス間（回線をまたいだ直接転送）かどうかに関係なく、同じイベントが配信されます。この機能は、CTI ポート、SCCP デバイス、SIP デバイスなど、サポートされるすべての電話でサポートされています。

JTAPI API から直接転送が試みられている 2 つのアドレスのいずれかにオブザーバが追加されていない場合、Cisco Unified JTAPI は、エラー「Transfer controller is not set and could not find a suitable TerminalConnection」と共に PlatformException をスローします。

使用上のガイドライン

次のポイントでは、アプリケーションで回線をまたいで直接転送（Direct Transfer Across Lines）機能をどのように使用する必要があるかを示します。

- アプリケーションは、直接転送を試みる両方の回線にコール オブザーバを追加する必要があります。
- 以前のバージョンでは、アプリケーションが、両方のコールで共通のアドレスを使用しているかどうか、およびそのアドレスが同じ端末上にあるかどうかを確認することを推奨していました。回線をまたいで直接転送（Direct Transfer Across Lines）では、直接転送が呼び出される 2 つのコール間で、アドレスが共通かを確認する必要はありません。両方のコールがそれぞれ、共通の端末に存在するアドレスを持つことは確認する必要があります。
- Cisco Unified JTAPI は、現在と同様に、同じアドレスのコールの転送と同じ一連のイベントを報告します。アプリケーションは、Transfer() の呼出し後、CiscoTransferEndEv を受け取るまで、これらのコールについて何もする必要はありません。
- 転送がアドレス間で行われるため、アプリケーションでは CiscoTransferStartEv で共通のコントローラを取得しないため、アプリケーション ロジックをアップグレードする必要があります。

イベント フローの比較とコード例

表 3-1 に、イベント フローの詳細とコード例を示します。

表 3-1　転送呼び出しのイベント フローの比較とコード例

同じ回線での転送	回線をまたがった転送
セットアップ	
端末 T1 のアドレス A 端末 T2 のアドレス B1、B2 端末 T3 のアドレス C	端末 T1 のアドレス A 端末 T2 のアドレス B1、B2 端末 T3 のアドレス C
機能の呼び出し	

表 3-1　転送呼び出しのイベント フローの比較とコード例 （続き）

同じ回線での転送	回線をまたがった転送
A が B1 にコール [GC1=GolbalCallID1]	A が B1 にコール [GC1=GolbalCallID1]
GC1：接続 A1-> Conn1	GC1：接続 A->Conn1
GC1：接続 B1->Conn2	GC1：接続 B1->Conn2
B1 が C にコール [GC2=GolbalCallID2]	B2 が C にコール [GC2=GolbalCallID2]
GG2：接続 B1->Conn3	GG2：接続 B2->Conn3
GC2：接続 C->Conn4	GC2：接続 C->Conn4
GC1.transfer(GC2);	GC1.transfer(GC2);
アプリケーションに配信されるイベント（すべての通話者が監視されているものとする）	

表 3-1　転送呼び出しのイベント フローの比較とコード例 （続き）

同じ回線での転送	回線をまたがった転送
GC1:	GC1:
CiscoTransferStartEv	CiscoTransferStartEv
[getTransferControllerAddress() は B1 を返す]	[getTransferControllerAddress() は B1 を返す]
ConnCreatedEv for C	ConnCreatedEv for C
ConnConnectedEv for C	ConnConnectedEv for C
CallCtlConnEstablishedEv for C	CallCtlConnEstablishedEv for C
TermConnCreatedEv for T3(Address C)	TermConnCreatedEv for T3(Address C)
ConnDisconnectedEv for B1	ConnDisconnectedEv for B1
CallCtlConnDisconnectedEv for B1	CallCtlConnDisconnectedEv for B1
TermConnDroppedEv for T2(Address B1)	TermConnDroppedEv for T2(Address B1)
CallCtlTermConnDroppedEv for T2(Address B1)	CallCtlTermConnDroppedEv for T2(Address B1)
CiscoTransferEndEv	CiscoTransferEndEv
GC2:	GC2:
CiscoTransferStartEv	CiscoTransferStartEv
[getTransfeControllerAddress() は B1 を返す]	[getTransferControllerAddress() は B1 を返す]
TermConnDroppedEv for T2(Address B1)	TermConnDroppedEv for T2(Address B2)
CallCtlTermConnDroppedEv for T2(Addresss B1)	CallCtlTermConnDroppedEv for T2(Addresss B2)
ConnDisconnectedEv for B1	ConnDisconnectedEv for B2
CallCtlConnDisconnectedEv for B1	CallCtlConnDisconnectedEv for B2
TermConnDroppedEv for T3(Address C)	TermConnDroppedEv for T3(Address C)
ConnDisconnectedEv for C	ConnDisconnectedEv for C
CallCtlConnDisconnectedEv for C	CallCtlConnDisconnectedEv for C
CallCtlTermConnDroppedEv for T3(Address C)	CallCtlTermConnDroppedEv for T3(Address C)
CiscoTransferEndEv	CiscoTransferEndEv
CallInvalidEv	CallInvalidEv
CallObservationEndedEv	CallObservationEndedEv
（注）　GC2：端末 T2 のアドレス B1 に対する切断イベント	（注）　GC2：端末 T2 のアドレス B2 に対する切断イベント

表 3-1　転送呼び出しのイベント フローの比較とコード例 （続き）

同じ回線での転送	回線をまたがった転送
アプリケーション コード例	

Handle(CiscoCallEv event) {
…..
…..
if (event instanceOf
CiscoTransferStartEv){
CiscoTransferStartEv ev =
(CiscoTransferStartEv)event;
processTransfer(ev);
}
}
processTransfer(CiscoTransferStartEv ev){
CiscoAddress commonAddr =
ev.getTransferControllerAddress();
 	
CiscoCall GC2 = ev.getTransferringCall();
CiscoCall GC1 = ev.getFinalCall();
CiscoConnection droppedConn1 =
findConnection(GC1, controllerAddr);
 	
CiscoConnection droppedConn2 =
findConnection(GC2, controllerAddr);
//Additional App logic to clear
connections.
}
Connection findConnection(CiscoCall GCx,
CiscoAddress addr){
CiscoConnection[] conns =
GCx.getConnections();
for (i=0; i<conns.length; i++){
if conns[i].getAddress().equals(addr)
{
return conns[i];
}
}
}
 	
（注） アプリケーション ロジックは、共通の transferControllerAddress に基づき、最後のコールとコンサルト コールの両方に commonAddr が あるため、この例では正常に機能します。
	Handle(CiscoCallEv event) {
	…..
	…..
	if (event instanceOf
	CiscoTransferStartEv){
	CiscoTransferStartEv ev =
	(CiscoTransferStartEv)event;
	processTransfer(ev);
	}
	}
	processTransfer(CiscoTransferStartEv ev){
	String termName =
	ev.getControllerTreminalName();
	
	CiscoCall GC2 = ev.getTransferringCall();
	CiscoCall GC1 = ev.getFinalCall();
	CiscoConnection droppedConn1 =
	findConnection(GC1, termName);
	
	CiscoConnection droppedConn2 =
	findConnection(GC2, termName);
	//Additional App logic to clear
	connections.
	}
	Connection findConnection(CiscoCall GCx,
	String termName){
	CiscoConnection[] conns =
	GCx.getConnections();
	for (i=0; i<conns.length; i++){
	CiscoTerminalConnection[] termConns =
	conns[i].getTerminalConnections();
	for(j=0; j<termConns.length; j++){
	
	if(termConns[j].getTerminal().getName.equal
	s(termName)
	&& termConns[i].getState()
	!=
	
	TerminalConnection.PASSIVE){
	return
	termConns[i].getConnection();
	}
	}
	}
	}
	（注） 最後のコールとコンサルト コールでコントローラの共通のアドレスはありませんが、コントローラ TerminalName は両方のコントローラ アドレスで同じです。そのため、アプリケーションは、CommonTerminalName を使用して、接続、TerminalConnection、およびコントローラを検出することができます。

（注）　回線をまたいだ接続転送のシナリオでは、説明したイベントと別に、アプリケーションで、別の一時コール GC3 がアクティブになり（CallActiveEv）、転送の完了直後に GC3 がアイドルになる（CallInvalidEv）ことを認識できます。

インターフェイスの変更

「CiscoTransferStartEv」（P.6-343）を参照してください。

メッセージ シーケンス

「回線をまたいで直接転送（Direct Transfer Across Lines）の使用例」（P.A-207）を参照してください。

下位互換性

この機能は下位互換性があります。アプリケーションに下位互換性を提供するため、回線をまたいで転送を可能にするデバイスへの新しい権限のほか、この権限の新しい標準権限と標準ユーザ グループが追加されました。この新しい権限 Standard Supports Connected Xfer/Conf がアプリケーション ユーザに関連付けられている場合にのみ、アプリケーションはこれらのデバイスを制御できます。アプリケーションは、この新しい権限「Standard CTI Allow Control of Phones supporting Connected Xfer and Conf」がアプリケーション ユーザに関連付けられている場合にのみ、これらのデバイスを制御できます。デフォルトでこれらのデバイスは制限付きとして表示され、アプリケーションで JTAPI 7.1.2 以上を使用することを前提とし、アプリケーションがこの機能を処理できるようにアップグレードされ、新しい権限を関連付けている場合にのみ、これらのデバイスを制御できます。アプリケーションで古い JTAPI クライアントを使用している場合、デバイスは制限されませんが、アプリケーションがこれらのデバイス（この機能の手動での起動をサポートしている）を監視しようとすると、JTAPI は例外をスローし、その時点からこれらのデバイスを制限付きとしてマークします。

ただし、アプリケーションは任意のタイプの電話で、既存の JTAPI transfer() API から 回線をまたいで直接転送（Direct Transfer Across Lines）を呼び出すことができ、アプリケーションは、この機能をサポートする場合にのみこの要求を発行することが予想されているため、この動作は制限されません。さらに、回線をまたいで（直接）転送（Direct/Connected Transfer Across Lines）を実行する FarEnd ポイントも制御されないため、アプリケーションに問題が発生する可能性があります。これはつまり、JTAPI が常にすべての電話について回線をまたいで直接転送（Direct Transfer Across Lines）のイベントを報告することを意味します。

古い JTAPI アプリケーションで、回線をまたいで直接転送（Direct Transfer Across Lines）が呼び出されていない（電話で、または JTAPI API を介して）環境で実行した場合、下位互換性の問題はまったく発生しません。ただし、そのような設定で、この機能を使用する場合は、アプリケーションの変更が必要になります。

シスコでは、複数のアプリケーションで同じ端末を制御または監視したり、同時に処理したりしないことを前提としています。アプリケーションでそのように実行する場合は、このアプリケーションのすべてのインスタンスを変更して、この機能をサポートするか、問題を避けるように調整します。そうしないと、アプリケーションの動作が予測できなくなることがあります。たとえば、App1 と App2 が同じ端末またはアドレスを制御または監視している 2 つのアプリケーションで、App1 がこの機能をサポートするように変更された場合、App2 もこの機能をサポートするように変更する必要があります。そうしないと、共通デバイスでの App1 によるこの機能の呼出しが App2 を切断させる可能性があります。

この機能は、ユーザの操作性を拡張するために設計されているため、すべての Cisco Unified JTAPI アプリケーションでこの機能を評価して、サポートし、必要に応じて、古い動作と新しい動作の両方を処理するコード ロジックでアップグレードすることをお勧めします。

回線をまたいで参加（Join Across Lines）または Connected Conference Across Lines

このリリースでは、既存の Join Across Lines サービス パラメータの範囲外の電話 Cisco Unified IP Phone モデルを採用することによって、ユーザの操作性を拡張しています。これらの電話では、この機能が常に有効にされており、これをオフにするサービス パラメータはありません。機能の詳細説明、インターフェイスの変更に関する情報、使用例については、「Join Across Lines with Conference 機能拡張」（P.3-17）を参照してください。

使用上のガイドライン

次のポイントでは、アプリケーションで回線をまたいで参加（Join Across Lines）機能をどのように使用する必要があるかを示します。

- アプリケーションは回線をまたいで、または接続された会議に参加しようとする両方の回線にコール オブザーバを追加する必要があります。
- 以前のバージョンでは、アプリケーションが、両方のコールで共通のアドレスを使用しているかどうか、およびその共通のアドレスが同じ端末上にあるかどうかを確認することを推奨しました。回線をまたいで参加（Join Across Lines）では、直接会議が呼び出される 2 つのコール間で、アドレスが共通であるかどうかを確認する必要はありません。両方のコールがそれぞれ、共通の端末に存在するアドレスを持つことは確認する必要があります。
- Cisco Unified JTAPI は、現在と同様に、同じアドレスのコールの会議の開始についてと同じ一連のイベントを報告します。アプリケーションは、Conference() の呼出し後、CiscoConferenceEndEv を受け取るまで、これらのコールについて何もする必要はありません。
- 会議がアドレス間で行われる場合は、アプリケーションは CiscoConferenceStartEv で共通のコントローラを取得しないため、アプリケーション ロジックをアップグレードする必要があります。詳細については表 3-2 を参照してください。

イベント フローの比較とコード例

表 3-2 に、イベント フローの詳細とコード例を示します。

表 3-2　会議呼び出しのイベント フローの比較とコード例

同じ回線で参加	回線をまたいで参加
セットアップ	
端末 T1 のアドレス A 端末 T2 のアドレス B1、B2 端末 T3 のアドレス C	端末 T1 のアドレス A 端末 T2 のアドレス B1、B2 端末 T3 のアドレス C
機能の呼び出し	

表 3-2　会議呼び出しのイベント フローの比較とコード例 （続き）

同じ回線で参加	回線をまたいで参加
A が B1 にコール [GC1=GolbalCallID1]	A が B1 にコール[GC1=GolbalCallID1]
GC1：接続 A1' Conn1	GC1：接続 A' Conn1
GC1：接続 B1' Conn2	GC1：接続 B1' Conn2
B1 が C にコール [GC2=GolbalCallID2]	B2 が C にコール [GC2=GolbalCallID2]
GG2：接続 B1' Conn3	GG2：接続 B2' Conn3
GC2：接続 C' Conn4	GC2：接続 C' Conn4
GC1.conference(GC2);	GC1.conference(GC2);
アプリケーションに配信されるイベント（すべての通話者が監視されているものとする）	
GC1:	GC1:
CiscoConferenceStartEv	CiscoConferenceStartEv
[getConferenceControllerAddress() は B1 を返す]	[getConferenceControllerAddress() は B1 を返す]
ConnCreatedEv for C	ConnCreatedEv for C
ConnConnectedEv for C	ConnConnectedEv for C
CallCtlConnEstablishedEv for C	CallCtlConnEstablishedEv for C
TermConnCreatedEv for T3(Address C)	TermConnCreatedEv for T3(Address C)
CiscoConferenceEndEv	CiscoConferenceEndEv
GC2:	GC2:
CiscoConferenceStartEv	CiscoConferenceStartEv
[getConferenceControllerAddress() は B1 を返す]	[getConferenceControllerAddress() は B1 を返す]
TermConnDroppedEv for T2(Address B1)	TermConnDroppedEv for T2(Address B2)
CallCtlTermConnDroppedEv for T2(Addresss B1)	CallCtlTermConnDroppedEv for T2(Addresss B2)
ConnDisconnectedEv for B1	ConnDisconnectedEv for B2
CallCtlConnDisconnectedEv for B1	CallCtlConnDisconnectedEv for B2
TermConnDroppedEv for T3(Address C)	TermConnDroppedEv for T3(Address C)
ConnDisconnectedEv for C	ConnDisconnectedEv for C
CallCtlConnDisconnectedEv for C	CallCtlConnDisconnectedEv for C
CallCtlTermConnDroppedEv for T3(Address C)	CallCtlTermConnDroppedEv for T3(Address C)
CiscoConferenceEndEv	CiscoConferenceEndEv
CallInvalidEv	CallInvalidEv
CallObservationEndedEv	CallObservationEndedEv
（注）　GC2：端末 T2 のアドレス B1 に対する切断イベント	（注）　GC2：端末 T2 のアドレス B2 に対する切断イベント
アプリケーション コード例	

表 3-2　会議呼び出しのイベント フローの比較とコード例 （続き）

同じ回線で参加	回線をまたいで参加

Handle(CiscoCallEv event) {
…..
…..
if (event instanceOf
CiscoConferenceStartEv){
CiscoConferenceStartEv ev =
(CiscoConferenceStartEv)event;
processConference(ev);
}
}
processConference(CiscoConferenceStartEv
ev){
CiscoAddress controllerAddr =
ev.getConferenceControllerAddress();
 	
CiscoCall[] consultCalls =
ev.getConferenedCalls();
CiscoCall GC1 = ev.getFinalCall();
CiscoConnection[] movedConns[] =
findConnections(consultCalls,
controllerAddr);
 	
//Additional App logic to clear
connections.
}
Connection[] findConnections(CiscoCall[]
calls, CiscoAddress addr){
ArrayList connList = new ArrayList();
for(x=0; x < calls.length; x++){
CiscoConnection[] conns =
calls[x].getConnections();
for (i=0; i<conns.length; i++){
if conns[i].getAddress().equals(addr)
{
connList.add(conns[i]);
}
}
}
return connList.toArray(Connection[]
conns);
}
	Handle(CiscoCallEv event) {
	…..
	…..
	if (event instanceOf
	CiscoConferenceStartEv){
	CiscoConferenceEv ev =
	(CiscoConferenceStartEv)event;
	processConference(ev);
	}
	}
	processConference(CiscoConferenceStartEv
	ev){
	String controllerTermName =
	ev.getControllerTreminalName();
	CiscoCall[] consultCalls =
	ev.getConferenedCalls();
	CiscoCall GC1 = ev.getFinalCall();
	CiscoConnection[] movedConns =
	findConnections(consultCalls,
	controllerTermName);
	
	//Additional App logic to clear
	connections.
	}
	Connection[] findConnections(CiscoCall
	calls, String
	termName){
	ArrayList connList = new ArrayList();
	for(x=0; x < calls.length; x++){
	CiscoConnection[] conns =
	calls[x].getConnections();
	for (i=0; i<conns.length; i++){
	CiscoTerminalConnection[] termConns =
	conns[i].getTerminalConnections();
	for(j=0; j<termConns.length; j++){
	
	if(termConns[j].getTerminal().getName.equal
	s(termName)
	&& termConns[i].getState()
	!=
	
	TerminalConnection.PASSIVE){
	connList.add(conns[i]);
	}
	}
	}
	}
	return connList.toArray(Connection[]
	conns);
	}

（注）　アプリケーション ロジックは、共通の transferControllerAddress に基づき、最後のコールとコンサルト コールの両方に commonAddr が 存在するため、この例では正常に機能します。	（注）　最後のコールとコンサルト コールでコントローラの共通のアドレスはありませんが、コントローラ TerminalName は両方のコントローラ アドレスで同じです。そのため、アプリケーションは、CommonTerminalName を使用して、接続、TerminalConnection、およびコントローラを検出することができます。

（注）　Connected Conference Across Lines のシナリオでは、説明したイベントと別に、アプリケーションで、別の一時コール GC3 がアクティブになり（CallActiveEv）、会議の完了直後に GC3 がアイドルになる（CallInvalidEv）ことを認識できます。

インターフェイスの変更

「CiscoConferenceStartEv」（P.6-120）を参照してください。

メッセージ シーケンス

「Connected Conference または回線をまたいで参加（Join Across Lines）の使用例：新しい電話の動作」（P.A-213）を参照してください。

下位互換性

この機能は下位互換性があります。

この機能は特定のデバイスでオフにできず、Cisco Unified JTAPI はこれらの電話に対し、常に回線をまたいで参加（Join Across Lines）のイベントを報告します。ただし、アプリケーションに下位互換性を提供するため、これらのデバイスを制御し、Connected Conference Across Lines を可能にする新しい権限が追加されています。新しい標準権限「Standard CTI Allow Control of Phones supporting Connected Xfer and conf」と標準ユーザ グループも追加されています。アプリケーションは、JTAPI クライアント 7.1.2 以上を使用していることを前提として、この新しい権限がアプリケーション ユーザに関連付けられている場合にのみ、これらのデバイスを制御できます。そのため、デフォルトで、これらのデバイスは制限付きとして表示されます。この機能を処理するために、アプリケーションをアップグレードし、これらのデバイスを制御するための新しい権限を関連付ける必要があります。アプリケーションで古い JTAPI クライアントを使用している場合、デバイスは制限されませんが、アプリケーションがこれらのデバイス（この機能の手動での起動をサポートしている）を監視しようとすると、JTAPI は例外をスローし、その時点からこれらのデバイスを制限付きとしてマークします。

シスコでは、複数のアプリケーションで同じ端末を制御または監視したり、同時に処理したりしないことを前提としています。アプリケーションでそのように実行する場合は、このアプリケーションのすべてのインスタンスを変更して、この機能をサポートするか、問題を避けるように調整します。そうしないと、アプリケーションの動作が予測できなくなることがあります。たとえば、App1 と App2 が同じ端末またはアドレスを制御または監視している 2 つのアプリケーションで、App1 がこの機能をサポートするように変更された場合、App2 もこの機能をサポートする必要があります。そうしないと、共通デバイスでの App1 によるこの機能の呼出しが App2 を切断させる可能性があります。

この機能は、ユーザの操作性を拡張するために設計されているため、すべての Cisco Unified JTAPI アプリケーションでこの機能を評価して、サポートし、必要に応じて、古い動作と新しい動作の両方を処理するコード ロジックでアップグレードすることをお勧めします。

スワップまたはキャンセルと転送または会議の動作

この機能により、Cisco Unified JTAPI で、サポートされる IP フォンでのスワップおよびキャンセル操作をサポートできます。

スワップ操作を呼び出すと、アクティブなコールが保留になり、保留したコールが取得されます。キャンセル操作を呼び出すと、プライマリ コールとコンサルト コール間のコンサルト関係が破棄されます。これらの操作はサポートされる電話からのみ呼び出すことができます。Cisco Unified JTAPI インターフェイスでは、アプリケーションから、スワップ/キャンセル操作を呼び出すことができません。ユーザが電話の [SWAP（切替）] キーを押すと、JTAPI がアクティブな保留されているコールに CallCtlTermConnHeldEv および CallCtlTermConnTalkingEv を配信し、CiscoFeatureReason.REASON_NORMAL でそれらの状態変更を示します。

キャンセル操作が呼び出され、プライマリ コールとコンサルト コールの関係が破棄されても、Cisco Unified JTAPI は直接転送または参加機能を使用して、転送や会議操作を実行できます。コンサルトの開始後に、ユーザが電話の [CANCEL] キーを押した場合、会議や転送が実行されません。電話の [CANCEL（キャンセル）] キーを押すと、アプリケーションへのキャンセル通知がトリガーされます。Cisco Unified JTAPI によって、キャンセル操作を示す CiscoCallFeatureCancelledEv が送信されます。CiscoCallFeatureCancelledEv.getConsultCall() は以前に作成されたコンサルト コールを返します。

接続転送または会議中にキャンセル操作が実行されると、次のことが発生する可能性があります。

- ユーザがアクティブ コール ソフトキーを選択する前に、[CANCEL（キャンセル）] キーを押す。

  この場合、[Transfer（転送）] キーを押すと、consultCall GC3 が作成され、[CANCEL（キャンセル）] キーを押すと、コンサルト コールとして GC2 と GC3 で CiscoCallFeatureCancelledEv がトリガーされます。

- アクティブ コール ソフトキーを押した後、ただし電話 UI でコールを選択する前に、ユーザが [CANCEL（キャンセル）] キーを押す。

  この場合、電話 UI でアクティブ コール ソフトキーを押すと、consultCall GC3 が IDLE になりますが、他の機能の操作が可能であるため、CANCEL 通知はありません。ただし、ユーザが [CANCEL（キャンセル）] キーを押した場合、コンサルト コールを Null として CiscoCallConsultCancelEv がトリガーされます。

- ユーザがアクティブ コール ソフトキーを押し、コールを選択して、[CANCEL（キャンセル）] を押す。

  この場合、選択したコールは、CiscoCallFeatureCancelledEv で consultCall として返されます。

インターフェイスの変更

「CiscoCallFeatureCancelledEv」（P.6-102）を参照してください。

メッセージ シーケンス

「スワップ / キャンセルおよび転送 / 会議の動作変更」（P.A-221）を参照してください。

下位互換性

この機能は下位互換性があります。

このリリースでは、スワップまたはキャンセル機能が有効にされており、それをオフにするサービス パラメータはありません。つまり、Cisco Unified JTAPI は常に、この機能をサポートする電話のスワップまたはキャンセルのイベントをサポートし、報告することを意味します。

ただし、アプリケーションの下位互換性を提供するため、これらのデバイスの制御を可能にし、スワップまたはキャンセル操作を可能にする新しい権限が追加されています。新しい標準権限 Standard Supports Connected Xfer/Conf と標準ユーザ グループがこの機能の admin ページに追加されています。アプリケーションは、JTAPI クライアント 7.1.2 以上を使用していることを前提として、この新しい権限がアプリケーション ユーザに関連付けられている場合にのみ、これらのデバイスを制御できます。デフォルトでこれらのデバイスは制限付きとして表示され、アプリケーションがこの機能を処理できるようにアップグレードされ、新しい権限を関連付けている場合にのみ、これらのデバイスを制御できます。アプリケーションで古い JTAPI クライアントを使用している場合、デバイスは制限されませんが、アプリケーションがこれらのデバイス（この機能の手動での起動をサポートしている）を監視しようとすると、JTAPI は例外をスローし、その時点からこれらのデバイスを制限付きとしてマークします。

この機能は、ユーザの操作性を拡張するために設計されているため、すべての Cisco Unified JTAPI アプリケーションでこの機能を評価して、サポートし、必要に応じて、古い動作と新しい動作を処理するコード ロジックでアップグレードすることを強くお勧めします。

メッセージ受信インジケータの拡張

メッセージ受信インジケータ（MWI）機能の拡張により、アプリケーションは、拡張メッセージ受信カウントをサポートする電話に次のメッセージ カウントを表示することができます。

- 新しい音声メッセージ（標準および高い優先度を含む）の合計件数
- 以前の音声メッセージ（標準および高い優先度を含む）の合計件数
- 高い優先度の新しい音声メッセージの件数
- 高い優先度の以前の音声メッセージの件数
- 新しい FAX メッセージ（標準および高い優先度を含む）の合計件数
- 古い FAX メッセージ（標準および高い優先度を含む）の合計件数
- 高い優先度の新しい FAX メッセージの件数
- 高い優先度の古い FAX メッセージの件数

拡張 MWI メッセージ概要を提供するために、CiscoAddress JTAPI 拡張として、2 つの新しい API が追加されています。既存の setMessageWaiting API と同様に、一方の API では、監視対象のアドレスに対して概要を設定できます。他方の API では、監視対象のアドレスの設定済みのコーリング サーチスペースに定義されている、監視対象のアドレスで到達可能な任意のアドレスにメッセージ概要を設定することができます。

これらの新しい API は拡張メッセージ カウントをサポートしない電話タイプで使用することもできます。これらの API をサポートされない電話で使用した場合、既存の setMessageWaiting メソッドと同じように動作します。つまり、メッセージ受信インジケータ ランプが点灯または消灯するだけで、カウントは表示されません。

インターフェイスの変更

「CiscoAddress」（P.6-37）を参照してください。

メッセージ シーケンス

「拡張された MWI の使用例」（P.A-214）を参照してください。

下位互換性

この機能は下位互換性があります。既存の setMessageWaiting API は変更されません。新しい拡張 MWI 機能を使用する必要がないアプリケーションでは、MWI ランプを設定するために、これらの API を引き続き使用することができます。

パーク監視と Assisted DPark のサポート

この機能は、パーク要求を呼び出すアプリケーションに、新しいパークの復帰動作を提供します。現在、パークの復帰タイマーが切れると、コールがパーク元のアドレスに戻されます。新しい動作によって、パーク モニタリング復帰タイマーが切れても、コールはパーク DN にパークされたままになります。

この機能により、パーク元のアドレスでパークされているコールのステータス監視も可能になります。新しい電話（6900 シリーズ以降）で既存の CiscoConnection.park() JTAPI API を使用するか、電話自体から直接、コールをパークすると、Cisco Unified JTAPI はパークされているコールの現在のステータスを含む新しいイベント CiscoAddrParkStatusEv を配信します。アプリケーションはパーク元のアドレスに AddressObserver を追加し、このイベントを受け取るフィルタを有効にする必要があります。コールがパークされた後にアプリケーションがオブザーバを追加した場合、CAUSE_SNAPSHOT でイベントが配信されます。新しいイベントのパーク ステータスは次のいずれかである可能性があります。

- Parked：アプリケーションのユーザによってコールがパークされたことを示します。
- Reminder：パークされているコールのパーク モニタリング復帰タイマーが切れたことを示します。
- Retrieved：以前にパークされたコールが取得されたことを示します。
- Abandoned：以前にパークされたコールが取得を待つ間に切断されたことを示します。
- Forwarded：Park Monitoring Forward-No-retrieve タイマーが切れたときに、パークされているコールが設定されている Forwarded No Retrieve Destination に転送されたことを示します。

原因が CAUSE_SNAPSHOT の場合、パーク ステータスは、Parked または Reminder 状態のいずれかになります。

電話がこれらの通知が対象となります。つまり、コールをパークしているデバイスのみがこれらの通知を認識できます（パーク元のデバイスと回線を共有する他のデバイスは同様の通知は受け取りません）。Cisco Unified JTAPI に、これを管理するための CiscoAddrParkStatusEv の getTerminal() インターフェイスが追加されています。このインターフェイスは、これらの通知を受け取ったアドレスの端末を返し、この端末はコールをパークした端末です。

さらに、Cisco Unified JTAPI はこの新しいイベントで、アプリケーションに CiscoCallID も提供します。アプリケーションはこれを使用して、コール オブジェクトを取得できます。ただし、このイベントを受け取った時点で、プロバイダーのドメインにコールが存在しない場合、CiscoCallID.getCall() は Null 値を返すことがあります。

Cisco Unified JTAPI は、アプリケーションへの新しいイベント通知を制御またはフィルタする新しいインターフェイス CiscoAddrEvFilter を提供しています。アプリケーションは、CiscoAddrEvFilter インターフェイスの API getCiscoAddrParkStatusEvFilter() および setCiscoAddrParkStatusEvFilter() によってフィルタ値を取得または設定できます。CiscoAddrEvFilter インターフェイスでフィルタの値を取得し、設定するための 2 つの新しいメソッド getFilter() と setFilter() も CiscoAddress に提供されています。フィルタが有効にされており、CiscoAddress で setFilter() が呼び出されている場合にのみ、アプリケーションは新しいイベント通知 CiscoAddrParkStatusEv を受け取ります。下位互換性を維持するために、デフォルトで、CiscoAddrParkStatusEvFilter のフィルタ値は False にされています。

コールがパークされると、パーク モニタリング復帰タイマーが開始し、期限が切れると Reminder が送られこの後、Park Monitoring Forward No Retrieve Timer が開始します。このタイマーが切れ、未取得時のパーク モニタリング転送の接続先が設定されている場合、コールがその宛先に転送されます。転送先で Connection が作成されると、Connection イベントで新しい CiscoFeatureReason FORWARD_NO_RETRIEVE が配信されます。Forward No Retrieve Destination が設定されていない場合、パークの復帰が発生した場合と同じ理由（CiscoFeatureReason.PARKREMINDER）で、パークコールが元の DN に返送されます。

アプリケーションが CiscoAddress.getAddressCallInfo(Terminal term) を呼び出したときに、返される CiscAddressCallInfo が拡張され、パークされているコール数が含まれるようになりました。これはパークされているコール数を返します。Cisco Unified IP Phone 7900 シリーズ（SIP/SCCP）は、このアドレスによってパークされたコールがある場合でも 0 の値を返します。

この機能は、新しい電話（6900 シリーズ以降）でコールがパークされている場合にのみ適用します。Cisco Unified IP Phone 7900 Series（SIP/ SCCP）でコールがパークされた場合、引き続き既存の動作が行われます。新しい Cisco Unified IP Phone でコールをパークし、Cisco Unified IP Phone 7900 シリーズ（SIP）と回線を共有している場合、新しいパーク監視拡張が機能します。ただし、Cisco Unified IP Phone 7900 シリーズ（SIP または SCCP）でパークした場合、アプリケーションがこれらの回線を監視している場合でも、すべての電話で古いパーク動作が行われます。

ユーザは必要に応じて、パーク モニタリング復帰タイマーを 0 に設定し、Park Monitoring Forward No Retrieve タイマーを既存の Park Reversion Duration タイマーに設定して、新しい Cisco Unified IP Phone（Forward No Retrieve Destination が設定されていない場合）で古い動作を得ることができます。ただし、イベント通知は制御できません。

Unified CM サービス パラメータ ページで、先述のタイマーを設定できます。これらは、Cisco Unified IP Phone の 将来のモデルの SIP バージョンにのみ適用されます。

パーク モニタリング復帰タイマー：このタイマーはコールがパークされるとただちに開始します。これは、ユーザにパークされているコールがあることを通知するまでのコールがパークされている時間です。この範囲は 0 ～ 1200 秒で、デフォルト値は 60 秒です。

Park Monitoring Periodic Reversion タイマー：ユーザがパークされているコールについて通知される頻度。この範囲は 0 ～ 1200 秒で、デフォルト値は 30 秒です。

Park Monitoring Forward No Retrieve タイマー：このタイマーはパーク モニタリング復帰タイマーが期限切れになると開始します。これは、パーク先がパーク元の未取得時のパーク モニタリング転送（FNR）の宛先にリダイレクトされるまでのパーク リマインダ通知が再生される秒数です。この範囲は 30 ～ 1200 秒で、デフォルト値は 300 秒です。

未取得時のパーク モニタリング転送の接続先（Park Monitoring Forward No Retrieve Destination）：Unified CM 回線ページ設定の回線ページで設定できます。

Assisted DPark は、電話で DPark 操作を実行するための代替の 1 手順の方法を提供します。ユーザが新しい電話から Assisted DPark を実行し、アプリケーションがパークされている相手を監視している場合に、Cisco Unified JTAPI は DPark DN の接続イベント（ConnCreatedEv、ConnInProgressEv、および CallCtlConnQueuedEv）に理由 CiscoFeatureReason.REASON_REFER を提供します。現在、DPark が実行されると、アプリケーションは CiscoFeatureReason.REASON_TRANSFER の接続イベントを受け取ります。

インターフェイスの変更

「CiscoAddrParkStatusEv」（P.6-59）を参照してください。

メッセージ シーケンス

「パーク モニタリング サポート」（P.A-261）を参照してください。

下位互換性

パーク監視拡張と Assisted DPark のサポートは下位互換性があります。

新しいパーク復帰動作により、ユーザの操作性が向上し、パークされているコールが可能な限り長く取得可能になります。さらに、配信される新しいイベントによって、パークされているコールのステータスを監視できることによって、パーク機能の操作性も向上しています。

アプリケーションは条件によってフィルタを有効/無効にして、CiscoAddeEvFilter の setCiscoAddrParkStatusEvFilter() API を介してイベントを受け取ることができます。デフォルトでこのフィルタは無効にされているため、下位互換性が維持されます。

アプリケーションで 7.1.2 より古い JTAPI クライアントを使用している場合、デバイスは制限されませんが、アプリケーションがこれらのデバイス（この機能の手動での起動をサポートしている）を監視しようとすると、JTAPI は例外をスローし、その時点からこれらのデバイスを制限付きとしてマークします。

論理パーティション化

管理者はこの機能を使用して、地理的位置を設定し、別の地理的位置の VoIP フォンまたは VoIP PSTN ゲートウェイに直接接続された PSTN ゲートウェイを通過するコールを制限できます。この機能により、単一回線のアナログ電話を使用でき、Telecom Regulatory Authority of India（TRAI）規制との互換性を維持できます。

この機能は、Logical Partitioning Enabled サービス パラメータを使用して無効にできます。これはデフォルトで無効にされています。

インターフェイスの変更

「CiscoJtapiException」（P.6-150）を参照してください。

メッセージ シーケンス

「論理パーティション設定機能の使用例」（P.A-293）を参照してください。

下位互換性

この機能は下位互換性があります。

Component Updater

Component Updater インターフェイスは、アプリケーションで更新ログの場所を指定できるように拡張されました。現在、更新ログは、アプリケーションと同じディレクトリに作成されます。この拡張により、アプリケーションはトレースの場所を指定できます。

インターフェイスの変更

「ComponentUpdater」（P.6-348）を参照してください。

メッセージ シーケンス

「ComponentUpdater 拡張の使用例」（P.A-296）を参照してください。

下位互換性

この機能は下位互換性があります。

Cisco Unified IP Phone 6900 シリーズのサポート

この機能により、Cisco Unified JTAPI アプリケーションは、ロールオーバー モードを有効にして端末を制御できます。ロールオーバー モードでは、同じ DN だが異なるパーティション内の、または異なる DN の複数のアドレスで端末を設定できます。ロールオーバー モードを有効にすると、端末で次に使用可能なアドレスでコンサルト コールを作成できます。Cisco Unified IP Phone 6900 シリーズはロールオーバー モードで設定できます。

ロールオーバー モードをサポートしている電話の新しい権限 Standard CTI Allow Control も導入されており、アプリケーションでロールオーバーが有効にされた端末を制御できます。この新しい動作をサポートするアプリケーションは、コンサルト コールが異なるアドレスで作成される場合に、それらのアプリケーションまたはエンド ユーザにこの権限を含める必要があります。そうしないと、ロールオーバー モードで設定されたすべての端末が制限され、addObserver() 要求に例外がスローされます。

この動作をサポートするアプリケーションは、端末にコール オブザーバを追加するか、端末上のすべてのアドレスにコール オブザーバを追加する必要があります。コンサルト コールは次に使用可能なアドレスで作成されるため、すべてのアドレスにコール オブザーバが追加されていない場合、コンサルト 要求に例外がスローされます。

アプリケーションから会議を正常に実行するために、Cisco Unified IP Phone 6900 シリーズで回線をまたいで参加（Join Across Lines）が有効にされている必要があります。

インターフェイスの変更

「CiscoProviderCapabilities」（P.6-211）および「CiscoProviderCapabilityChangedEv」（P.6-213）を参照してください。

メッセージ シーケンス

「Cisco Unified IP Phone 6900 シリーズのサポート」（P.A-297）を参照してください。

下位互換性

この機能は下位互換性があります。

Join Across Lines with Conference 機能拡張

Join Across Lines with Conference 機能は、次のように拡張されました。

- 異なる回線間での会議のチェーニング。たとえば、アプリケーションから、同じ端末上でそれぞれアドレスの異なる 2 つの電話会議を開催できます。
- アプリケーションは、同じ端末上のアドレスが異なる 2 つのコールで会議を開催できます。
- コントローラ以外を使用して会議に参加者を追加できます。

（注）　回線をまたいで参加（Join Across Lines）機能を無効にするには、Join Across Lines Policy サービス パラメータをオフにします。会議のチェーニング機能と、コントローラ以外が参加者を会議に追加できる機能を無効にするには、Advanced Ad Hoc Conference Enabled および Non-linear Ad Hoc Conference Linking Enabled サービス パラメータを無効にします。

アプリケーションから会議要求が発行されたが、選択したコールとアクティブなコールが会議要求に含まれていない場合、次のような動作が発生します。また、この動作は、ユーザが選択したコールが、会議要求には含まれていないが、結果として開催された会議には含まれていた場合にも適用されます。

- 会議が端末の任意のアドレスのコールによって呼び出された場合は、その端末上のアクティブ コールは、必ず結果として開催された会議に追加されます。B1 と B2 アドレスは同じ端末上にあるとします。
  - A --> B1- GC1
  - C --> B1- GC2
  - D --> B2- GC3（アクティブ コール）

  アプリケーションが GC1.conference (GC2) を呼び出した場合、会議要求に D が含まれていなくても、A-B1-C-D（GC1）の会議が開催されます。

  会議が端末の任意の回線上のコールで開催された場合、その端末上のアクティブな電話会議が、結果として開催される会議に追加されます。この場合、アクティブな電話会議が、存続されるコールとなります（アプリケーションにより指定されたプライマリ コールが電話会議ではない場合）。

  この例では、会議の動作が開始されると、アプリケーションが指定したプライマリ コールはクリアされます。アプリケーションが指定したプライマリ コールが、結果として開催された会議に参加できない場合があります。この場合、会議が終了してもプライマリ コールはクリアされません。
- B1 と B2 のアドレスが同じ端末上にあり、conf1 は、B1 をコントローラとする A-B1-C の会議との電話会議であるとします。
  - B1 --> D – GC1（保留）
  - conf1 – GC2（アクティブ コール）
  - B2 --> E – GC3（保留）

  アプリケーションは GC1.conference(GC2, GC3) を呼び出します。これにより、GC2 が存続するコールとなる A-B1-C-D-E の会議が開催されます。アプリケーションは GC1 をプライマリ コールに指定していましたが、会議の終了後は GC1 は存続しません。

この動作は、通常のコントローラで開催される定例会議にも適用されます。A、B、C、および D が異なる端末上に存在するとします。

- A --> B - GC1
- C --> - GC2
- D --> - GC3（アクティブ コール）

  アプリケーションから GC1.conference (GC2) が要求されます。これにより、GC1 で A-B-C-D の会議が開催されます。D とのダイレクト コールが会議要求に含まれていなくても、D は会議に参加できます。

インターフェイスの変更

インターフェイスの変更はありません。現在のインターフェイスを使用して、同じ端末上の異なるアドレスで電話会議を開催できます。

メッセージ シーケンス

「拡張された回線をまたいで参加（Join Across Lines）」（P.A-11）

下位互換性

この機能は下位互換性があります。

ロケール インフラストラクチャの変更

リリース 7.0(1) から、Cisco Unified JTAPI クライアントのインストールで現在サポートされている言語が削除されます。Cisco Unified JTAPI クライアントのインストールでサポートされている言語は英語だけです。また、Cisco Unified Communications Manager サーバから、JTAPI Preference アプリケーションのロケールを動的に更新する機能も追加されます。JTAPI Preference アプリケーションでは、以前のリリースでサポートされていたすべての言語も継続的にサポートされます。新しい言語の追加およびロケール ファイルの更新のサポートも追加されています。

以前のリリースでは、Cisco Unified JTAPI クライアント インストールと JTAPI Preferences アプリケーションはビルド時にローカライズされ、新しい言語のサポートや既存の言語のロケールの更新に対するサポートは追加されていませんでした。JTAPI クライアントのロケールの更新は、Cisco Unified Communications Manager のメンテナンス リリースで実行されていました。ロケールインフラストラクチャの変更により、JTAPI Preferences アプリケーションのロケール ファイルを動的に更新する機能が追加され、JTAPI クライアントのインストールは、英語でだけ実行可能になります。

JTAPI クライアントのインストールには、Cisco Unified Communications Manager TFTP サーバの IP アドレスが必要です。Preferences アプリケーションのロケール ファイルのダウンロードには、TFTP IP アドレスを使用します。TFTP IP アドレスが入力されていない場合や、入力された IP アドレスに誤りがある場合は、Preferences アプリケーションには英語だけが表示されます。今後、新しいロケールの更新が可能になった場合は、JTAPI Preferences アプリケーションから、利用できる更新と最新のロケール ファイルがユーザに通知されます。

インターフェイスの変更

インターフェイスの変更はありません。

メッセージ シーケンス

「ロケール インフラストラクチャ変更シナリオ」（P.A-147）

下位互換性

この機能は JTAPI アプリケーションの観点からは下位互換性がありますが、JTAPI クライアント インストールの観点からすると、この機能により現在サポートされている言語が削除されます。そのため、この点においては、この機能は下位互換性はありません。

Do Not Disturb–Reject

Do Not Disturb–Reject（DND–R）は、既存の Do Not Disturb（DND; サイレント）機能の拡張機能です。これまで、Cisco Unified Communications Manager と JTAPI では、呼出音をオフにする DND（サイレント）機能だけがサポートされていました。ユーザは DND–Reject を使用してコールを拒否できます。DND–R は、Cisco Unified Communications Manager Administration の電話の設定ウィンドウ、または電話プロファイルの設定ウィンドウから設定できます。

DND–R が有効である場合、コール拒否が有効な端末にコールは表示されません。つまり、そのエンドポイントには、着信コールを示す音声または視覚的な表示は行われません。DND–R を有効にするには、DND のステータスを true に設定し、DND オプションを [Call Reject] に設定します。

FeaturePriority により DND が上書きされます。次のいずれかの値が返されます。

- 1: Normal
- 2: Urgent
- 3: Emergency

このリリースでは、CiscoCall の connect() API に FeaturePriority が導入されました。selectRoute() と redirect() API の FeaturePriority は、以前のリリースからすでにサポートされています。connect() API で機能プライオリティが EMERGENCY に指定されており、宛先の端末で DND–R が有効である場合、宛先端末にコールがあると呼出音が鳴り、DND–R 設定が上書きされます。

端末で DND–R が有効であり、インターコム コールを受信すると、DND–R 設定は上書きされ、コールが表示されます。これは、インターコム コールの機能プライオリティは、必ず 2（URGENT）に設定されているためです。

共用回線以外では、A が B にコールし、端末 B の DND–R が有効である場合、原因が USER_BUSY の CallCtlConnFailedEv が A に配信されます。共有 DN を持つすべての端末上で DND–R が有効である場合、ユーザは同じ動作を受信します。

共用回線の 1 つ以上の端末で DND–R が無効であり、その共用回線にコールが発信されると、Cisco Unified JTAPI により、DND–R が有効に設定されている端末に対して TermConnPassiveEv と CallCtlTermConnInUseEv が配信されます（コールの機能プライオリティが NORMAL に設定されていること仮定した場合）。端末上で DND–R が無効である場合は、コール時に、TermConnPassiveEv と CallCtlTermConnBridgedEv が配信されます。

Cisco Unified Communications Manager Administration の電話の設定ウィンドウ、または電話プロファイルの設定ウィンドウで DND（サイレント）オプションが変更されると、新しいイベントである CiscoTermDNDOptionChangedEv が、端末オブザーバに送信されます。

デフォルトの DND オプションは [Ringer–off] で、ルート ポイントで DND はサポートされません。

インターフェイスの変更

「CiscoTermDNDStatusChangedEv」（P.6-301）、「CiscoCall」（P.6-70）、「CiscoTermEvFilter」（P.6-304）

メッセージ シーケンス

「DND–R」（P.A-118）

下位互換性

この機能は下位互換性があります。この機能を設定すると、アプリケーションは新しいイベントを受信します。新しいイベントは、TerminalEventFilter インターフェイスを使用してフィルタされます（CiscoTermEvFilter）。デフォルトでは、このフィルタは無効に設定されており、新しいイベントは配信されません。

発信側の正規化

発信側の正規化（CPN）は拡張された機能です。この機能を使用すると、着信コール番号が変換または正規化され、（国番号、状態コード、番号タイプ）を含む E.164 形式に変換されます。番号タイプフィールドには、加入者、国内、海外、または不明が表示されます。番号タイプは会議のシナリオではサポートされません。

インターフェイスの変更

この機能により、CiscoCall の新しいメソッド getGlobalizedCallingParty() と、CiscoPartyInfo の新しいメソッド getNumberType() が導入されました。詳細については、「CiscoCall」（P.6-70）と「CiscoPartyInfo」（P.6-196）を参照してください。

メッセージ シーケンス

「発信側の正規化」（P.A-148）

下位互換性

この機能は下位互換性があります。

クリック ツー会議

クリック ツー会議機能では、Instant Messenger（IM）などのアプリケーションで会議に通話者を追加できる、SIP トランクのインターフェイスが提供されます。ユーザは、このようなアプリケーションを使用して、他のユーザを会議に追加したり、ユーザを削除することが可能です。クリック ツー会議機能を使用してユーザを会議に追加すると、ターゲット アドレスにコールが提供されます。ターゲット アドレスの 1 つのアドレスだけが、この最初のコール上に作成されます。これにより、このコールは会議に追加され、ターゲット アドレスに新しい callID が作成されます。さらに、このコールの他のアドレスへの接続は新しいコール上に作成されます。

ここでは、クリック ツー会議機能を使用してアドレスが会議に追加されたときに、その対話を処理するために Cisco Unified JTAPI のインターフェイスの変更について説明します。クリック ツー会議機能を使用すると、コンサルト コールは行われず、Cisco Unified JTAPI アプリケーションでは CiscoConferenceStartEv または CiscoConferenceEndEv が受信されません。

この機能を無効にするには、CallManager のサービス パラメータ「ENABLE CLICK TO CONFERENCE」をオフにします。

インターフェイスの変更

「CiscoFeatureReason」（P.6-143）

メッセージ シーケンス

「クリック ツー会議」（P.A-156）

下位互換性

この機能は下位互換性があります。この機能を設定していない場合、または使用していない場合は、Cisco Unified JTAPI アプリケーションに変化はありません。

エクステンション モビリティのユーザ名ログイン

エクステンションモビリティのログイン ユーザ名機能を使用すると、アプリケーションは CiscoTerminal に提供された API からエクステンションモビリティのログイン ユーザ名を取得できます。

インターフェイスの変更

「CiscoTerminal」（P.6-307）

メッセージ シーケンス

「エクステンション モビリティ ログイン ユーザ名」（P.A-182）

Java ソケット接続のタイムアウト

Java ソケット接続のタイムアウト拡張機能を使用すると、Cisco Unified JTAPI 仕様を使用して秒単位でタイムアウトを設定できるので、プライマリ CTI マネージャでは、CTIManager への接続遅延を防ぐことができます。デフォルトは 15 秒です。

デフォルトの 15 秒がアプリケーションに受け入れられない場合は、JAVA API のデフォルトである 0 が、通常の JAVA ソケット接続 API に設定されます。

値の範囲は、5 ～ 180 秒です。デフォルトのゼロを設定すると、JAVA のソケット接続はタイムアウトしません。

インターフェイスの変更

「CiscoJtapiProperties」（P.6-164）

メッセージ シーケンス

「CiscoJtapiProperties」（P.A-184）

下位互換性

この機能は下位互換性があります。

コーリングサーチスペースおよび機能プライオリティを使用した selectRoute()

selectRoute() には、機能プライオリティとコーリング サーチ スペースを配列で持っています。この API では、選択したルートごとに、異なる機能プライオリティとコーリング サーチ スペースが柔軟に提供されます。

インターフェイスの変更

「CiscoRouteSession」（P.6-224）

メッセージ シーケンス

「コーリングサーチスペースおよび機能プライオリティを使用した selectRoute()」（P.A-181）

下位互換性

この機能は下位互換性があります。selectRoute() API は、機能はそのままで、オーバーロードされた selectRoute() API と相互運用されます。

コール ピックアップ

コール ピックアップを使用すると、コール ピックアップ グループ内のデバイスは、グループ内の他の電話が鳴っているという警告を受け、そのコールをピックアップできます。つまり、呼出音が鳴っている元のデバイスからコールを取得できます。

この機能拡張では、Cisco Unified JTAPI はコール ピックアップ イベントが発生した端末とアドレスの監視をサポートします。ただし、API がアプリケーションからコール ピックアップを起動できません。これは、Auto Call Pickup サービス パラメータが有効に設定されているときの、API によるコール ピックアップ イベントの処理方法に対する小さな変更点です。このパラメータが無効に設定されているときに、ユーザが [Pickup（ピックアップ）] ソフトキーを押すと、そのユーザの電話機の呼出音が鳴りだし、その後はすべて通常通りの処置が可能です。「Auto Call Pickup」が有効に設定されている場合に、このソフトキーを押すと、電話機の呼出音は鳴らず、そのままコールをピックアップできます。

インターフェイスの変更

インターフェイスの変更はありません。

メッセージ シーケンス

「コール ピックアップ」（P.A-168）

下位互換性

この機能は下位互換性があります。

証明書ダウンロードの API 機能拡張

現在、Cisco Unified JTAPI の証明書のダウンロード API にはセキュリティ上の問題があり、この問題を解決するために、新しい証明書のダウンロード API を提供しています。新しい API では、アプリケーションが証明書のパス フレーズを指定する必要があります。証明書のパス フレーズは、クライアント / サーバ証明書が格納された Java キー ストアの暗号化に使用されます。

古い証明書のダウンロード API の使用は推奨しませんが、この API は、アプリケーションの下位互換性の問題を避けるために、まだ残されています。アプリケーションを新しい API に移行することを強くお勧めします。

また、Cisco Unified JTAPI では、新しい API の deleteCertificate() と deleteSecurityPropertyForInstance() も提供されます。これらをアプリケーションで使用すると、すでにインストールされている証明書を削除できます。証明書の Java キー ストアのパス フレーズを変更する場合は、この API を使用して古い証明書を削除し、新しい証明書をアップロードする必要があります。

JTAPIPreferences UI の [セキュリティ] タブの拡張により、[証明書の削除] と [証明書の更新] という 2 つの新しいボタンが追加されました。[証明書の削除] ボタンを使用すると、ユーザは目的のユーザ名またはインスタンス ID の証明書を削除できます。[証明書の更新] ボタンを使用すると、ユーザは

CAPF サーバから証明書をアップロードできます。証明書のアップロードが成功すると、[証明書の更新] ボックスが更新されて更新済みであることが示され、認証文字列と証明書パス フレーズがクリアされます。証明書の更新操作が失敗した場合は、証明書が前に更新されていない限り、証明書ボックスに [未更新] のステータスが示されます。ユーザとアプリケーションは、証明書の更新を行うたびに証明書パス フレーズを提供する必要があり、Cisco Unified JTAPI ではどのような環境下でもセキュリティ上の理由で証明書パス フレーズは保存されません。パス フレーズのセキュリティを確保し、必要な場合には API を通じてパス フレーズを提供することはアプリケーション側の責任です。

下位互換性

この機能は下位互換性があります。

回線をまたいで参加（Join Across Lines）

このバージョンでこの機能を使用すると、アプリケーションは、同じ端末の異なるアドレス上にある 2 つのコールが参加する会議を開くことができます。また、この機能では、アプリケーションはコントローラ以外を使用して、参加者を会議に追加できます。回線をまたいで参加（Join Across Lines）機能は、SIP を実行する CTI 対応の電話でもサポートされます。

回線をまたいで参加（Join Across Lines）機能は、Join Across Lines Policy サービス パラメータをオフにすることで無効にできます。また、会議のチェーニング機能と、コントローラ以外が参加者を会議に追加できる機能は、「Advanced Ad Hoc Conference Enabled」と「Non-linear Ad Hoc Conference Linking Enabled」サービス パラメータを無効にすることで無効にできます。

インターフェイスの変更

この機能には、インターフェイスの変更はありません。アプリケーションは、現在の会議用インターフェイスを使用して、同じ端末上の異なるアドレスが参加する電話会議を開くことができます。

下位互換性

この機能は下位互換性があります。

エクステンション モビリティのインターコム サポート

Cisco Unified Communication Manager のリリース 6.0(1) では、インターコム機能のサポートが追加されました。インターコム機能には、着信側が音声による一方向の自動応答を受け取る必要があるため、インターコムに共有アドレスを設定できません。ユーザがエクステンション モビリティ（EM）プロファイルを使用してログインした場合、インターコムと共有アドレスとなる可能性があります。そのため、これまでエクステンション モビリティはインターコムでサポートされていませんでした。エクステンション モビリティは幅広く利用されるため、この 1 つの宛先を共有できないというインターコム アドレスの性質は維持すると同時に、エクステンション モビリティによるインターコムのサポートというニーズに対処しています。

この機能には、デフォルトの端末で設定されたインターコム アドレスが必要です。この機能を使用すると、インターコム アドレスを EM プロファイル上で設定できます。EM ユーザが端末に、インターコム アドレスで設定された EM プロファイルを使用してログインを行うと、インターコム アドレスのデフォルトの端末がユーザがログインした端末と同じ場合に限り、インターコム アドレスが使用可能になります。インターコム アドレスが端末に設定されているが、この端末はインターコム アドレスのデフォルトの端末ではない場合、インターコム アドレスは端末に表示されません。この端末が Cisco Unified JTAPI アプリケーションの制御リストに設定されている場合、JTAPI はプロバイダーのドメイン内にインターコム アドレスを作成しません。Cisco Unified JTAPI の観点からは、この機能をサポー

トするための新しいインターフェイスや変更はありません。ただし、この機能によって、インターコム機能がインターコム アドレス上で動作しなくなるいくつかの移行シナリオが発生します。使用例を参照してください。

下位互換性

この機能は下位互換性があります。

録音とサイレント モニタリング

この機能を使用すると、アプリケーションによるコールの録音およびサイレント モニタリングが可能になります。この場合の発信者は、モニタリングのターゲットか録音開始側にコールするか、そこからコールを受信する側のエンド ポイントを表します。モニタリングのターゲットはモニタリングの対象側（エージェント）で、モニタリング側はモニタリングの開始側（スーパーバイザ）です。

録音機能により、アプリケーションは任意のモニタリング対象アドレスの会話を録音できます。3 つの録音設定があります。

- 録音なし
- 自動録音

  システムによって録音セッションが始められ、コールが接続状態になると、設定された録音デバイスにメディアがストリーミングされます。
- アプリケーション制御による録音

  アドレスに対してアプリケーション制御による録音が設定されると、アプリケーションは録音を開始および停止できます。アプリケーションが録音を開始する前に、コールが接続状態で存在している必要があります。

管理者は、回線上にいずれかの録音設定を設定できます。

自動録音またはアプリケーションの要求によりトリガーされた録音セッションが確立されると、システムから録音デバイスに、録音開始側からの音声および発信者からの音声の 2 つの音声ストリームが送信されます。

サイレント モニタリング機能により、アプリケーションは他の 2 者間の会話をライブで傍受できます。モニタリングの開始側は、モニタリングのターゲットおよび発信者のいずれとも通話できません。

モニタリングは、アプリケーションの要求によってだけ開始されます。アプリケーションは、モニタリングする各コールに対し、そのつど、モニタリング要求を送信する必要があります。接続状態にあるコールだけをモニタリングできます。モニタリングの要求が確立されると、モニタリングのターゲットと発信者間の音声ストリームがモニタリングの開始側にストリーミングされます。次のような場合、モニタリングのターゲットはトーンを受信します。

- モニタリングのターゲットがトーンを受信するように設定されている場合、または
- アプリケーションにより、モニタリングの開始時にトーンを鳴らすように要求されている場合

録音とサイレント モニタリングの機能では、セキュリティ保護されているコールはサポートしていません。モニタリングのターゲットと録音開始側の両方で、セキュリティを無効にする必要があります。

Cisco Unified Communications Manager Administration の 2 つのユーザ グループで、録音とサイレント モニタリングの機能がサポートされています。アプリケーションでは、それぞれ Standard CTI Allow Call Recording および Standard CTI Allow Call Monitor ユーザ グループに所属している場合に、コールの録音とモニタリングができます。録音とモニタリングに関するイベントは、すべてのコール オブザーバに配信されます。アプリケーションがこれら 2 つの特殊なグループに所属していない場合でも、これらのイベントを受け取ります。

「Monitor」および「Recording」は予約語なので、システム内の回線の表示名としては設定できません。これ以外の予約語には、「Conference」、「Park Number」、「Barge」および「CBarge」があります。

モニタリング セッションが確立されると、モニタリング開始側の端末オブザーバが Cisco RTP イベントを受信します。サイレント モニタリングのコールではメディアが一方向にしか流れませんが、getMediaConnectionMode() は CiscoMediaConnectionMode.RECEIVE_ONLY の代わりに CiscoMediaConnectionMode.TRANSMIT_AND_RECEIVE を返します。CTIPort がモニタリング開始側として使用されている場合、アプリケーションでは CiscoMediaOpenLogicalChannelEv から同じ動作を受信する可能性があることを考慮する必要があります。

モニタリング コール（モニタリング開始側によって使用されているコール）を会議にした場合は、最後のコールはモニタリングのターゲットとの接続がありません。モニタリング開始側が別の通話者をモニタリング コールの会議に追加すると、両者にモニタリング ターゲットと発信者との間の音声が聞こえます。

次のインターフェイスは TermConnEv を拡張し、コール オブサーバに配信されます。共用回線の場合、通話中の TerminalConnection のアドレスまたは端末上のコール オブザーバに、これらのイベントが配信されます。端末の接続が INUSE または BRIDGED 状態だけの場合、アプリケーションはイベントを受信しません。

CiscoTermConnRecordingStartedEv

`CiscoTermConnRecordingStartedEv`
録音の開始を示し、録音開始側のコール オブザーバに配信されます。自動録音設定またはアプリケーションの要求によって録音がトリガーされます。

CiscoTermConnRecordingEndEv

`CiscoTermConnRecordingEndEv`
録音の終了を示し、録音開始側に配信されます。

CiscoTermConnMonitoringStartEv

`CiscoTermConnMonitoringStartEv`
モニタリングの開始を示し、モニタリング ターゲットのコール オブザーバに配信されます。このイベントに対して getMonitorType() を使用すると、モニタリングのタイプを返します。

CiscoTermConnMonitoringEndEv

`CiscoTermConnMonitoringEndEv`
モニタリングの終了を示し、モニタリング ターゲットのコール オブザーバに配信されます。

CiscoTermConnMonitorInitiatorInfoEv

モニタリングの開始側の情報が公開され、モニタリング ターゲットのコール オブザーバに配信されます。このインターフェイスには 1 つのメソッドがあります。

`CiscoMonitorInitiatorInfo getCiscoMonitorInitiatorInfo ()`
モニタリングの開始側の端末名およびアドレスを公開する CiscoMonitorInitiatorInfo を返します。

CiscoTermConnMonitorTargetInfoEv

モニタリングのターゲットの情報が公開され、モニタリング ターゲットのコール オブザーバに配信されます。このインターフェイスには 1 つのメソッドがあります。

`CiscoMonitorInitiatorInfo getCiscoMonitorTargetInfo()`
モニタリングのターゲットの端末名およびアドレスを公開する CiscoMonitorInitiatorInfo を返します。

2 つの新しいエラー コードにより、モニタリングの失敗がアプリケーションに通知されます。

- CTIERR_PRIMARY_CALL_INVALID は、コールがアイドル状態になるか転送されたことが原因でモニタリングの要求が失敗し、例外が発生した場合に、CiscoException.getErrorCode() によって返されます。
- CTIERR_PRIMARY_CALL_STATE_INVALID は、モニタリングを開始できない状態にコールが移行したことが原因でモニタリングの要求が失敗した場合に返されます。

このリリースでは、新しい AddressType、MONITORING_TARGET が導入されています。JTAPI では、モニタリングのターゲット アドレスとして、このタイプのアドレスに Connection を作成します。この値は、CiscoAddress.getType() によって返されます。

下位互換性

この機能は下位互換性があります。この機能が設定されて、アプリケーション制御によるアドレスに対して使用されない限り、アプリケーションが新しいイベントを受信することはありません。この機能を有効にするには、Cisco Unified JTAPI アプリケーション ユーザを Standard CTI Allow Call Recording および Standard CTI Allow Call Monitor ユーザ グループに追加します。

これらのインターフェイスの変更に関する詳細については、次のトピックを参照してください。

- CiscoJtapiException
- CiscoAddress
- CiscoCall
- CiscoMediaTerminal
- CiscoMonitorTargetInfo
- CiscoMonitorInitiatorInfo
- CiscoProvider
- CiscoProviderCapabilities
- CiscoProviderCapabilityChangedEv
- CiscoProviderObserver
- CiscoRecorderInfo
- CiscoTerminalConnection
- CiscoTermConnMonitorInitiatorInfoEv
- CiscoTermConnMonitorTargetInfoEv
- CiscoTermConnRecordingTargetInfoEv

インターコム

インターコム機能を使用すると、あるユーザが別のユーザにコールした際に、着信側がビジーであるかアイドルであるかにかかわらず、コールが発信側から着信側への一方向メディアによって自動的に応答されます。着信側は、電話機のディスプレイの応答ソフトキー（無印のキー）を押すか、または TerminalConnection で提供されている join() JTAPI API を起動して発信者と会話を始めることができます。電話機上で特に設定されたインターコム アドレスだけがインターコム コールを開始できます。電話機に設定されたインターコム アドレスに対し、Cisco Unified JTAPI によって CiscoIntercomAddress という新しいタイプのアドレス オブジェクトが作成されます。アプリケーションは、CiscoProvider の getIntercomAddresses () インターフェイスを呼び出すことによって、プロバイダーのドメインに存在するすべての CiscoIntercomAddresses を取得できます。

インターコム コールは、Cisco Unified JTAPI インターフェイスから CiscoIntercomAddress.ConnectIntercom () インターフェイスをコールして開始できます。アプリケーションは、このインターフェイスにインターコム ターゲット DN を提供します。インターコム ターゲット DN がアプリケーションによって事前に構成または設定されている場合、アプリケーションは CiscoIntercomAddress.getTargetDN() インターフェイスを呼び出してターゲットの DN を取得できます。そうでない場合、アプリケーションは有効なインターコム ターゲットを提供しないとコールできません。

インターコム コールはインターコム ターゲットで自動応答されます。Cisco Unified JTAPI により、インターコム ターゲットの TerminalConnection/CallCtlTerminalConnection が Passive/Bridged 状態に移行されます。インターコム ターゲットが応答を開始するには、インターコム ターゲットの TerminalConnection に対してアプリケーションが join () インターフェイスを呼び出します。join () が成功すると、インターコム ターゲットの TerminalConnection/CallCtlTerminalConnection は Active/Talking 状態に移行します。インターコム コールの場合、Cisco Unified JTAPI は次のインターフェイスだけをサポートしています。

- Call.drop ()
- Connection.disconnect ()
- CallCtlTerminalConnection.join ()

アプリケーションは、インターコム コールに対して機能操作を実行できません。CiscoIntercomAddress での接続に対してリダイレクト、コンサルト、転送、会議またはパークを呼び出すと、Cisco Unified JTAPI によって例外がスローされます。また、アプリケーションが CiscoIntercomAddress に対して setForwarding ()、getForwarding ()、cancelForwarding ()、unPark ()、setRingerStatus ()、setMessageWaiting ()、getMessageWaiting ()、setAutoAcceptStatus () または getAutoAcceptStatus () を呼び出すと、例外を受け取ります。

アプリケーションは、提供された API から、設定されたインターコム ターゲットの DN の値および CiscoIntercomAddress のラベルを取得できます。Cisco Unified JTAPI では、デフォルトを返す API と、インターコム ターゲットに現在設定されている値を返す API の 2 種類の API を提供しています。デフォルト値は、Cisco Unified Communications Manager Administration によって事前に設定されているインターコム ターゲットの DN およびラベルです。現在の値は、アプリケーションによって設定されている暫定ターゲットの DN およびラベルです。アプリケーションによって値が設定されていない場合、現在の値はデフォルト値と同じままです。アプリケーションは、CiscoIntercomAddress に対して API の setIntercomTarget () を呼び出して、インターコム ターゲットの DN、ラベルおよび Unicode ラベルを設定できます。インターコム アドレスに対してインターコム ターゲット、ラベルおよび Unicode ラベルを設定できるのは、1 つのアプリケーションだけです。2 つのアプリケーションが値を設定しようとすると、最初のアプリケーションが成功し、2 番目は例外を受信します。インターコム ターゲットの DN とラベルが変更されると、Cisco Unified JTAPI は CiscoIntercomAddress に追加された AddressObserver に対して CiscoAddressIntercomInfoChangedEv を提供します。アプリケーションがインターコム ターゲットの DN およびラベルを設定しており、JTAPI または CTI でフェールオーバーまたはフェールバックが発生すると、JTAPI または CTI は、以前に設定されたインターコム ターゲットの DN、ラベルおよび Unicode ラベルを復元します。JTAPI または CTI がインターコム ターゲットの DN、ラベルまたは Unicode ラベルを復元できない場合、Cisco Unified JTAPI によって CiscoIntercomAddress の AddressObserver に CiscoAddrIntercomInfoRestorationFailedEv が送信されます。アプリケーションが失敗するか、何らかの理由でアプリケーションが停止すると、ターゲットの DN、ラベルおよび Unicode ラベルはデフォルトにリセットされます。JTAPI は、CiscoIntercomAddress に対して resetIntercomTarget () インターフェイスを提供して、インターコム ターゲットをリセットします。

自動応答は、CiscoIntercomAddress に対して常に有効になります。アプリケーションから CiscoAddress に対して getAutoAnswerEnabled () メソッドを呼び出すと、特定のアドレスの自動応答機能を取得できます。

インターコムの開始側に一方向メディアで接続されているインターコム ターゲットの場合、デバイスの状態は CiscoTermDeviceStateWhisper に設定されます。これは、端末オブジェクトの新しいデバイス状態です。この状態では、端末は新しいコールを開始したり、新しい着信コールを受けられます。アプリケーションがこのデバイス状態の受信に対してフィルタを有効にしている場合、アプリケーションは CiscoTermDeviceStateWhisperEv を受信します。アプリケーションは、CiscoTermEvFilter に対して setDeviceStateWhisperEvFilter() をコールしてフィルタを有効にできます。DeviceStates の DEVICESTATE_ACTIVE、DEVICESTATE_HELD、DEVICESTATE_ALERTING は、いずれも、DEVICESTATE_WHISPER を無効にします。もし 1 つのコールがアクティブ、保留またはアラート状態で、別のコールが whisper の場合、DeviceState はそれぞれ DEVICESTATE_ACTIVE、DEVICESTATE_HELD または DEVICESTATE_ALERTING になります。

（注） Cisco Unified JTAPI では、インターコム ターゲットが開始側に応答するために、javax.telephony.TerminalConnection インターフェイスの join() を実装しています。システムでは、CiscoIntercomAddresses に対してのみこのインターフェイスを実装しています。Passive または Bridged 状態の通常の共用回線 … 対してアプリケーションがこのインターフェイスを呼び出すと、JTAPI によって MethodNotImplimented 例外がスローされます。

ヒント アプリケーションで制御されているデバイス（端末）にインターコム回線が設定されていない場合、この機能には下位互換性があります。アプリケーションは、アプリケーションで制御されているデバイス（端末）にインターコム回線を設定しないことにより、インターコム機能を無効にできます。

これらのインターフェイスの変更に関する詳細については、次のトピックを参照してください。

- CiscoIntercomAddress
- CiscoAddrIntercomInfoRestorationFailedEv
- CiscoAddress
- CiscoCall
- CiscoProvider
- CiscoTermEvFilter
- CiscoTerminal
- CiscoTerminalConnection
- CiscoTermDeviceStateWhisperEv

アラビア語とヘブライ語の言語サポート

このバージョンの Cisco Unified JTAPI では、アラビア語とヘブライ語をサポートしています。これらの言語は、インストール時および Cisco Unified JTAPI Preferences のユーザ インターフェイスで選択できます。

下位互換性

この機能は下位互換性があります。

Do Not Disturb（サイレント）

Do Not Disturb（DND; サイレント）機能を使用すると、電話機のユーザが、電話機を離れるときや受信コールに応答したくない場合に、電話機を DND 状態にできます。[DND] ソフトキーを使用して、この機能を有効または無効に設定します。

ユーザ ウィンドウから、DND の次の設定を指定できます。

- DND Option : Ringer オフ
- DND Incoming Call Alert : ビープのみ / フラッシュのみ / 無効
- DND Timer : 0 ～ 120 分の値。「DND（サイレント）がアクティブであることをユーザに知らせる時間（分単位）」を指定します。
- DND status : on/off

（注） アプリケーションからは、DND ステータスを有効または無効に設定することだけが可能です。

- アプリケーションから DND ステータスを設定するには、CiscoTerminal で新しいインターフェイスを呼び出します。
- DND ステータスが電話機、Cisco Unified Communications Manager Administration、またはアプリケーションによって設定されると、JTAPI も DND ステータスの変更をアプリケーションに照会します。
- アプリケーションでは、事前の通知を受信するためには、CiscoTermEvFilter のフィルタを有効にする必要があります。
- アプリケーションでは、CiscoTerminal の新しいインターフェイスから、DND ステータスも照会できます。
- アプリケーションでは、CiscoTerminal の新しいインターフェイスから、DND オプションも照会できます。

（注） この機能は、電話機と CTI ポートに適用できます。ルート ポイントは該当しません。

CER アプリケーションが発信した緊急コールが、DND が有効になっているアプリケーションに着信すると、DND 設定が無効にされ、コールがアプリケーションに繋がれます。CiscoCall、CiscoConnection および CiscoRouteSession の redirect() および selectRoute() API に含まれる FeaturePriority という新しいパラメータにより、この機能が提供されます。緊急コールを発信する CER アプリケーションにより、FeaturePriority が FeaturePriority_Emergency に設定されます。アプリケーションが feature priority を設定するのは、緊急コールだけです。通常のコールの場合、アプリケーションは feature priority をまったく設定しないか、または FeaturePriority_Normal に設定します。通常のコールで、アプリケーションが FeaturePriority_Emergency に設定することはありません。インターコムなどの機能コールを発信する場合、アプリケーションは FEATUREPRIORITY_URGENT を設定する必要があります。

（注） CiscoCall の connect() API では FeaturePriority パラメータをサポートしていません。

デバイスがイン サービスになる前に getDNDStatus()、setDNDStatus() または getDNDOption() を実行すると、アプリケーションに例外が返されます。

setDNDStatus() 要求の送信後に発生した DB アップデートの失敗またはデバイスのアウト オブ サービスの状況に対処するために、DND（サイレント）に事後条件が追加されます。setDNDStatus() 要求の送信後に DB アップデートの失敗またはデバイスのアウト オブ サービスの状況が発生すると、setDNDStatus() によってアプリケーションに CiscoTermDNDStatusChangedEv が配信されます。このイベントを受信しなかった場合、事後条件によるタイムアウトが発生し、「could not meet post conditions of setDNDStatus()」という例外がスローされます。

下位互換性

この機能は下位互換性があります。この機能を設定すると、アプリケーションは新しいイベントを認識します。新しいイベントは、TerminalEventFilter インターフェイスからフィルタできます（CiscoTermEvFilter）。デフォルトで、このフィルタは無効に設定されており、システムは新しいイベントを配信しません。

詳細は、次のトピックを参照してください。

- CiscoTerminal
- CiscoTermDNDStatusChangedEv
- CiscoTermEvFilter
- CiscoCall
- CiscoConnection
- CiscoRouteSession
- CiscoTermInServiceEv

セキュア会議

この機能では、コールがセキュアかどうかアプリケーションに通知され、セキュアな電話会議を可能にします。コール全体のセキュリティ ステータスが変更されると、セキュア会議からは、コールでのイベントという形でアプリケーションに通知されます。コール全体のセキュリティ ステータスが変更されると、CiscoCallSecurityStatusChangedEv で全体的なコールのセキュリティ ステータスを受信します。端末が通話状態になると、JTAPI により、コールのセキュリティ ステータス情報がアプリケーションに配信されます。アプリケーションは、CiscoCall の新しいインターフェイスを使用して、コールのセキュリティ ステータスを照会できます。アプリケーションが既存のコールのモニタリングを始めると、アプリケーションではセキュリティ ステータス情報を参照できるようになります。

共有アドレスの場合、CiscoCallSecurityStatusChangedEv も Remote In Use の状態のアドレスにも報告されます。OverallCallSecurityStatus は、アクティブな端末で報告されるステータスと一致します。たとえば、A（暗号化）、B（暗号化）、C（認証済）および C'（認証済）という三者会議の場合、システムにより、C および C' に CiscoCallSecurityStatusChangedEv with OverallCallSecurityStatus = Authenticated が報告されます。このイベントは、コールごとに配信されます。

OverallCallSecurityStatus が Encrypted であるかどうかに関係なく、SRTP キー情報は引き続き暗号化されている通話者について送信されます。たとえば、A（暗号化）、B（暗号化）および C（セキュアでない）という三者会議の場合、電話会議の OverallCallSecurityStatus は NotAuthenticated です。ただし、暗号化されている通話者が含まれているため、A、B および会議ブリッジを繋ぐメディアは暗号化されたままです。したがって、OverallCallSecurityStatus に関係なく、A と B は SRTP キーを受信します。

下位互換性

この機能は下位互換性があります。jtapi.ini ファイルの新しいパラメータ EnableSecurityStatusChangedEv が、セキュア会議の機能によって生成される新しいイベント CiscoCallSecurityStatusChangedEv を制御します。アプリケーションでは、jtapi.ini ファイルに

「EnableSecurityStatusChangedEv=1」という行を追加することにより、このパラメータをオンにしてこの新しいイベントを受信できます。デフォルトで、このパラメータは jtapi.ini ファイルに含まれていないため、イベント通知は無効です。com.cisco.jtapi.extensions.CiscoJtapiProperties の setCallSecurityStatusChangedEv() インターフェイスにより、アプリケーションでこの ini パラメータをプログラムから設定できます。

詳細は、CiscoCallSecurityStatusChangedEv を参照してください。

Cisco Unified IP 7931G フォンの対話

Cisco Unified IP 7931G フォンは、次の 2 つのモードに設定できます。

- NoRollOver
- RollOver（同じ DN 間または異なる DN 間で）

Cisco Unified IP 7931G フォンが NoRollOver モードに設定されると、SCCP を実行する通常の電話と同様に機能します。このモードでは、異なるアドレス間で転送や会議を実行できません。JTAPI は、7931G フォンが NoRollOver モードに設定されている場合、制御およびモニタリングをサポートします。

RollOver モードでは、Cisco Unified IP 7931G フォンは、異なるアドレス間での転送や会議をサポートします。このモードでは、JTAPI は、Cisco Unified IP 7931G フォンの制御およびモニタリングができません。アプリケーションでは、このような端末やアドレスは制限されていると認識されます。Cisco Unified IP 7931G フォンがアプリケーション ユーザの制御リストに含まれており、設定が NoRollOver から RollOver モードに変更されると、JTAPI は、原因 CiscoRestrictedEv.CAUSE_UNSUPPORTED_DEVICE_CONFIGURATION で、CiscoAddrRestrictedEv イベントを Cisco Unified IP 7931G フォンの各アドレスに送信し、CiscoTermRestrictedEv を各端末に送信します。

ただし、電話の設定が RollOver から NoRollOver モードに変更されると、JTAPI は、CiscoAddrActivatedEv イベントを Cisco Unified IP 7931G フォンの各アドレスに送信し、CiscoTermActivatedEv を各端末に送信します。

RollOver モードに設定されている Cisco Unified IP 7931G フォンが JTAPI によって制御されたアドレスに転送または会議を発信すると、JTAPI のアプリケーションは最後のコールとコンサルト コールで共通のコントローラを認識しません。これにより、JTAPI アプリケーションの動作が異なってきます。JTAPI アプリケーションがイベントの情報を処理している方法によっては、この転送または会議の場合に JTAPI イベントを処理する方法を変更する必要があります。

異なるアドレス間での転送および会議を無効にするには、Cisco Unified Communications Manager Administration の phone configuration ウィンドウで、Cisco Unified IP 7931G フォンを NoRollOver（ロールオーバーなし）モードに設定します。

CiscoRestrictedEv インターフェイスには、2 つの新しい原因コードがあります。Cisco Unified IP 7931G フォンが RollOverMode に設定されているために端末またはアドレスが制限されている場合、JTAPI は原因 CiscoRestrictedEv.UNSUPPORTED_DEVICE_CONFIGURATION で CiscoAddrRestrictedEv を送信します。また、このリリースではデフォルトの原因コード CAUSE_UNKNOWN も導入されており、これはアプリケーションが処理可能です。

下位互換性

この機能は下位互換性があります。この機能を無効にするには、クラスタ内のすべての Cisco Unified IP 7931G フォンを NoRollOver モードに設定するか、または Cisco Unified Communications Manager のクラスタ内に Cisco Unified IP 7931G フォンを含めないようにします。Cisco Unified Communications Manager のクラスタ内の電話が 1 つでも RollOver モードに設定されている場合、JTAPI が制御するアドレスや端末の動作に変更を及ぼす可能性があります。

詳細は、CiscoRestrictedEv を参照してください。

バージョン形式の変更

リリース 6.0 では、Cisco Unified JTAPI のバージョンが 4 桁の形式から 5 桁の形式に変更になります。これは、Cisco Unified Communications Manager で使用されている形式と同じです。JTAPI のバージョンは Cisco Unified Communications Manager のバージョンと同じままです。新しいインターフェイスでは、アプリケーションは拡張されたバージョン番号を取得します。CiscoJtapiVersion を参照してください。

下位互換性

この機能は下位互換性があります。

発信側の IP アドレス

CallCtlConnOfferedEv および RouteEvent の拡張により、発信側の IP アドレス取得の手段が提供されます。この機能は、基本的な通話、転送や会議のためのコンサルト コール、および基本的なリダイレクトと転送の着信側に、発信側の IP アドレスを通知します。発信側が変更になる場合を含め、これ以外の状況や機能対話はサポートされていません。この機能では発信側のデバイスとして IP フォンだけをサポートしていますが、他の発信デバイスの IP アドレスも提供できます。
CiscoCallCtlConnOfferedEv および CiscoRouteEvent を参照してください。

下位互換性

この機能は下位互換性があります。

MLPP（Multilevel Precedence and Preemption）のサポート

Cisco Unified Communications Manager では、MLPP（Multilevel Precedence and Preemption）用に設定された電話機による補足サービスの使用が可能です。Cisco Unified Communications Manager は、これを実現するために、コールの優先順位レベルを設定しています。

（注）　JTAPI では、アプリケーションの優先順位レベルを提供していません。

コントローラ以外による会議への通話者の追加

会議に参加しているすべての通話者が、会議に参加者を追加できるようになりました。以前のリリースでは、参加者を追加できるのは会議コントローラだけでした。

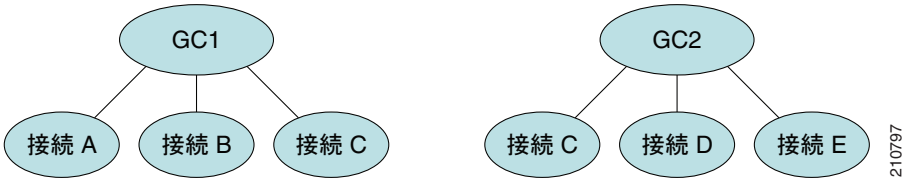
- CiscoConferenceStartEv には、要求者の識別子が含まれています。
- メソッド getConferenceControllerAddress は、要求者の Terminal ConnectionTerminalConnection を返します。
- CiscoConferenceStartEv の新しいメソッド getOriginalConferenceControllerAddress() は、元のコントローラの TerminalConnectionTerminal Connection を返します。

会議のチェーニング

会議のチェーニング機能を使用すると、アプリケーションが 2 つの別々の電話会議を繋げることができます。JTAPI アプリケーションでは、チェーニングされた電話会議は 2 つの別々のコールとして認識されます。電話会議がチェーニングされると、JTAPI によって会議チェーン用に新しい Connection が作成され、CallCtlCallObserver に CiscoConferenceChainAddedEv イベントが提供されます。会議のチェーンがコールから削除されると、JTAPI によって会議チェーンの接続が解除され、CallCtlCallObserver に CiscoConferenceChainRemovedEv イベントが提供されます。アプリケーションは、CiscoConferenceChainAdded/RemovedEv から、すべての会議チェーン接続へのリンクを提供する CiscoConferenceChain を取得できます。

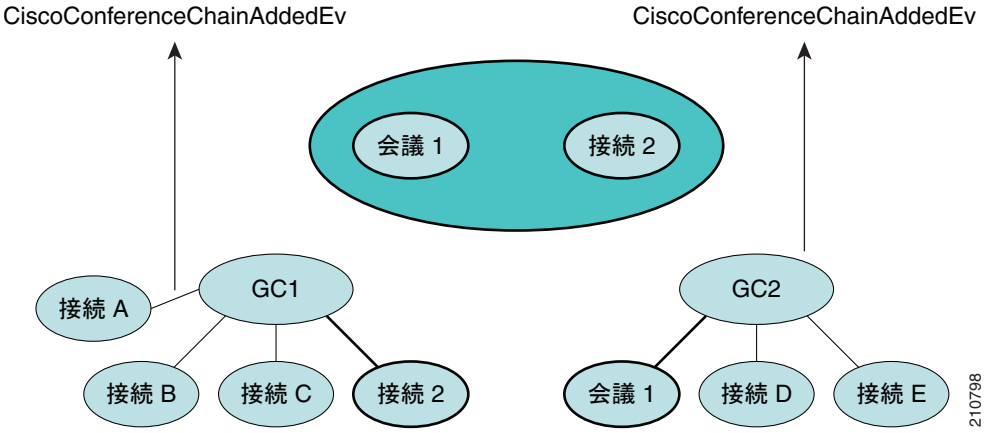
図 3-1 は、電話会議 GC1 に参加している通話者 A、B および C と、電話会議 GC2 に参加している通話者 C、D および E を示しています。

図 3-1　　チェーニング前のコール



会議チェーンが作成された後、コールは図 3-2 のようになります。

図 3-2　　チェーニング後のコール



アプリケーションは、チェーニングされた会議のすべての参加者を CiscoChainedConference オブジェクトから取得できます。このオブジェクトは、チェーニングされているすべての電話会議の会議チェーン Connection を返します。Connection のリストを参照することにより、すべてのチェーニングされた電話会議のリストを取得できます。ただし、各会議から最低 1 人の参加者がアプリケーションで監視されている必要があります。

（注） 会議のチェーニング、またはチェーニングされた電話会議への参加者の追加が関わる会議のシナリオでは、JTAPI は ConferenceStarted/Ended イベントを提供しません。

詳細は、次のトピックを参照してください。

- CiscoCall（getConferenceChain() インターフェイス用）

- CiscoConferenceChain
- CiscoConferenceChainAddedEv
- CiscoConferenceChainRemovedEv

帯域幅不足および未登録 DN 発生時の転送

この機能では、帯域幅不足および未登録 DN 発生時に対処するために、転送ロジックが次のように拡張されています。

- 帯域幅不足のためにコールをリモートの宛先に配信できない場合、コールは AAR 宛先マスクまたはボイスメールに再ルーティングされます。これらの設定は、Cisco Unified Communications Manager GUI の電話番号ウィンドウでユーザが変更できます。
- 未登録 DN：コールが未登録 DN に対して発信されると、コールは Call Forward on No Answer（CFNA; 無応答時コール転送）に設定されている DN に配信されます。

Call Forward No Bandwidth（CFNB）のためにコールが QSIG を使用するトランク / ゲートウェイを越えて別の着信先クラスタに転送された場合、コールの履歴が失われる可能性があります。たとえば、電話 A から帯域幅の少ない地域にある電話 B をコールし、CFNB でコールを別のクラスタにある電話 C に転送するように設定されている場合、クラスタ間の転送に QSIG 使用されていると、元の着信側と最後にリダイレクトした通話者は着信先に伝達されない可能性があります。

ダイレクト コール パーク

この機能では、ユーザが自分の選択したパーク コードにコールを転送して、コールをパークできます。

例

A が B にコールし、B がそのコールをパーク DN に転送する。転送が完了すると、A から B へのコールは指定済みのパーク DN でパークされる。A には MOH が再生される（設定されている場合）。C が（プレフィクス コードとパーク コードをダイヤルして）コールをパーク解除すると、A と C が接続される。

A が パーク DN に直接コールする場合は、A はパーク DN に接続され、このパーク DN はビジーとしてマークされる。A はパークの復帰まで、このパークされた DN に接続されたままになる。

C がパーク DN でコールのパークを解除しない場合は、コールをパークした DN（B）に、このコールのパークが復帰して A と B が再び接続されます。B は別のパーク DN に d-Park を再試行することができます。パークの復帰が発生すると、Cisco Unified Communications Manager JTAPI が新しい理由コードをアプリケーションに渡します。

CTI は、「パーク番号 : (<プレフィクス コード>)<DPark DN>」の形式でパークされた番号を Cisco Unified Communications Manager JTAPI に送信します。Cisco Unified Communications Manager JTAPI はこの情報を解析し、プレフィクス コードと DPark DN の両方をアプリケーションに公開します。

コールのパークが解除されると、パークされていた通話者とパークを解除している通話者の両方が、CTI が付与した理由コードが設定された CPIC イベントを受信し、パークされていた通話者がパークを解除している通話者と接続されます。

通話者 A が dPark DN をコールして、B も同じ dPark DN をコールすると、A か B のどちらかは dPark DN に接続できますが、他方は接続解除されます。

Cisco Unified Communications Manager JTAPI のサポート

Cisco Unified Communications Manager JTAPI では、この機能がサポートされています。ダイレクト コール パーク DN にコールが転送される（dPark される）と、ダイレクト コール パーク DN 用に Connection が作成されてその Call Control Connection のステータスが CallControlConnection.QUEUED になったことがアプリケーションで認識され、CiscoTransferstart イベントと End イベントが配信されます。アプリケーションは、CiscoConnection の新しいインターフェイスを使用して、コールのパークを解除するために必要なプレフィクス コードを取得できます。

パフォーマンスとスケーラビリティ

この機能のパフォーマンスへの影響は、既存の転送機能と同じです。

ボイス メールボックスのサポート

この機能では、ボイス メールボックスの番号が公開されるので、Cisco Unified Communications Manager JTAPI アプリケーションで、電話番号から正しいボイス メールボックスへコールを転送できます。

Cisco Unified Communications Manager の管理者は、各電話番号にボイスメール プロファイルを関連付けることかできます。いずれかの転送設定でボイスメール オプションが有効になっているときに、対応する転送が有効になっている場合は、ボイスメール プロファイルに関連付けられているボイス メール パイロット番号にコールが転送されます。

ボイスメール プロファイルには、ボイスメール パイロット番号フィールドとボイス メールボックス マスク フィールドがあります。ボイス メールボックス マスクでは、自動登録された電話機のボイス メールボックス番号をフォーマットするために使用するマスクを指定します。自動登録された電話機の電話回線からボイス メッセージング システムにコールを転送するときには、Cisco Unified Communications Manager によって、その電話回線の Voice Mail Box フィールドに設定されている番号にこのマスクが適用されます。

たとえば、972813XXXX というマスクを指定した場合、ディレクトリ番号 7253 のボイス メールボックス番号は 9728137253 になります。マスクを入力しない場合は、ボイス メールボックス番号はディレクトリ番号（この例では 7253）と同じ番号になります。

Cisco Unified Communications Manager JTAPI のサポート

この機能をサポートするために、Cisco Unified Communications Manager JTAPI では、called、lastRedirected、originalCalled の各通話者のボイス メールボックス番号が公開されています。これらのボイス メールボックスのフィールドは、CiscoCall オブジェクトで公開されている CiscoPartyInfo で公開されています。通話者にボイスメールが設定されていない場合、Cisco Unified Communications Manager JTAPI は、ボイス メールボックスのフィールドに空の文字列を返します。

Cisco Unified Communications Manager JTAPI の以前のリリースでは、ボイス メールボックスのフィールドをアプリケーションに公開していませんでした。そのため、Cisco Unified Communications Manager JTAPI ボイス メールボックス アプリケーションでは、voicemailboxmask がボイスメール プロファイルに設定されているかどうかが判断できず、電話番号とは異なるボイス メールボックス番号が生成される可能性がありました。

パフォーマンスとスケーラビリティ

この機能により、Cisco Unified Communications Manager JTAPI レイヤからアプリケーション レイヤへのトラフィックが増加することはありません。ただし、ネットワーク上で追加のフィールドが渡されるので、パフォーマンスに多少影響する可能性があります。

Privacy On Hold

この機能は、保留中のプライベート コールのプライバシーを強化するものです。プライバシーが有効になっていると、コールを保留状態にした電話機だけがそのコールを取得でき、発信者の名前と番号は表示されません。

この機能では、共用アドレスで、他の共用アドレスからのコールへの割り込みを許可するかどうかを決定することができます。プライバシーを有効にすると、他の共用アドレスからコールに割り込めなくなります。プライバシーは、端末のプロパティです。IP フォンでは、プライバシー機能ボタンによって、ユーザがプライバシー機能の有効と無効を切り替えることができます。端末のアクティブ コールに対しては、プライバシーの有効と無効を動的に切り替えることができます。コールのプライバシーがオンになっているときには、他の共用アドレスから参照できる TerminalConnection の状態は In Use（使用中）に設定されます。CallProgress 中にプライバシーの状態が変化した場合は、CiscoTermConnPrivacyChangedEvent がアプリケーションに配信されます。

以前のリリースでは、プライバシーが有効の場合にコールが保留状態にされると、すべての TerminalConnection が TermConnHeld 状態になり、他の共用アドレスの任意の terminalConnection からコールの保留を解除できていました。Cisco Unified Communications Manager 4.2 では、Enforce Privacy on Held Calls サービス パラメータが有効になっている場合に、コールのプライバシーが有効になっていると、コールを保留状態にしても他の共有アドレスの terminalConnection は変更されず、In Use（使用中）状態のままになります。

パフォーマンスとスケーラビリティ

この機能では、Cisco Unified JTAPI、アプリケーション、Cisco Unified Communications Manager の間に追加のトラフィックは生成されないので、パフォーマンスへの影響はありません。

Cisco RTP イベントの CiscoRTPHandle インターフェイス

次のインターフェイスは、アプリケーションがイベントから CiscoRTPHandle を取得できるように拡張されました。

- CiscoRTPInputStartedEv
- CiscoRTPInputStoppedEv
- CiscoRTPOutputStartedEv
- CiscoRTPOutputStoppedEv

CiscoRTPHandle は、Cisco Unified Communications Manager のコールの CallID を表し、端末上でコールがアクティブな間は変化しません。コールと関連付けられた GCID が変更されても、特定の端末/アドレス上では CiscoRTPHandle は一定に保たれます。

保留の復帰

保留の復帰機能では、アプリケーションに通知が送信されます。この通知には、保留中のコールの存在を Cisco Unified Communications Manager がアドレスに通知するものと、コールが一定時間 ONHOLD 状態であることを通知するものがあります。アプリケーションは、コールを ONHOLD にしたアドレスのコール オブザーバ上で、この通知を CiscoCallCtlTermConnHeldReversionEv Call ControlTerminalConnection イベントとして受信します。この通知は、保留中のコールについて 1 回だけアプリケーションに送信されます。

イベントは、コールが保留された端末の TerminalConnection にだけ送信されます。アドレスが共用回線のアドレスである場合、共用回線アドレスの他の TerminalConnection はイベントを受信しません。

このイベントを受信するには、アプリケーションはアドレスにコール オブザーバを追加する必要があります。このイベントの原因は CAUSE_NORMAL です。保留復帰タイマーの期限が切れて、通知が電話機に送信された後にコール オブザーバが追加された場合、アプリケーションには原因 CAUSE_SNAPSHOT で CiscoCallCtlTermConnHeldReversionEv が送信されます。

詳細は、CiscoCallCtlTermConnHeldReversionEv を参照してください。

トランスレーション パターンのサポート

JTAPI アプリケーションが制御する Address に適用されているトランスレーション パターンに対して発信者番号変換マスクを設定すると、発信側と着信側の両方を監視している場合、アプリケーションは余分の Connection の作成および Connection 解除を認識できます。着信側だけが監視されている場合、実際の発信者ではなく、トランスフォームされた発信者に対して Connection が作成され、CiscoCall.getCurrentCallingParty() はトランスフォームされた発信者を返します。一般的に、JTAPI は Call 内の適切な Connection を作成できない可能性があり、currentCalling、currentCalled、calling、called および lastRedirecting の通話者について正確な情報を提供できない可能性があります。

たとえば、トランスレーション パターン X で、発呼側トランスフォーメーション マスク Y と着信側トランスフォーメーション マスク B が設定されているとします。A が X をコールすると、コールは B に着信します。この場合、次のようになります。

- アプリケーションが B だけを監視している場合、JTAPI は Y と B に対して Connection を作成し、CiscoCall.getCurrentCallingParty() は Address Y を返します。
- アプリケーションが A と B の両方を監視している場合、A と B に対して Connection が作成され、Y に対して Connection が一時的に作成されて破棄され、CiscoCall.getCurrentCallingParty() は Address Y を返します。

基本的なコールに対して他の機能も実行すると、これ以外にもコールの情報の不統一が発生する可能性があります。JTAPI アプリケーションが制御するアドレスに適用される可能性のあるトランスレーション パターンに対しては、発呼側変換マスクを設定しないことを推奨します。

発信側の IP アドレス

発信側の IP アドレスの拡張機能は、基本的な通話、転送や会議のためのコンサルト コール、および基本的なリダイレクトと転送の着信側に、発信側の IP アドレスを通知します。この機能では発信側のデバイスとして IP フォンだけをサポートしていますが、他の発信デバイスの IP アドレスも提供できます。

（注）　この他の機能対話は、発信側が変更される場合の対話を含め、サポートされていません。

Cisco では、新しく CallCtlConnOfferedEv と RouteEvent クラスに対する機能拡張が行われ、発信側の IP アドレスを取得するためのメソッドが公開されました。この新しい拡張機能は、CiscoCallCtlConnOfferedEv と CiscoRouteEvent です。発信側の IP アドレスが使用できない場合は、空の値が戻ります。

基本的なコール シナリオ

JTAPI アプリケーションは相手 B を監視する

相手 A は IP フォンである

A が B にコールする

A の IP アドレスが B を監視する JTAPI で使用可能な場合のコンサルト転送のシナリオ

JTAPI アプリケーションは相手 C を監視する

相手 B は IP フォンである

A は B に通話する

B は C へのコンサルト転送コールを開始する

B の IP アドレスは相手 C を監視している JTAPI アプリケーションから使用できる

コンサルト会議のシナリオ

JTAPI アプリケーションは相手 C を監視する

相手 B は IP フォンである

A は B に通話する

B は C へのコンサルト会議コールを開始する

B の IP アドレスは相手 C を監視している JTAPI アプリケーションから使用できる

リダイレクトのシナリオ

JTAPI アプリケーションは相手 B と相手 C を監視する

相手 A は IP フォンである

A が B にコールする

A の IP アドレスは相手 B を監視している JTAPI アプリケーションから使用できる

通話者 A が B を通話者 C にリダイレクトする

B を監視する JTAPI アプリケーションでは、発信側の IP アドレスは*使用できない*

B の IP アドレスのコールが相手 C を監視している JTAPI アプリケーションに提供される

下位互換性

この機能は下位互換性があります。アプリケーションは、新しい API を起動してコールの IP アドレスを問い合せる必要があります。

回線をまたいで参加（Join Across Lines）

回線をまたいで参加（Join Across Lines）機能を使用すると、回線をまたいで参加をサポートできます。この機能では、電話機の [Join] ソフトキー、または JTAPI が提供する conference() API を使用して、同じ端末にある異なるアドレス上の複数のコールを会議に参加させることができます。アプリケーションは最後のコールとコンサルト コールで共通のコントローラを認識できない場合、JTAPI アプリケーションに対する動作が変わります。

API 自体は変更されず、会議に参加するコールが同じアドレス（通常の会議）か異なるアドレス間（回線をまたいで参加）かどうかに関係なく、同じイベントが配信されます。回線をまたいで参加（Join Across Lines）機能が実行されると、1 つの会議に結合されるコンサルト コールまたは最後のコールが存在するコントローラの端末上のすべてのアドレスに、CiscoConferenceStartEv/EndEv が提供されます。

CiscoConferenceStartEv では、conferenceControllerAddress が必ずプライマリ コントローラ アドレスになります。アプリケーションで setConferenceController() API を使用してコントローラを設定できるようになりました。アプリケーションでコントローラが設定されていない場合、JTAPI によって会議に適切なコントローラが検出されます。回線をまたいで参加（Join Across Lines）機能が起動されている場合、アプリケーションでコントローラ アドレスを設定することをお勧めします。

オブザーバをコントロール アドレスに追加しないと、アプリケーションでは、CiscoConferenceStartEv の通話中または保留中端末のいずれかの Connection 値が、null 値と認識される可能性があります。このリリースよりも前のリリースでは、アプリケーションが回線をまたいで会議に参加しようとすると、その要求は JTAPI レイヤで失敗していました。このリリースでは、conference() API の実装が強化されたため、最後のコールとコンサルト コールに適した TerminalConnection が検出された後はすべての要求がパス スルーします。JTAPI は、コールに関与する複数のアドレスに共通の端末に基づいて、適切な TerminalConnection を検出します。2 つ以上のコールが参加する必要がある場合、異なるアドレス間での複数の会議もサポートされます。5.1.2 リリースの SIP デバイスではこの機能をサポートしていません。SIP デバイスでこの機能が要求された場合、JTAPI により例外（ILLEGAL_HANDLE）がスローされます。

この機能には、インターフェイスの変更はありません。アプリケーションに提供されるイベントによって、動作は変わります。

下位互換性

この機能が無効になっている場合は、会議の動作は変更されないため、この機能は下位互換性があります。この機能は、デバイスごとに有効または無効に設定できます。デバイスの回線をまたいで参加設定をデフォルトに設定すると、システム全体の CallManager サービス パラメータの Join Across Lines Policy 設定が使用されます。この機能を有効に設定し、アプリケーションが回線をまたいで参加（Join Across Lines）を実行した場合、上記のように、動作に違いがあります。

リリース 5.1 用に作成された JTAPI アプリケーションは、リリース 5.1.2 と同時にリリースされた JTAPI と下位互換性があります。JTAPI クライアントのアップグレードを考慮するのは、新しい機能を使用する場合だけです。

CiscoTermRegistrationFailedEv の新しいエラー コード

このイベントは、TerminalRegistration が何らかの理由で失敗したときに、アプリケーションに送信されます。getErrorCode() インターフェイスの戻り値で、障害の種類が示されます。このイベントを受け取った場合、アプリケーションは Terminal の再登録を試みる必要があります。このバージョンでは、新しい戻り値がこのインターフェイスに追加されました。不明な障害を処理するために、CiscoTermRegistraionFailedEv.UNKNOWN が、このバージョンから導入されました。

下位互換性

この機能は下位互換性があります。

アスタリスク（*）50 の更新

アスタリスク（*）50 機能を使用すると、ユーザは電話機の UI から、コールを元の着信者（CiscoCall.getCalledAddress() メソッドによる戻り値）と、着信者（CiscoCall.getCurrentCalledAddress() メソッドによる戻り値）に転送できます。[iDivert] ソフトキーを押すと、メニューに元の着信者と着信者の名前が表示されます。

ユーザがこの 2 つ名前のうちいずれかを選択すると、コールは選択した着信者のボイス メールボックスに転送されます。従来の即転送機能では、[iDivert] ソフトキーを押すだけで、コールは元の着信者のボイス メールボックスに転送されます。Cisco Unified Communications Manager Administration には、この機能を設定するために、次のサービス パラメータが導入されています。

- iDivert Legacy Behavior : ユーザが [iDivert] ソフトキーを押したときに、電話機が、従来の即転送機能と拡張された *50 即転送機能のどちらの動作をするかを指定します。iDivert legacy サービス パラメータが true に設定されている場合は、従来の即転送の動作が適用され、false に設定されている場合は、従来の即転送機能適用されません。

- Allow QSIG during iDivert : 従来の即転送機能の使用を、QSIG トランクを越えたボイス メッセージングが統合された構成内でも許可するか、Use Legacy iDivert サービス パラメータが true に設定されている場合に限定するかを指定します。
- iDivert User Response timer : Cisco Unified Communications Manager Administration が、iDivert 画面が削除される前に、ユーザからの応答を待つ時間を秒単位で指定します。この時間が経過してもユーザからの操作がない場合、タイマーは期限切れとなり、電話機から画面が削除されます。Use Legacy iDivert サービス パラメータが true に設定されている場合、Cisco Unified Communications Manager Administration はこのパラメータを無視します。

この機能では、JTAPI レイヤに対するインターフェイスの変更はありません。JTAPI アプリケーション側から見た動作の変更点は、コールは、元の着信者または着信者のいずれかのボイス メールに転送される可能性があることを示します。

下位互換性

この機能は下位互換性があります。

コール転送のオーバーライド

この機能では、コール転送のすべての機能をオーバーライドするメカニズムが提供されます。ユーザ（CFA の開始者）が CFA を別のユーザ（CFA ターゲット）に設定した場合、CFA ターゲットが CFA の開始者にコールした場合は、CFA は無視される必要があります。これにより、CFA ターゲットは重要なコールを CFA の開始者に着信させることができます。

この CallManager 機能の動作は、サービス パラメータ（CFADestinationOverride）を使用して設定できます。

例：Alice は DN 1000 の電話機を所持しています。* Bob は DN 2000 の電話機を所持しています。* Daniel は DN 4000 の電話機を所持しています。* Alice は CFA を 2000 に設定しています。

CFA の動作を次に示します。* Bob が Alice にコールするとします。コールは Alice に着信し、CFA の設定に従った Bob に戻るという動作はしません。* Daniel が Alice にコールするとします。このコールは CFA 設定に従って Bob に転送されます。* Bob はコールに応答し、Alice にそのコールを転送します。Alice が自分への電話を Bob に転送したため、Bob はこの操作を実行できます。この機能では、JTAPI レイヤに対するインターフェイスの変更はありません。ただし、CiscoAddress.setForward() API の起動時には、JTAPI アプリケーションでは異なる動作が確認される場合があります。例のように、CFA ターゲットが CFA の開始者にコールするシナリオでは、機能が有効に設定されている場合、コールは転送されません。

下位互換性

リリース 5.0 用に作成された JTAPI アプリケーションは、リリース 5.1 と下位互換性があります。JTAPI クライアント アップグレード アプリケーションでは、その実行や下位互換性のための JTAPI クライアントのアップグレードは必要ありません。JTAPI クライアントのアップグレードが必要なのは、新しい機能を使用する場合だけです。

パーティションのサポート

Cisco Unified Communications Manager リリース 5.0 よりも前では、JTAPI はパーティションをサポートしていませんでした。JTAPI では、DN が同じでパーティションが異なるアドレスが、同じアドレスと見なされていました。このような場合、アドレスが DN のみによって識別され、パーティション情報によって識別されないため、Address オブジェクトが 1 つしか作成されませんでした。

リリース 5.0 からは、同じ DN を持ちながら異なるパーティションに属するアドレスがサポートされ、それぞれを異なるアドレスとして扱うことが可能になりました。アドレスのパーティション情報は、後述のメソッドを通じてアプリケーションに公開されます。アプリケーションでこのパーティション サポート機能を利用するには、JTAPI インターフェイスを通じて提供される API と、それに応じたアドレス オブジェクトを使用する必要があります。

この機能は下位互換性があります。JTAPI は、アドレス オブジェクトのオープンとアクセスに現在使用されている API をサポートしています。

Cisco Unified Communications Manager リリース 5.0 では、JTAPI がパーティションに対応しており、次の構成がサポートされています。

- パーティションが同じで、デバイスが異なる同一 DN のアドレスは、共用回線として扱われます。
- 同じパーティションと同じデバイス内の同一 DN のアドレスは使用できません。
- パーティションが異なり、同じデバイスにある同一 DN のアドレスは、異なるアドレスとして扱われます。この場合は 2 つのアドレス オブジェクトが作成され、アプリケーションではアドレス オブジェクトの getPartition() API を呼び出すことによってこれら 2 つを区別できます。
- パーティションが異なり、デバイスが異なる同一 DN のアドレスは、異なるアドレスとして扱われます。この場合は 2 つのアドレス オブジェクトが作成され、アプリケーションではアドレス オブジェクトの getPartition() API を呼び出すことによってこれら 2 つを区別できます。

JTAPI でのパーティション サポートに関する変更は、アドレス オブジェクトに限定されており、JTAPI の他の関数やクラスには影響はありません。次のセクションではインターフェイスの変更点について説明します。

CiscoAddress インターフェイス

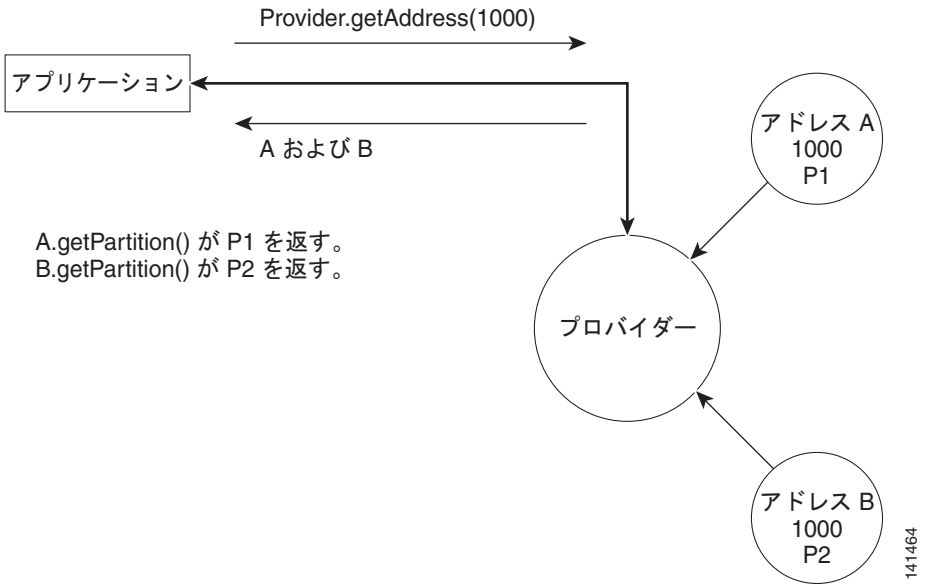
このクラスでは、次のシグニチャを持つ新しいメソッドが提供されています。

string **getPartition ()**

アドレス オブジェクトのパーティション文字列を返します。アプリケーションがパーティション情報を取得するにはこのメソッドを使用する必要があります。JTAPI ではこのパーティション情報を使用して、DN が同じでパーティションが異なるアドレスを区別し、特定のアドレスを開くためのパーティション情報を送信します。

たとえば、プロバイダーのオープンで A(1000, P1) および B(1000, P2) という 2 つのアドレスが返されたとします。A と B はアドレス オブジェクト、1000 はアドレス オブジェクトの DN、P1 と P2 はアドレスが属するパーティションを表します。

図 3-3　プロバイダーのオープンで 2 つのアドレスが返される



ユーザが A.getPartition () を呼び出すと P1 が返され、B.getPartition () を呼び出すと P2 が返されます。

provider.getAddresses() メソッドは、DN が同じでパーティション情報が異なる Address オブジェクトの場合に、複数のアドレスを返します。アプリケーションでこのメソッドを使用することで、DN が同じで異なるパーティションに属する 2 つの Address オブジェクトを区別できます。

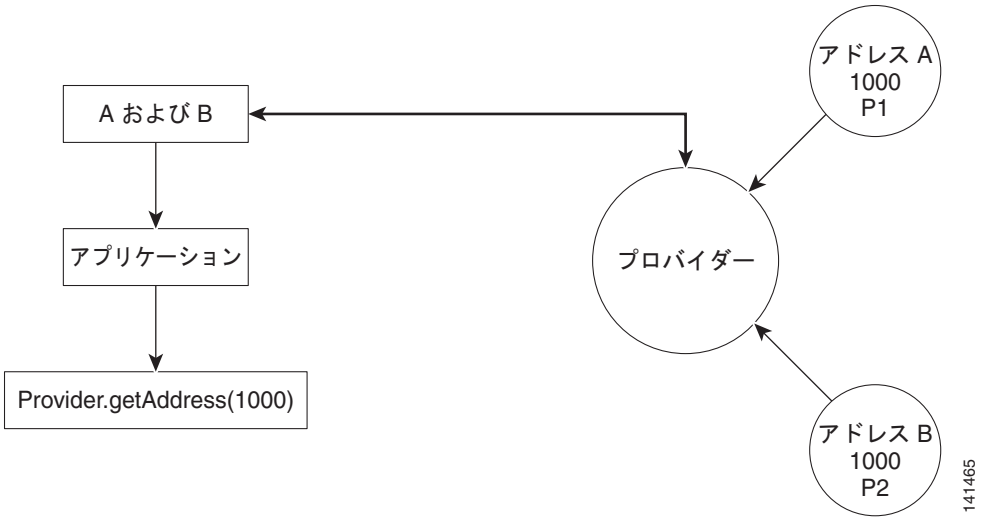
CiscoProvider インターフェイス

CiscoProvider インターフェイスでは次のメソッドが提供されています。

Address[]	**getAddress**(String number) 各パーティションの number に対応する Address オブジェクトの配列を返します。
Address	**getAddress**(String number, String partition) number パラメータと同じ DN を持ち、partition パラメータで指定されているのと同じパーティションに属する Address オブジェクトを返します。

A(1000, P1) および B(1000, P2) という 2 つのアドレスがあるとします。A と B はアドレス オブジェクトで、1000 は 2 つのアドレス オブジェクトの DN を表しており（どちらも同じ値になっています）、P1 および P2 は各アドレスが属するパーティションを示します。アプリケーションが provider.getAddress("1000") をコールすると、A および B という 2 つのアドレス オブジェクトが取得されます。

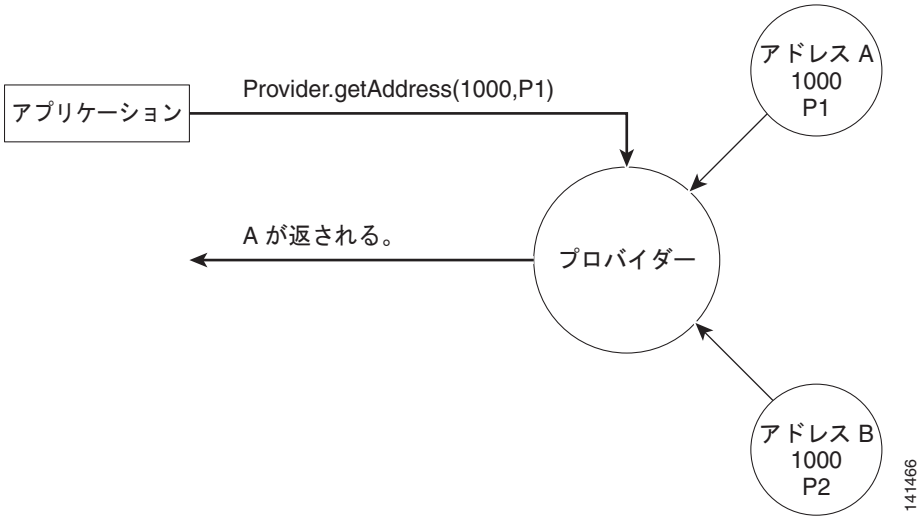
図 3-4　provider.GetAddress() によって 2 つのアドレス オブジェクトが返される



アプリケーションで A.getPartition() を呼び出すと P1 が返され、B.getPartition() を呼び出すと P2 が返され、以降、同様です。アプリケーションでは getPartition メソッドを使用して、2 つのアドレス オブジェクトを区別できます。

アプリケーションで provider.getAddress(1000, P1) を呼び出した場合を考えます。この場合、アプリケーションでは、DN が 1000 でパーティションが P1 のアドレス オブジェクトを探します。この場合は、プロバイダー オブジェクトによって「A」が返されます。

図 3-5　プロバイダーが特定のアドレスとパーティションを呼び出す



CiscoProvCallParkEv イベント

CiscoProvCallParkEv は次のメソッドをこのインターフェイスで提供します。

string **`getParkingPartyPartition()`**

パークしている通話者のパーティション文字列を返します。

string **`getParkedPartyPartition()`**

パークされた通話者のパーティション文字列を返します。

string　**getParkPartyPartition()**

パーク DN のパーティション文字列を返します。

インターフェイスの変更点については、第 6 章「Cisco Unified JTAPI 拡張」を参照してください。パーティションのサポートに関するメッセージ シーケンスを確認するには、付録 A「メッセージ シーケンスの図」を参照してください。

ヘアピン サポート

ヘアピン コールは、1 つのクラスタを離れゲートウェイを経由して他のデバイスに発信されたコールが、同じクラスタのデバイスに戻ってきたときに発生します。どちらのコールも同じクラスタ内にありますが、クラスタに戻ってきたコールの GCID とコールを発信した元の GCID は異なります。以前のリリースでは、両方が JTAPI で制御されている場合は、2 つの Connection（CiscoAddress.Internal の Connection と CiscoAddress.External の Connection）がありました。

JTAPI では、アプリケーションがヘアピン コールの両端を監視する場合に、ヘアピン コールがサポートされます。以前はアドレスが DN だけで表現されていたため、ヘアピン コールの一端しか監視できませんでした。

現在のリリースでは、DN が同じ 2 つのアドレスが存在し、一方が同じクラスタ内にあり、他方がゲートウェイを経由する場合に、JTAPI により外部 DN に対して別々のアドレス オブジェクトが作成され、アドレスのタイプごとに Connection が 1 つだけ返されるようになっています。以前のリリースではアドレスが常に DN で表されており、アプリケーションでアドレスの Connection を取得すると 2 つの Connection を取得していましたが、このプロセスによりヘアピンの問題が回避されます。

これらの問題を個別に解決すると以前のリリースとの互換性に問題が生じる恐れがあるため、このリリースにはこれらの問題に対処するための包括的なソリューションが組み込まれています。監視対象のローカル通話者と同じ DN を持つ外部通話者に関連するコールについては、適切にサポートされるようになりました。ただし、この機能に対応した新しいインターフェイスは追加されていません。

下位互換性

この機能には下位互換性はありません。

QoS のサポート

このリリースでは QoS のサポートが拡張され、アプリケーションと CTIManager 間の接続の双方向で QoS（DSCP マーキング）が可能になりました。以前のリリースでは、CTIManager からアプリケーションへの一方向の QoS のみがサポートされていました。

このリンクの双方向の DSCP（QoS）値は、CTIManager のサービス パラメータ「DSCP IP CTIManager to Application」の値によって設定されます。デフォルト値は CS3（優先順位 3）DSCP（011000）です。

「DSCP value for Audio calls」サービス パラメータは音声コールに対して推奨される QoS 値です。この値は JTAPI アプリケーションに公開されます。

JTAPI の QoS が Windows プラットフォームで機能するためには、次の設定手順をクライアント マシンで実行する必要があります。

Windows 2000 を実行している場合は、次の手順に従います。

ステップ 1　レジストリ エディタ（Regedt32.exe）を起動します。

ステップ 2　次のキーに移動します：
Local Machine¥System¥CurrentControlSet¥Services¥Tcpip¥Parameters¥ 上の HKEY_LOCAL_MACHINE

ステップ 3　[編集] メニューで、[値の追加] をクリックします。

ステップ 4　[値の名前] ボックスに、**DisableUserTOSSetting** と入力します。

ステップ 5　[データ型] リストで、[REG_DWORD] をクリックし、[OK] をクリックします。

ステップ 6　[データ] ボックスに、値 **0** を入力し、[OK] をクリックします。

ステップ 7　レジストリ エディタを終了し、コンピュータを再起動します。

Windows XP または Windows Server 2003 を実行している場合は、次の手順に従います。

ステップ 1　レジストリ エディタ（Regedt32.exe）を起動します。

ステップ 2　次のキーに移動します：
Local Machine¥System¥CurrentControlSet¥Services¥Tcpip¥Parameters¥ 上の HKEY_LOCAL_MACHINE

ステップ 3　[編集] メニューで、[新規] をポイントし、[DWORD 値] をクリックします。

ステップ 4　エントリ名として **DisableUserTOSSetting** と入力し、**Enter** キーを押します。

このエントリを追加すると、値が 0 に設定されます。値は変更しないでください。

ステップ 5　レジストリ エディタを終了し、コンピュータを再起動します。

レジストリ エディタを使用して Internet Protocol Type of Service の各ビットを設定する方法の詳細については、Microsoft のテクニカル サポート Web サイトの「Setsockopt is unable to mark the Internet Protocol type of service bits in Internet Protocol packet header」というトピックを参照してください。

次の JTAPI インターフェイスでは QoS をサポートしています。

プロバイダー インターフェイス

int	**`getAppDSCPValue`**`()` 「DSCP IP for CTI applications」サービス パラメータを返します。この値によって、JTAPI が CTI へのリンクに対して設定する DSCP 値が指定されます。アプリケーションは、ProviderInServiceEvent を受け取るたびにこの API を使用してプロバイダー オブジェクトを照会すれば、この値を取得できます。
private int	`precedenceValue = 0x00` CTI によって提供された DSCP 値を保存します。

これらのインターフェイスの詳細については、第 6 章「Cisco Unified JTAPI 拡張」を参照してください。QoS のメッセージ フローを確認するには、付録 A「メッセージ シーケンスの図」を参照してください。

トランスポート レイヤ セキュリティ（TLS）

この機能を使用すると、JTAPI アプリケーションと CTIManager の間でセキュアな接続を通じて通信できます。CTIManager は JTAPI からの接続を受け入れるために TLS リスナー ソケットを実行します。TLS 接続を確立するには、クライアント認証のためにサーバが使用するクライアント証明書と、サーバ認証のためにクライアントが使用するサーバ証明書が必要です。

Cisco Unified Communications Manager 環境では、サーバ証明書は TFTP サーバ上に CTL 形式で存在し、JTAPI はこの証明書をダウンロードします。CTL の最初のダウンロードは信頼され、検証なしで行われるため、このダウンロードはセキュアな環境で実行することを強くお勧めします。CTL は、CTL ファイル内に存在する 2 つの System Administrator Security Tokens（SAST）の一方によって署名され、その後の CTL のダウンロードは古い CTL ファイルの SAST を使用して検証されます。

JTAPI は CAPF プロトコルを使用して CAPF に接続し、クライアント証明書（LSC）を取得します。これらの証明書は、CTL 内に存在する発行元の証明書を使用して認証できます。

CTI は、クライアント証明書ごとに作成されるプロバイダー接続の数を追跡します。アプリケーションでは、クライアント証明書を使用してプロバイダーを 1 つだけ作成できます。プロバイダーのインスタンスを複数作成した場合は、両方のプロバイダーが CTI から切断され、アウト オブ サービスになります。JTAPI では、元のプロバイダーがイン サービスになるように CTI への接続を再試行しますが、プロバイダーの両方のインスタンスが引き続き存在する場合は、一定回数の再試行の後、プロバイダーが恒久的にシャットダウンされ、クライアント証明書は信頼のおけないものとしてマークされます。それ以降は、このクライアント証明書を使用してプロバイダーを作成しようとしても失敗します。アプリケーションでは、管理者に連絡して新しいインスタンス ID を設定し、新しいクライアント証明書をダウンロードして操作を再開する必要があります。

（注） クライアント証明書にはそれぞれ、Cisco Unified Communications Manager データベースで設定された一意のインスタンス ID が関連付けられています。アプリケーションでは、providerString でオプションのパラメータとしてインスタンス ID を提供することで、CiscoProvider の作成時に一意の証明書を使用できます。

TLS を使用して複数のアプリケーション インスタンスを実行するには、アプリケーション ユーザが複数のインスタンス ID で Cisco Unified Communications Manager データベースに登録されていることを確認します。これらの一意のインスタンス ID は、各インスタンスに固有のクライアント証明書を取得するためにアプリケーションによって使用されます。

JTAPI Preferences アプリケーションには、セキュリティ パラメータの設定とサーバ/クライアント証明書の更新を行うための GUI が用意されています。アプリケーション ユーザは、アプリケーション サーバの証明書をダウンロードしてインストールするために、JTAPI Preferences で TFTP サーバの IP アドレス、CAPF サーバの IP アドレス、ユーザ名、インスタンス ID、AuthorizationString の各パラメータを設定する必要があります。

クライアント レイヤ オブジェクトに関する新しいインターフェイスが JTAPI クライアント アプリケーション用に追加されました。たとえば、CTIClientProperties クラスに関する JTAPI クライアント インターフェイスが用意されています。

この機能は以前のリリースと下位互換性があり、JTAPI アプリケーションは引き続きセキュアでないソケット接続でも CTIManager に接続できます。ただし、このオプションを使用できるリリースは限定されているため、将来のリリースでは削除される可能性があります。セキュアな接続を使用するようにアプリケーションを修正することをお勧めします。

次のセクションでは、TLS をサポートするための JTAPI でのインターフェイスの変更点について説明します。

CiscoJtapiPeer.getProvider()

```
public javax.telephony.Provider getProvider(java.lang.String providerString)
throws
javax.telephony.ProviderUnavailableException
```

この変更されたインターフェイスは、新しいオプションのパラメータである InstanceID を受け取ります。必要なサービス名を含む文字列引数を渡すと、Provider オブジェクトのインスタンスを返します。

次の形式で、オプションの引数をこの文字列に提供することもできます。

```
< service name > ; arg1 = val1; arg2 = val2; ...
```

< service name > は必須で、オプションの引数＝値の各ペアはセミコロンで区切って指定します。これらの引数のキーは、次の 2 つの標準定義キーを除いて、実装に固有のものになります。

- login：ログイン ユーザ名を Provider に提供します。
- passwd：パスワードを Provider に提供します。

CiscoJtapiPeer の providerString では新しいオプションの引数が使用できます。

- InstanceID：アプリケーション インスタンスのインスタンス ID を提供します。

InstanceID は、アプリケーションの複数のインスタンスが同じクライアント マシンから TLS 接続を通じて Provider（CTIManager）に接続する場合に必要になります。アプリケーションの各インスタンスでは、TLS 接続を確立するために、それぞれ一意の X.509 証明書が必要になります。JTAPI で同じユーザ名またはインスタンス ID を使用して複数の接続をオープンしようとした場合は、CTIManager によって TLS 接続が拒否されます。インスタンス ID を提供しない場合は、JTAPI によって USER のインスタンスが 1 つランダムに選択されます。その場合、選択されたインスタンスに対応する接続がすでに存在していると、接続が失敗することがあります。

引数が null の場合、このメソッドは、JtapiPeer オブジェクトによって決められた何らかのデフォルトのプロバイダーを返します。返されるプロバイダーの状態は Provider.OUT_OF_SERVICE になります。

事後条件：

```
this.getProvider().getState() = Provider.OUT_OF_SERVICE
```

指定元：インターフェイス `javax.telephony.JtapiPeer` の `getProvider`。

パラメータ：`providerString`：必要なサービスの名前と、オプションの引数。

戻り値：Provider オブジェクトのインスタンス。

例外：`javax.telephony.ProviderUnavailableException`：指定された文字列に対応するプロバイダーが使用できないことを示します。

CiscoJtapiProperties

JTAPI では、セキュリティ オプションを有効または無効にしてセキュアな TLS ソケット接続の確立に必要なクライアント/サーバ証明書をインストールするためのインターフェイスが CiscoJtapiProperties で提供されます。

```
com.cisco.jtapi.extensions
Interface CiscoJtapiProperties
```

---

getSecurityPropertyForInstance

```
public java.util.Hashtable getSecurityPropertyForInstance()
```

このインターフェイスは、User/InstanceID に対するすべてのパラメータが設定されたハッシュ テーブルを返します。ハッシュ テーブルは次の「キー」と「値」のペアで設定されます。

表 3-3

キー	値
「user」	userName
文字列「instanceID」	InstanceID
文字列「AuthCode」	authCode
文字列「CAPF」	capfServer の IP アドレス
文字列「CAPFPort」	capfServer の IP アドレス / ポート
文字列「TFTP」	tftpServer の IP アドレス
文字列「TFTPPort」	tftpServer の IP アドレス / ポート
文字列「CertPath」	証明書パス
文字列「securityOption」	セキュリティ オプションを表すブール値（有効なら true、無効なら false）。
文字列「certificateStatus」	証明書のステータスを表すブール値（更新済みなら true、未更新なら false）。

戻り値：上記の形式による、最初のユーザおよびインスタンスのハッシュ テーブル。

getSecurityPropertyForInstance

```
public java.util.Hashtable getSecurityPropertyForInstance
(java.lang.String user, java.lang.String instanceID)
```

このインターフェイスは、User/InstanceID に対するすべてのパラメータが設定されたハッシュ テーブルを返します。ハッシュ テーブルは次の「キー」と「値」のペアで設定されます。

表 3-4

キー	値
「user」	userName
文字列「instanceID」	InstanceID
文字列「AuthCode」	authCode
文字列「CAPF」	capfServer の IP アドレス
文字列「CAPFPort」	capfServer の IP アドレス / ポート
文字列「TFTP」	tftpServer の IP アドレス
文字列「TFTPPort」	tftpServer の IP アドレス / ポート
文字列「CertPath」	証明書パス
文字列「securityOption」	セキュリティ オプションを表すブール値（有効なら true、無効なら false）。
文字列「certificateStatus」	証明書のステータスを表すブール値（更新済みなら true、未更新なら false）。

パラメータ：

`user`：セキュリティ パラメータを取得する UserName。

`instanceID`：セキュリティ パラメータを取得する InstanceID。

戻り値：上記の形式のハッシュ テーブル。

---

setSecurityPropertyForInstance

```
public void setSecurityPropertyForInstance(java.lang.String user,
                      java.lang.String instanceID,
                      java.lang.String authCode,
                      java.lang.String tftp,
                      java.lang.String tftpPort,
                      java.lang.String capf,
                      java.lang.String capfPort,
                      java.lang.String certPath,
                      boolean securityOption)
```

このインターフェイスを使用して、次のパラメータのセキュリティ プロパティを設定できます。

パラメータ：

`user`：セキュリティ パラメータを更新するユーザ名。

`instanceID`：セキュリティ パラメータを更新するインスタンス ID。

`authCode`：認証文字列。

`capf`：CAPF サーバの IP アドレス。

`capfPort`：CAPF サーバが稼動中の IP アドレスのポート番号。これは CallManager のサービス パラメータで定義されます。値が null の場合、デフォルト値は 3804 です。

`tftp`：TFTP サーバの IP アドレス。

`tftpPort`：TFTP サーバが稼動している IP アドレス / ポート番号。通常、Cisco Unified Communications Manager TFTP サーバはポート 69 上で実行されます。値が null の場合、デフォルト値は 69 です。

`certPath`：証明書をインストールする必要があるパス。

---

updateCertificate

```
public void updateCertificate(java.lang.String user,
               java.lang.String instanceID,
               java.lang.String authcode,
               java.lang.String ccmTFTPAddress,
               java.lang.String ccmTFTPPort,
               java.lang.String ccmCAPFAddress,
               java.lang.String ccmCAPFPort,
               java.lang.String certificatePath)
```

このインターフェイスは、Cisco Unified Communications Manager Certificate Authority Proxy Function（CAPF）サーバに接続することによって、USER インスタンスの X.509 クライアント証明書を証明書ストアにインストールします。また、Cisco Unified Communications Manager TFTP サーバから Certificate Trust List（CTL; 証明書信頼リスト）をダウンロードします。

ユーザ クレデンシャルが無効な場合、このメソッドは PrivilegeViolationException をスローします。TFTP サーバまたは CAPF サーバのアドレスが不正な場合、このメソッドは InvalidArgumentException をスローします。アプリケーションのすべてのインスタンスには、それぞれ固有のクライアント証明書が必要です。Cisco Unified Communications Manager データベースで複数の instanceID が設定されている場合、アプリケーションはこのインターフェイスを複数回実行してすべてのインスタンスのクライアント証明書をインストールすることができます。

事前条件：このインターフェイスを実行する際は、アプリケーションが Cisco Unified Communications Manager CAPF サーバおよび TFTP サーバとネットワーク接続されている必要があります。

事後条件：これによって、クライアント証明書とサーバ証明書が JTAPI アプリケーションのマシンにインストールされます。

パラメータ：

`user`：Cisco Unified Communications Manager データベースで設定された CTI アプリケーションのユーザの名前。

`instanceID`：Cisco Unified Communications Manager データベースで設定されたアプリケーションのインスタンス ID。アプリケーションのすべてのインスタンスには、一意の ID が必要になります。

`authCode`：Cisco Unified Communications Manager データベースで設定された認証文字列。authCode は証明書を取得するために一度だけ使用できます。

`ccmTFTPAddress`：Cisco Unified Communications Manager TFTP サーバの IP アドレス。

`ccmTFTPPort`：Cisco Unified Communications Manager TFTP サーバが稼動している IP アドレス/ポート番号。通常、Cisco Unified Communications Manager TFTP サーバはポート 69 上で実行されます。null の場合、デフォルト値は 69 です。

`ccmCAPFAddress`：Cisco Unified Communications Manager CAPF サーバの IP アドレス。

`ccmCAPFPort`：Cisco Unified Communications Manager CAPF サーバが稼動中のポート番号。これは Cisco Unified Communications Manager のサービス パラメータで定義されます。値が null の場合、デフォルト値は 3804 です。

`certificatePath`：証明書をインストールする必要があるディレクトリ パス。

例外：

`InvalidArgumentException`：指定した TFTP サーバ アドレスまたは CAPF サーバ アドレスが無効な場合、この例外がスローされます。

`PrivilegeViolationException`：指定した user、instanceID、または authCode が無効な場合、この例外がスローされます。

IsCertificateUpdated

```
public boolean IsCertificateUpdated
    (java.lang.String user, java.lang.String instanceID)
```

このインターフェイスは、指定した user/instanceID のクライアントおよびサーバの証明書が更新されているかどうかに関する情報を提供します。

パラメータ：

`user`：Cisco Unified Communications Manager Administration に定義されたユーザ名。

`instanceID`：指定したユーザ名のインスタンス ID。

戻り値：証明書が更新済みの場合は true、未更新の場合は false。

---

updateServerCertificate

```
public void updateServerCertificate(java.lang.String ccmTFTPAddress,
                 java.lang.String ccmTFTPPort,
                 java.lang.String ccmCAPFAddress,
```

```
                    java.lang.String ccmCAPFPort,
                    java.lang.String certificatePath)
```

このインターフェイスは、証明書パスに指定された X.509 サーバ証明書をインストールします。TFTP サーバのアドレスが不正な場合、このメソッドは InvalidArgumentException をスローします。自動アップデートを行うアプリケーションは、Cisco Unified Communications Manager との HTTPS 接続を呼び出す前に、このインターフェイスを使用してサーバ証明書を更新する必要があります。

事前条件：このインターフェイスを実行する際は、アプリケーションが TFTP サーバとネットワーク接続されている必要があります。

事後条件：これによって、サーバ証明書が JTAPI アプリケーションのマシンにインストールされます。

パラメータ：

`ccmTFTPAddress`：Cisco CallManager TFTP サーバの IP アドレス。

`ccmTFTPPort`：Cisco Unified Communications Manager TFTP サーバが稼動中のポート番号。null の場合、デフォルト値は 69 です。

`certificatePath`：証明書をインストールするディレクトリ パス。

`ccmCAPFAddress`：Cisco Unified Communications Manager CAPF サーバの IP アドレス。

`ccmCAPFPort`：Cisco Unified Communications Manager CAPF サーバが稼動中のポート番号。値が null の場合、デフォルト値は 3804 です。

例外：

`InvalidArgumentException`：TFTP サーバのアドレスが無効な場合。

JTAPI Preferences で提供されるインターフェイス

[JTAPI Preferences] ダイアログボックスには [セキュリティ] タブが提供されています。アプリケーション ユーザはこのタブから、ユーザ名、インスタンス ID、authCode、TFTP IP アドレス、TFTP ポート、CAPF IP サーバ アドレス、CAPF サーバ ポート、証明書パスをそれぞれ設定でき、セキュアな接続を有効にすることができます。

- [CAPF server port] 番号のデフォルトは 3804 です。

  この値は Cisco Unified Communications Manager Administration サービス パラメータ ウィンドウで設定できます。JTAPI Preferences から入力する CAPF サーバ ポート値は、Cisco Unified Communications Manager Administration ページで設定した値と同じである必要があります。

- [TFTP server port] 番号のデフォルトは 69 です。

  システム管理者からの指示がない限り、この値は変更しないでください。

- [Certificate Path] は、アプリケーションがサーバ証明書とクライアント証明書をインストールする場所です。

  このフィールドが空の場合、証明書は JTAPI.jar のクラスパスにインストールされます。

- [Certificate Update Status] には、証明書の更新状態に関する情報が表示されます。

- Cisco Unified Communications Manager へのセキュアな TLS 接続を有効にするには、[Enable Secure Connection] を選択する必要があります。

  [Enable Secure Connection] が選択されていない場合は、証明書が更新またはインストールされていても、JTAPI では CTI へのセキュアではない接続が行われます。

- [Enable Security Tracing] チェックボックスで、証明書のインストール操作のトレースを有効または無効にできます。

トレースを有効にした場合は、ドロップダウン メニューから [Error]、[Debug]、[Detailed] の 3 つの異なるレベルを選択できます。

JTAPI Preference UI を使用して、ユーザ名とインスタンス ID の 1 つ以上のペアに対してセキュリティ プロファイルを設定できます。ユーザ名とインスタンス ID のペアに対してすでにセキュリティ プロファイルが設定されている場合、アプリケーション ユーザがこのウィンドウに再度アクセスし、ユーザ名とインスタンス ID を入力して他の編集ボックスをクリックすると、セキュリティ プロファイルが自動的に編集ボックスに入力されます。

JTAPI Preference UI の [Trace Levels] タブの名前は [JTAPI トレース] に変更されました。この変更により、[JTAPI トレース] タブで変更できるトレース設定が JTAPI 層のトレース設定のみであることが明確になりました。セキュリティ証明書のインストールに対するトレースについては、[セキュリティ] タブで有効にする必要があります。

SIP フォンのサポート

このリリースの Cisco Unified Communications Manager では、SIP を実行する電話を登録して SCCP を実行する電話と相互運用することができます。以降のセクションでは、SIP を実行する電話をサポートするために導入された新しいインターフェイスに加えて、SCCP をサポートする電話と比較した場合の動作の制限および相違点について説明します。既存のすべての機能が SIP を実行する電話でサポートされるわけではありませんが、JTAPI のイベントとインターフェイスの点で、SIP を実行する電話の全般的な動作は SCCP を実行する電話とほぼ同じです。

JTAPI アプリケーションで制御できるのは SIP を実行する Cisco Unified IP Phone 7900 シリーズ（Cisco Unified IP 7970 フォンなど）のみです。Cisco Unified IP 7960 や 7940 などの SIP プロトコルを実行する電話機はアプリケーションの制御リストに含めないでください。また、サード パーティ製の SIP を実行する電話も JTAPI アプリケーションでは制御できないので、制御リストには含めないでください。

以前のリリースでは、JTAPI は SIP を実行する電話で初期的なフィーチャ セットをサポートしていました。このリリースでは、SIP を実行する電話で次の機能のサポートが追加されました。

- SIP を実行する電話のパーク
- SIP を実行する電話のパーク解除

ヒント　SIP を実行する電話と SCCP フォンでは、コンサルト コールのイベントの順序が異なります。次のような例が考えられます。

a. 端末 A が共用回線 B/B' にコールを発信します。

b. 共用回線が端末 C にコンサルト コールを発信します。

共用回線が SIP デバイスの場合、次のコール イベントが発生します。

- B（アクティブ）は OnHold -> Select -> NewCall の順に受信します。
- B'（使用中のリモート）は Select -> NewCall -> OnHold の順に受信します。

ただし、共用回線が SCCP デバイスの場合、コール イベントは両方の端末で Select -> OnHold -> NewCall の順になります。

アプリケーションがモニタリングだけを行っている場合、call.getConsultingTerminalConnection() によって null が返される可能性があります。

JTAPI は SIP を実行する電話で次の機能をサポートしています。

- Call.connect、offhook

- answer、disconnect、drop、hold、unhold
- consult、transfer、conference、redirect
- playdtmf、deviceData

JTAPI は SIP を実行する電話で次のイベントをサポートしています。

- CiscoTermDeviceStateEv、RTP イベント、inService および OutOfService
- MediaTermConnDtmfEv（アウトバンドのみ）、転送開始および終了イベント、会議の開始および終了イベント、CiscoToneChangedEv および CiscoTermConnPrivacyChangedEv

SIP を実行する電話の動作は、次の点で SCCP を実行する電話と異なります。

- コール拒否：SIP を実行する電話に対してコールが発信されたとき、電話はそのコールを拒否することを選択できます。この場合、アプリケーションはその SIP 端末のアドレスに関する CallActive, ConnCreatedEv を受信した後に ConnDisconnectedEv を受信します。これは RP がコールを拒否した場合とほぼ同じです。
- メディアのないコンサルト コールに SIP フォンが含まれている場合は、接続後 1.5 秒以内に転送する必要があります。
- SIP を実行する電話では常に一括ダイヤルが使用されます。これはユーザが最初にオフフックしてから番号をダイヤルする場合も同じです。SIP フォンはすべての番号を収集してから、その番号を Cisco Unified Communications Manager に送信します。そのため、CallCtlConnDialingEv は常に、いずれかの設定済みダイヤル パターンに一致する桁数の番号が電話機で押された後に配信されます。
- MediaTermConnDtmfEv を受信するには、すべてのデバイスに対して「アウトバンド DTMF」を設定する必要があります。

CTI ポート、ルート ポイント、および SCCP を実行する電話に関するイベントは変更されていません。

トランスポートとして UDP を使用して SIP を実行する Cisco Unified IP Phone 7900 シリーズ モデルと Cisco Unified Communications Manager の間の接続が失われると、JTAPI アプリケーションはその電話機に対して定義されている端末とアドレスに関するイベント CiscoTermOutOfServiceEv と CiscoAddrOutOfServiceEv を受信します。UDP ではその性質上、接続切れが検出されるまでに時間差が生じるため、アプリケーションにアウト オブ サービス イベントが通知された後でも SIP を実行する Cisco Unified IP Phone 7900 シリーズは登録されているように見える場合があります。

Cisco Unified IP Phone 7960、7940、および SIP を実行する Cisco Unified IP Phone 7900 シリーズ以外が制御リストに含まれている場合、オブザーバ（オブザーバとコール オブザーバの両方）がアドレスまたは端末に追加されて CiscoTermRestrictedEv がプロバイダー オブザーバに配信されると、例外がスローされます。これらのイベントの原因は CiscoRestrictedEv.CAUSE_UNSUPPORTED_PROTOCOL です。

端末が SCCP を実行する電話か SIP を実行する電話かを示す新しいインターフェイス getProtocol() が CiscoTerminal で公開されました。getProtocol() が返す値は CiscoTerminalProtocol で定義されています。

CiscoCall に定義されている次の新しいインターフェイスにより、アプリケーションは外部の SIP エンティティの URL 情報を取得できます。

Public interface CiscoCall

CiscoPartyInfo	**getLastRedirectingPartyInfo()**
CiscoPartyInfo	**getCurrentCallingPartyInfo()**

CiscoPartyInfo	**getCurrentCalledPartyInfo()**
CiscoPartyInfo	**getCalledPartyInfo()**

Public interface CiscoPartyInfo

CiscoUrlInfo	**getUrlInfo()**
Address	**getAddress()**
string	**getDisplayName()**
string	**getUnicodeDisplayName()**
boolean	**getAddressPI()**
boolean	**getDisplayNamePI()**
boolean	**getlocale()**

Public interface CiscoUrlInfo

int	**getUrlType()**
	`Final int URL_TYPE_TEL`
	`Final int URL_TYPE_SIP`
	`Final int URL_TYPE_UNKNOWN`

string	**getHost()**
string	**getUser()**
int	**getPort()**

int	**getTransportType()**
	`Final int TRANSPORT_TYPE_UDP`
	`Final int TRANSPORT_TYPE_TCP`

Public interface CiscoTerminal

int	**getProtocol ()**

CiscoTerminalProtocol

static int	`PROTOCOL_NONE` プロトコルのタイプが認識不能かまたは不明であることを示します。
static int	`PROTOCOL_SCCP` デバイスが SCCP を使用して Cisco Unified Communications Manager と通信を行っていることを示します。
static int	`PROTOCOL_SIP` デバイスが SIP を使用して Cisco Unified Communications Manager と通信を行っていることを示します。

Secure Real-Time Protocol 鍵情報

この機能は、暗号化されたメディア セッションの Secure Real-Time Protocol（SRTP）鍵情報を Cisco Unified Communications Manager をベースとするエンタープライズ システム内の認証済みエンド ポイント間で配信するために必要なメカニズムを提供します。この鍵情報を受け取るには、Cisco Unified Communications Manager の [管理者] ウィンドウで TLS Enabled フラグと SRTP Enabled フラグを設定し、JTAPI と CTIManager の間で TLS リンクを確立する必要があります。

鍵情報は CiscoRTPInputKeyEv および CiscoRTPOutputKeyEv で公開されます。アプリケーションでこれらのイベントを受信するには、CiscoTermEvFilter の rtpKeyEvenabled を有効にする必要があります。デフォルトでは、下位互換性を維持するためにフィルタは無効になっています。フィルタを有効にすると、アプリケーションは常に CiscoRTPInputKeyEv と CiscoRTPOutputKeyEv を受信します。これらのイベントのセキュリティ インジケータは、メディアが暗号化されているかどうか、および鍵が使用可能かどうかを示します。

CiscoRTPInputKeyEv には入力ストリームの鍵情報が含まれ、CiscoRTPOutputKeyEv には出力ストリームの鍵情報が含まれます。この鍵情報は、パケットの復号化してモニタリングを開始したり、メディアを録音する目的に使用できます。この鍵情報は改ざん可能な形で保存してはならず、不要になったときは鍵のエントリをゼロに戻すかクリアする必要があります。

この鍵情報には次の項目が含まれます。

- Key Length
- Master Key
- Salt Length
- Master Salt
- AlgorithmID
- isMKIPresent
- Key Derivation Rate

この機能拡張では、CTIPort と RoutePoint におけるセキュアなメディア終端もサポートされています。これを行うには、サポートされている暗号化アルゴリズムを CTIPort とルート ポイントの登録要求で渡します。TLS リンクが存在せず SRTP Enabled フラグが設定されていない場合は、アプリケーションにエラーが返されます。メディアが暗号化されるかどうかは、相手側がセキュア メディア対応かどうか、およびアルゴリズムのネゴシエーションが成功するかどうかによって決まります。

コール中のモニタリングに関しては、2 つのエンド ポイント間でコールが確立された後にアプリケーションが起動された場合、アプリケーションは Terminal.createSnapshot() とスナップショット イベント CiscoTermSnapshotEv を照会する必要があります。CiscoTermSnapshotCompletedEv が送信され、エンド ポイント間の現在のメディアがセキュアかどうかが示されます。アプリケーションは CiscoMediaCallSecurityIndicator を照会してコールのセキュリティ インジケータを取得できますが、この場合はイベントに鍵情報が含まれません。端末のどの回線にもコールが存在しない場合、アプリケーションは CiscoTermSnapshotCompletedEv のみを取得します。下位互換性を維持するため、これらのイベントはアプリケーションで CiscoTermEvFilter の snapShotRTPEnabled フィルタが有効にされた場合にのみ生成されます。

1 つのコールに関連する RTP イベントを相互に関連付けることができるように、すべての RTP イベントに CiscoRTPHandle が追加されます。下位互換性を維持するため、セキュアなメディアが存在しない場合、新しいイベントは生成されません。

SRTP の詳細については、SourceForge.net の『Secure RTP Library API Documentation』（David McGrew 著）を参照してください。

次のセクションでは、SRTP 鍵情報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。

Public Interface CiscoMediaEncryptionKeyInfo

int　**getAlgorithmID()**

このメソッドは現在のストリームのメディア暗号化アルゴリズムを返します。

int　**getIsMKIPresent()**

MKI が存在するかどうかを示す MKI インジケータ。MKI は鍵管理によって定義、通知、および使用されます。

int　**getKeyLength ()**

このメソッドはマスター鍵の長さを返します。

byte[]　**getKey()**

このメソッドはストリームのマスター鍵を返します。

int　**getSaltLength ()**

このメソッドはソルトの長さを返します。

byte[]　**getSalt()**

このメソッドはストリームのソルト鍵を返します。

int　**keyDerivationRate()**

このセッションの SRTP 鍵逸脱率を示します。

CiscoMediaSecurityIndicator

static int　MEDIA_ENCRYPTED_KEYS_AVAILABLE

終端されたメディアはセキュリティ保護されており、鍵を参照できます。

static int　MEDIA_ENCRYPTED_KEYS_UNAVAILABLE

メディアはセキュリティを確保したモードで終端されていますが、SRTP が Cisco Unified Communications Manager Administration User ウィンドウで有効にされていないため、鍵は参照できません。このアプリケーションで、TLS が存在しないか IPSec が設定されていない可能性があります。

static int　MEDIA_ENCRYPTED_USER_NOT_AUTHORIZED

メディアはセキュリティを確保したモードで終端されていますが、ユーザが鍵を取得する権限を持っていないため、鍵は参照できません。

static int　MEDIA_NOT_ENCRYPTED

メディアのこのコールは暗号化されていません。

CiscoRTPInputKeyEv

CiscoMediaEncryptionKeyInfo	**getCiscoMediaEncryptionKeyInfo ()**	プロバイダーが TLS リンクで開かれており、Cisco Unified Communications Manager User Administration でアプリケーションの SRTP を有効にするオプションが設定されている場合だけ、CiscoMediaEncryptionKeyInfo を返します。そうでない場合は、null を返します。
int	**getCiscoMediaSecurityIndicator()**	メディア セキュリティ インジケータを返します。次に示す CiscoMediaSecurityIndicator の定数のうち 1 つを返します。 MEDIA_ENCRYPTED_KEYS_AVAILABLE MEDIA_ENCRYPT_USER_NOT_AUTHORIZED MEDIA_ENCRYPTED_KEYS_UNAVAILABLE MEDIA_NOT_ENCRYPTED
CiscoCallID	getCallID ()	このイベントの送信時に CiscoCall が存在している場合、CiscoCallID オブジェクトを返します。CiscoCall がない場合、このメソッドは null を返します。
CiscoRTPHandle	getCiscoRTPHandle ()	CiscoRTPHandle オブジェクトを返します。アプリケーションは、CiscoProvider.getCall(CiscoRTPHandle) を使用してコール参照を取得できます。コール オブザーバが存在しない場合や、このイベントの配信時にコール オブザーバがなかった場合、CiscoProvider.getCall（CiscoRTPHandle）は null を返す可能性があります。

CiscoRTPOutputKeyEv

CiscoMediaEncryptionKeyInfo	**getCiscoMediaEncryptionKeyInfo ()**	プロバイダーが TLS リンクで開かれており、Cisco Unified Communications Manager User Administration でアプリケーションの SRTP を有効にするオプションが設定されている場合にだけ、CiscoMediaEncryptionKeyInfo を返します。そうでない場合は null を返します。
int	**getCiscoMediaSecurityIndicator()**	メディア セキュリティ インジケータを返します。次に示す CiscoMediaSecurityIndicator の定数のうち 1 つを返します。 MEDIA_ENCRYPTED_KEYS_AVAILABLE MEDIA_ENCRYPT_USER_NOT_AUTHORIZED MEDIA_ENCRYPTED_KEYS_UNAVAILABLE MEDIA_NOT_ENCRYPTED

CiscoCallID　`getCallID ()`

このイベントの送信時に CiscoCall が存在している場合、CiscoCallID オブジェクトを返します。CiscoCall がない場合、このメソッドは null を返します。

CiscoRTPHandle　`getCiscoRTPHandle ()`

CiscoRTPHandle オブジェクトを返します。アプリケーションは、CiscoProvider.getCall（CiscoRTPHandle）を使用してコール参照を取得できます。コール オブザーバが存在しない場合や、このイベントの配信時にコール オブザーバがなかった場合、CiscoProvider.getCall（CiscoRTPHandle）は null を返す可能性があります。

CiscoTermSnapshotEv

CiscoMediaCallMediaSecurityIndicator[]　**`getMediaCallSecurityIndicator ()`**

このデバイス上の各アクティブ コールにおけるメディアのセキュリティ状態を返します。

CiscoTermSnapshotCompletedEv

このイベントにはメソッドがありません。

CiscoMediaCallSecurityIndicator

int　**`getCiscoMediaSecurityIndicator()`**

メディア セキュリティ インジケータを返します。次に示す CiscoMediaSecurityIndicator の定数のうち 1 つを返します。

```
MEDIA_ENCRYPTED_KEYS_AVAILABLE
MEDIA_ENCRYPT_USER_NOT_AUTHORIZED
MEDIA_ENCRYPTED_KEYS_UNAVAILABLE
MEDIA_NOT_ENCRYPTED
```

CiscoCallID　`getCallID ()`

このイベントの送信時に CiscoCall が存在している場合、CiscoCallID オブジェクトを返します。CiscoCall がない場合、このメソッドは null を返します。

CiscoRTPHandle　`getCiscoRTPHandle ()`

CiscoRTPHandle オブジェクトを返します。アプリケーションは、CiscoProvider.getCall(CiscoRTPHandle) を使用してコール参照を取得できます。コール オブザーバが存在しない場合や、このイベントの配信時にコール オブザーバがなかった場合、CiscoProvider.getCall（CiscoRTPHandle）は null を返す可能性があります。

CiscoRTPInputStartedEv

CiscoRTPHandle　getCiscoRTPHandle ()

CiscoRTPHandle オブジェクトを返します。アプリケーションは、CiscoProvider.getCall（CiscoRTPHandle）を使用してコール参照を取得できます。コール オブザーバが存在しない場合や、このイベントの配信時にコール オブザーバがなかった場合、CiscoProvider.getCall（CiscoRTPHandle）は null を返す可能性があります。

CiscoRTPInputStoppedEv

CiscoRTPHandle　getCiscoRTPHandle ()

CiscoRTPHandle オブジェクトを返します。アプリケーションは、CiscoProvider.getCall(CiscoRTPHandle) を使用してコール参照を取得できます。コール オブザーバが存在しない場合や、このイベントの配信時にコール オブザーバがなかった場合、CiscoProvider.getCall（CiscoRTPHandle）は null を返す可能性があります。

CiscoRTPOutputStartedEv

CiscoRTPHandle　getCiscoRTPHandle ()

CiscoRTPHandle オブジェクトを返します。アプリケーションは、CiscoProvider.getCall（CiscoRTPHandle）を使用してコール参照を取得できます。コール オブザーバが存在しない場合や、このイベントの配信時にコール オブザーバがなかった場合、CiscoProvider.getCall（CiscoRTPHandle）は null を返す可能性があります。

CiscoRTPOutputStoppedEv

CiscoRTPHandle　getCiscoRTPHandle ()

CiscoRTPHandle オブジェクトを返します。アプリケーションは、CiscoProvider.getCall(CiscoRTPHandle) を使用してコール参照を取得できます。コール オブザーバが存在しない場合や、このイベントの配信時にコール オブザーバがなかった場合、CiscoProvider.getCall（CiscoRTPHandle）は null を返す可能性があります。

CiscoTermEvFilter

boolean　`getSnapshotEnabled ()`

端末の CiscoTermSnapshotEv および CiscoTermSnapshotCompletedEv の有効または無効のステータスを返します。

void　`setSnapshotEnabled (boolean enabled)`

CiscoTermSnapshotEv の有効または無効のステータスを設定します。無効な場合、CiscoTermSnapshotEv および CiscoTermSnapshotCompletedEv はアプリケーションに送信されません。

boolean　`getRTPKeyEvEnabled ()`

CiscoRTPInputKeyEv および CiscoRTPOutputKeyEv の有効または無効のステータスを返します。

void　`setRTPKeyEvEnabled (boolean enabled)`

CiscoRTPInputKeyEv および CiscoRTPOutputKeyEv の有効または無効のステータスを設定します。

CiscoTerminal

void　**`createSnapshot ()`** throws InvalidStateException

このは、端末における現在のアクティブ コールのセキュリティ状態を格納した CiscoTermSnapshotEv を生成します。このメソッドにアクセスするには、端末が CiscoTerminal.IN_SERVICE 状態になっており、CiscoTermEvFilter.setSnapshotEnabled() が True に設定されている必要があります。

CiscoMediaTerminal

void	**register**(CiscoMediaCapability[] capabilities, int[] supportedAlgorithms) CiscoMediaTerminal は CiscoTerminal.UNREGISTERED 状態になっている必要があり、プロバイダーは Provider.IN_SERVICE 状態になっている必要があります。このインターフェイスではセキュアなメディアでの動的登録が可能です。アプリケーションがこのメソッドを呼び出さない場合、メディアはセキュアでないモードで終端されます。
void	**register**(java.net.InetAddress address, int port, CiscoMediaCapability[] capabilities, int[] algorithmIDs) CiscoMediaTerminal は CiscoTerminal.UNREGISTERED 状態になっている必要があり、プロバイダーは Provider.IN_SERVICE 状態になっている必要があります。このインターフェイスではセキュアなメディアでの静的登録が可能です。アプリケーションがこのインターフェイスを登録しない場合、メディアのセキュリティは確保されません。AlgorithmIDs はこの CTIPort がサポートする SRTP アルゴリズムを示します。AlgorithmID は、CiscoSupportedAlgorithm のいずれか 1 つだけになります。

CiscoRouteTerminal

void	register (CiscoMediaCapability)[] capabilities, int registrationType, int[] algorithmIDs CiscoRouteTerminal は CiscoTerminal.UNREGISTERED 状態になっている必要があり、プロバイダーは Provider.IN_SERVICE 状態になっている必要があります。デフォルトでは、メディアはセキュアでないモードで終端されます。AlgorithmIDs はこの CTIPort がサポートする SRTP アルゴリズムを示します。AlgorithmID は、CiscoSupportedAlgorithm のいずれか 1 つだけになります。

CiscoSupportedAlgorithm の定数

AES_128_COUNTER

SIP REFER または REPLACE

REFER は RFC 3515 で定義されている SIP メソッドです。REFER メソッドは、受信者（Request-URI によって識別される Referee）に対して、要求で提供されたコンタクト情報を使用して第三者（ターゲットと呼ばれる）にコンタクトするよう指示します。この REFER メソッドを使用すると、REFER を送信した端末（Referrer）はリクエストの結果を受け取ることができます。

Cisco Unified Communications Manager は Back-To-Back User Agent（B2BUA）として、内部ダイアログと外部ダイアログの両方で、Referee の代わりにインバウンド REFER を処理します。REFER の結果として、Cisco Unified Communications Manager は Referee と Refer To Target の間にコールを作成します。Referrer と Referee の間に既存のコールが存在する場合は、REFER が完了した後に Referrer 側でコールがドロップされます。

REPLACES 機能は既存の SIP ダイアログを新しいダイアログに置き換えます。SIP ダイアログが 2 つの SIP ユーザ エージェント間のコールであるのに対し、Cisco Unified Communications Manager ダイアログは半コール（コールレッグ）です。REPLACES 機能は REFER または INVITE によって開始されます。Cisco Unified Communications Manager は REPLACES ヘッダーの受信者の代わりに REPLACES 要求を処理します。この要求は新しいダイアログに関連付けられ、REPLACES ヘッダーで指定されている既存のダイアログ（コール）の相手側に代わりにコールに参加したい通話者が要求者になります。Cisco Unified Communications Manager は REPLACES ヘッダーで指定されているダイアログ（コール）の接続を解除して、要求者を接続します。

JTAPI は、Cisco Unified Communications Manager の REFER 機能と REPLACE 機能によって発生するコール イベントを JTAPI コール モデルでモデル化するように拡張されており、これらのコール イベントをアプリケーションで処理することを可能にします。JTAPI には REPLACES 要求を使用して REFER または REFER/INVITE を開始するためのインターフェイスはありませんが、これらのコール イベントを適切に処理することは可能です。

これら 2 つの機能には下位互換性があります。JTAPI は REFER/REPLACE によって発生したイベントを CAUSE_NORMAL で提供します。機能固有の原因は、新しいインターフェイスである CiscoCallEv.getCiscoFeatureReason() から取得できます。

（注）　このインターフェイスは現行機能と新機能の両方に関して機能固有の原因を提供しますが、将来のリリースでは下位互換性はなくなる予定です。アプリケーションでこのインターフェイスを使用する場合は、将来的な下位互換性の問題を回避するために、デフォルトの処理を実装する必要があります。

次のセクションでは、SIP の REFER/REPLACE 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。

REFER/REPLACE に対して提供される CAUSE

JTAPI は、REFER/REPLACES に起因するイベントに対して、CAUSE_NORMAL を提供します。アプリケーションでは CiscoCallEv.getCiscoFeatureReason() を使用して機能固有の原因を取得してください。

CiscoCallEv で提供されるインターフェイス

このインターフェイスは、JTAPI コール イベントに CiscoFeatureReason を提供します。転送などの既存の機能は、現状どおり元の CiscoCallEv.getCiscoCause() インターフェイスから CiscoCause を受け取ります。このインターフェイスは転送の REASON_TRANSFER を提供します。

```
com.cisco.jtapi.extensions
Interface CiscoCallEv
```

int　　**getCiscoFeatureReason()**

このインターフェイスは Cisco Unified Communications Manager 機能原因を返します。

Interface CiscoFeatureReason

JTAPI は、機能に起因する Call イベントで CiscoFeatureReason を提供します。CiscoFeatureReason は、Cisco Unified Communications Manager の既存の機能に対してだけでなく新規の機能に対しても提供されます。REFER および REPLACES の機能では、原因は REASON_REFER および REASON_REPLACES になります。このインターフェイスは、今後導入される新機能に対して新しい原因を提供するため、下位互換性はありません。

CiscoFeatureReason を使用するアプリケーションでは、今後のリリースで新しい原因コードが返される可能性があることを考慮してください。また、アプリケーションの下位互換性を保つためにはデフォルトの動作を実装する必要があります。

CiscoFeatureReason を使用するアプリケーションでは、今後のリリースで新しい原因コードが返される可能性があることを考慮してください。また、下位互換性を保つためにはデフォルトの動作を実装する必要があります。

Public interface CiscoFeatureReason

static int　REASON_REFER

REFER に対して Cisco Unified Communications Manager から送信されたイベントに対して返される原因。

static int　REASON_REPLACE

REPLACE に対して Cisco Unified Communications Manager から送信されたイベントに対して返される原因。

SIP 3XX リダイレクション

SIP リダイレクト サーバは SIP 要求を受信すると 3xx（リダイレクション）応答を返して、別の SIP アドレスにコンタクトするようクライアントに指示します。この機能拡張では、RFC 3261 に準拠した Cisco Unified Communications Manager リダイレクション（3xx）コール制御プリミティブがサポートされます。Cisco Unified Communications Manager リダイレクション プリミティブは SIP 3xx 応答を処理して、3xx 応答に含まれる各コンタクト アドレスに対して順次ハントを実行します。Cisco Unified Communications Managerまた、この操作の結果として発生する機能対話も処理します。Cisco Unified JTAPI は、このプリミティブの結果として Connection と TerminalConnection がいつ作成され、破棄されたかを示す新しい原因コードをすべての CallEv で公開します。

リダイレクション プリミティブがターゲットをハントしているときに JTAPI Redirect や CallForwardNoAnswer などの機能対話が発生した場合は、lastRedirectAddress が変更されることがあります。ターゲットが応答せず、Cisco Unified Communications Manager リダイレクトがコールの制御を引き継いで次のターゲットにコールを送信する場合、lastRedirectAddress は最初に SIP 3xx 応答を返した通話者に設定されます。

SIP 3xx 応答に Diversion ヘッダーが含まれている場合、3xx プリミティブは lastRedirectParty に対して Diversion ヘッダーの最初の値を使用し、JTAPI アプリケーションは Diversion ヘッダー要素を lastRedirectAddress として認識します。

JTAPI では、下位互換性を維持するために、CM_REDIRECTION という原因を含む新しい API、CiscoCallEv.getCiscoFeatureReason() を CiscoCallEv インターフェイスで公開しています。

（注） アプリケーションを開発する際には、この API から新しい機能固有の原因コードが返される可能性があることを考慮し、認識不能な原因コードに対してはデフォルトの動作を実行するようにしてください。

次のセクションでは、SIP 3XX Redirection 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。

Public interface CiscoFeatureReason

static int　**REASON_CM_REDIRECTION**

この原因は、イベントが Cisco Unified Communications Manager の CM_REDIRECTION プリミティブからの 3xx 応答の結果であることを示します。

CiscoCallEv

int　`getCiscoFeatureReason()`

このイベントに対する機能固有の原因。アプリケーションは、未知の原因を処理してデフォルト動作を提示できる必要があります。これは将来新しい原因が追加され、このインターフェイスの下位互換性がなくなる可能性があるためです。

端末とアドレスの制限

この機能拡張では、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制限されます。

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知ることができます。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送信されます。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送信されます。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送信されます。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送信されます。制限リストから端末が削除されと、CiscoTermActivatedEv と、対応するアドレスの CiscoAddrActivatedEv 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することはできません。

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送信されます。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送信されます。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送信されます。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送信されます。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送信されます。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アプリケーションに CiscoAddrActivatedEv が送信され、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リストに追加されている場合、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。一部の共用回線だけ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だけがイン 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持ち、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 イベントは受信できません。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 Connection 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であり、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、このイベントを受信します。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 を受信し、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、このイベントを受信します。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 を受信します。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、このイベントを受信します。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は受信できません。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。

Unicode のサポート

Cisco Unified Communications Manager リリース 5.0 は、Unicode 対応の IP フォンで Unicode の表示名をサポートします。表示名には ASCII 名と Unicode 名を設定できます。JTAPI は、すべての名前を Unicode 形式と ASCII 形式で受け取り、アプリケーションが表示名を Unicode で取得できるように、2 つの新しいインターフェイス getCurrentCalledPartyUnicodeDisplayName および getCurrentCallingPartyUnicodeDisplayName を提供します。また、コール プログレス中に Unicode 表示名を取得する機能もあります。

JTAPI は、デバイス登録イベントとデバイス イン サービス イベントの中で、アプリケーションによって制御される IP フォンのエンコード機能を CTI から受け取ります。また、ロケールおよび言語グループ情報をデバイス情報の応答で受け取ります。そして、IP フォンのロケール、代替スクリプト、および Unicode 機能を取得するためのインターフェイスをアプリケーションに提供します。CiscoTerminal および CiscoTermInServiceEv インターフェイスが拡張され、CiscoTerminal がイン サービス状態のときに、アプリケーションの制御リストにある電話にこれらの情報を提供します。

JTAPI は、コール内のすべての通話者の代替スクリプト情報を受け取り、現在の発信者と現在の着信者の言語グループを取得するためのインターフェイスを、アプリケーションに提供します。これらの情報を取得するための 2 つのインターフェイス getCurrentCallingPartyLanguageGroup および getCurrentCalledPartyLanguageGroup が、CiscoCall で使用できます。また、アプリケーションは、現在の発側アドレスと着側アドレスの表示名を ASCII 形式と Unicode（UCS-2）形式の両方で受信します。

JTAPI の Unicode サポートには次の変更も含まれます。

- CiscoCall インターフェイスの変更
- CiscoLocales インターフェイスの変更
- CiscoTerminal / CiscoTerminalInServiceEv インターフェイスの変更

リリース 5.0 にアップグレードした後は、アプリケーションでユーザ名 / パスワードの再設定が必要になる可能性があります。

次のセクションでは、Unicode のサポート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。

CiscoCall インターフェイスの変更

次に示す CiscoCall の新しいメソッドにより、アプリケーションは Unicode 表示名と対応するロケールを取得できます。

```
/**
     * This interface returns the unicode display name of the current called party
     * in the call.
     */
    public String getCurrentCalledPartyUnicodeDisplayName();

    /**
     * This interface returns the locale of the current called party unicode
     * display name. CiscoLocales interface lists the supported locales.
     */
    public int getCurrentCalledPartyUnicodeDisplayNamelocale();

    /**
     * This interface returns the unicode display name of the current calling party
     * in the call.
     */
    public String getCurrentCallingPartyUnicodeDisplayName ();

    /**
     * This interface returns the locale of the current called party
     * unicode display name
     */
    public int getCurrentCallingPartyUnicodeDisplayNamelocale();
```

CiscoLocales

CiscoLocales インターフェイスは、Cisco Unified JTAPI でサポートされているすべてのロケールを列挙します。

（注） 最新のリリースでサポートされているすべてのロケールのリストは、CiscoLocales の man ページを参照してください。

```
public interface CiscoLocales
{
    public static final int LOCALE_ENGLISH_UNITED_STATES;
    public static final int LOCALE_FRENCH_FRANCE;
    public static final int LOCALE_GERMAN_GERMANY;
    public static final int LOCALE_RUSSIAN_RUSSIA ;
    public static final int LOCALE_SPANISH_SPAIN ;
    public static final int LOCALE_ITALIAN_ITALY ;
    public static final int LOCALE_DUTCH_NETHERLAND ;
    public static final int LOCALE_NORWEGIAN_NORWAY ;
    public static final int LOCALE_PORTUGUESE_PORTUGAL;
    public static final int LOCALE_SWEDISH_SWEDEN ;
```

```
    public static final int LOCALE_DANISH_DENMARK
    public static final int LOCALE_JAPANESE_JAPAN;
    public static final int LOCALE_HUNGARIAN_HUNGARY ;
    public static final int LOCALE_POLISH_POLAND ;
    public static final int LOCALE_GREEK_GREECE ;
    public static final int LOCALE_TRADITIONAL_CHINESE_CHINA;
    public static final int LOCALE_SIMPLIFIED_CHINESE_CHINA;
    public static final int LOCALE_KOREAN_KOREA;
}
```

CiscoTerminalInServiceEv インターフェイス

int　**getLocale()**

このメソッドはこの端末に関連付けられた現在のロケール情報を返します。

int　**getSupportedEncoding ()**

このメソッドは、この端末が Unicode をサポートしている場合に true を返します。

CiscoTerminal インターフェイス

int　**getLocale()**

このメソッドはこの端末に関連付けられた現在のロケール情報を返します。このメソッドにアクセスするには、CiscoTerminal が CiscoTerminal.IN_SERVICE 状態になっている必要があります。

int　**getSupportedEncoding ()**

このメソッドは、この Terminal の Unicode 機能を返します。このメソッドにアクセスするには、CiscoTerminal が CiscoTerminal.IN_SERVICE 状態になっている必要があります。

getSupportedEncoding () は、CiscoTerminal で定義されている次のいずれかの結果を返します。

```
/**
  * Indicates the <Code>CiscoTerminal.getSupportedEncoding ()</CODE>
  * for this Terminal is UNKNOWN
*/
 public final static int UNKNOWN_ENCODING = 0;

/**
  * Indicates the <Code>CiscoTerminal.getSupportedEncoding ()</CODE>
  * for this is NOT_APPLICABLE.
  * This is valid for only CiscoMediaTerminals and RoutePoints
*/
 public final static int NOT_APPLICABLE = 1;

/**
  * Indicates the <Code>CiscoTerminal.getSupportedEncoding ()</CODE> for this
  * Terminal is ASCII and this terminal supports only ASCII_ENCODING
*/
 public final static int ASCII_ENCODING = 2;
```

```
/**
  * Indicates the <Code>CiscoTerminal.getSupportedEncoding ()</CODE>
  * for this Terminal is UCS2UNICODE_ENCODING
 */
 public final static int UCS2UNICODE_ENCODING = 3;
```

Linux、Windows および Solaris へのインストール

この機能は、Windows、Linux、および Solaris プラットフォームにおける Cisco Unified JTAPI クライアントのインストール/アンインストール手順を統一できます。以前のリリースでは、Windows プラットフォーム用の Cisco Unified JTAPI クライアント インストーラしか提供されていなかったため、Linux および Solaris マシンへのインストールは手動で行う必要がありました。

Cisco Unified JTAPI Install は InstalledShieldMultiPlatform（ISMP）を使用して開発されています。Linux 版と Solaris 版のインストーラはバイナリ ファイル（.bin）として提供され、Windows 版のインストーラは実行可能（.exe）ファイルとして提供されます。これら 3 種類のインストーラはすべて、Cisco Unified Communications Manager のプラグイン ページから入手できます。

Windows プラットフォームの JTAPI クライアントについては、このリリースへのアップグレードおよびこのリリースからのダウングレードに関して次のような変更点があります。

- 新しいインストーラは、以前のリリースへのダウングレードをサポートしていません。以前のバージョンの JTAPI クライアントを再インストールする場合は、事前に JTAPI を手動でアンインストールする必要があります。
- 新しいインストーラを使用して以前のリリースからアップグレードする場合は、現在のバージョンの JTAPI クライアントをアンインストールするよう求めるメッセージが表示され、以前のバージョンをアンインストールした後にインストールが開始されます。

サイレント インストール呼び出しを使用すると、アプリケーションのインストーラに JTAPI Installer を組み込むことができます。インストーラをサイレント モードで起動するアプリケーションでは、次のいずれかのコマンドを使用できます。

- Windows :　CiscoJTAPIClient.exe –silent
- Linux :　CiscoJTAPIClient-linux.bin –silent
- Solaris（Sparc）: CiscoJTAPIClient-solarisSparc.bin –silent
- Solaris（X86）: CiscoJTAPIClient-solarisX86.bin –silent

コマンドラインから JTAPIInstaller を実行する必要があるアプリケーションの場合は、コマンド プロンプトから次のいずれかのコマンドを使用できます。

- Windows :　CiscoJTAPIClient.exe -console
- Linux :　CiscoJTAPIClient-linux.bin –console
- Solaris（Sparc）: CiscoJTAPIClient-solarisSparc.bin –console
- Solaris（X86）: CiscoJTAPIClient-solarisX86.bin –console

コマンドを使用する場合は、文字入力ベースのメニューによってインストール手順が示されます。GUI を使用してインストールする場合と同じオプションを使用できます。このタイプのインストールは、Linux のような非 GUI プラットフォームに最も適しています。

Linux プラットフォームで新規インストールまたはアップグレード/ダウングレードを実行すると、JTAPIInstaller がインストール先フォルダを自動的に検出してサイレント インストールを実行します。ただし、システムに以前のバージョンが存在する場合は、インストーラがアプリケーション パスを判断できないため、デフォルトの場所にフォルダが作成され、それらのフォルダにアプリケーションがインストールされます。

JTAPIInstaller の詳細については、第 4 章「Cisco Unified JTAPI のインストール」を参照してください。

Linux へのインストール

インストーラで新規インストールまたはアップグレードを実行すると、インストール先フォルダが自動的に検出されて、サイレント インストールが実行されます。ただし、システムに以前のバージョンが存在する場合は、インストーラがアプリケーション パスを判断できないため、デフォルトの場所にフォルダが作成され、そのフォルダにアプリケーションがインストールされます。

下位互換性

この機能には下位互換性はありません。CiscoJtapiClient インストールのコマンド ラインまたはサイレント インストール機能では、インストールのコマンド呼び出しを変更する必要があります。

サイレント インストール

JTAPIInstaller でサイレント インストール呼び出しを実行すると、アプリケーションのインストールに JTAPIInstaller を組み込むことができます。サイレント インストールを呼び出す方法については、セクション（5.1.2.14.1）で説明しています。システムに SD よりも前のバージョンが存在し、SD バージョンへのアップグレード操作が発生した場合、新しいインストーラによって、SD よりも前のバージョンのアンインストーラが自動的に呼び出されます。アプリケーションで JTAPI の SD よりも前のバージョンをサイレント モードでアンインストールする必要がある場合、次の方法でコマンド ラインからインストーラを起動する必要があります。CiscoJtapiClient.exe -W newversion.silent=1。このメソッドは、Windows プラットフォームにだけ適用されます。新しい SD インストーラのインストール前、つまりインストーラを実行する前に、アプリケーションで JTAPI のバージョンを検出する必要がある場合は、次のコマンドを使用して検出を行ってください。

CiscoJtapiClient.exe silent -W newversion.check=1 goto showversion このコマンドをコマンド ラインから実行すると、インストーラにより、現在のディレクトリに jtapiversion.txt というファイルが作成されます。これにより JTAPI バージョンのインストーラが a.b(c.d) 形式になります。このコマンドは、すべてのクライアント プラットフォーム（Linux、Solaris、Windows）に適用されます。

コマンドライン呼び出し

このモードは、Linux アカウントなど、GUI をサポートしていないシステムに JTAPI をインストールする場合に便利です。すべてのインストール手順が文字入力ベースのメニューによって示され、ユーザはインストール時の条件に基づいて一連の入力を行うように求められます。このモードでは GUI ベースのインストーラで提供されているその他のオプションもすべて使用できます。

JTAPI クライアント インストーラ

ISMP インストーラを使用したインストール時に、システムに JTAPI クライアント インストーラの前のリリースが存在する場合、ISMP インストーラにより、レジストリから自動的にこのリリースが検出され、前のリリースのアンインストーラが起動されます。ISMP インストーラが次の操作に進んだ場合、ユーザは前のリリースのアンインストール操作が正常に完了したことを確認する K 要があります。JTAPI クライアント インストーラの前のリリースのアンインストール中に何らかのエラーが発生しても、ISMP インストーラはそのエラーを検出できないため、インストールを続行します。

ISMP インストーラには、実際のインストールを続行する前に、古い JTAPI クライアント インストーラをアンインストールする上記のメソッドが導入されていますが、ユーザは次の手順に従うことをお勧めします。

手順

ステップ 1　[プログラムの追加と削除] に進みます。

ステップ 2　前のリリースのアンインストールを実行します。

この手順を推奨するのは、ユーザが直接現在のリリースのインストーラを起動しないようにするためです。インストーラは前のリリースのアンインストーラを呼び出します。アンインストールが完了すると、ユーザはシステムの再起動を今すぐ行うか、後で行うかを指定するように求められます。ユーザが今すぐに再起動する方を選択すると、サーバはリブートされ、同時にインストーラは新しいバージョンをインストールしようとします。前のリリースのアンインストーラと現在のリリースのインストーラは、異なる種類の IS/ISMP で作成されており、この 2 つのインストーラはリンクしていないため、この問題が発生します。

また、ISMP インストーラは、前のリリースのダウングレード操作もサポートしていません。

下位互換性

この機能には下位互換性はありません。

JRE 1.2 および JRE 1.3 のサポートの削除

このリリースの Cisco JTAPI クライアントは JRE 1.4 のみをサポートします。インターフェイスの変更はありませんが、JRE 1.2 と 1.3 はサポートされません。この変更の目的は、JDK 1.4 以降のみで利用可能な QoS をサポートすることです。また、jtapi.jar に含まれる Cisco の暗号化ファイルを利用するには JRE 1.3 以降が必要になります。このファイルは TCP を介して CTIManager にパスワードを送信するときに強力なパスワード暗号化アルゴリズムを提供します。JTAPI は、この機能の一部として、パスワードを送信する前に IMS（Identity Management System : Cisco Unified Communications Manager のコンポーネント）が提供する API を呼び出してパスワードを暗号化します。

また、JRE 1.4 を使用することで、JDK 1.4 に追加された新しい API を Cisco Unified JTAPI で利用できるようになります。以前のバージョンの JRE を使用するアプリケーションで Cisco Unified JTAPI を使用するには、JDK 1.4 をインストールする必要があります。

（注）　JTAPI アプリケーションへのインターフェイスに変更はありませんが、JTAPI.jar には RSA jsafe.jar（3.3）ファイルと Apache log4j-1.2.8.jar ファイルが含まれています。これらのバージョンの jsafe.jar（バージョン 3.3）および log4j-1.2.8.jar と互換性がない jar ファイルをアプリケーションで使用している場合、どちらのファイルがクラスパスで先に指定されているかによって、JTAPI またはアプリケーションが正常に動作しないことがあります。

また、この移行により、JTAPIPreferences とサンプル アプリケーションの MS-JVM に対する依存関係も解消されました。[JTAPI Preferences] ダイアログボックスの [詳細設定] タブに次の 2 つの設定パラメータが新しく追加されました。

- JTAPI Post Condition Timeout
- Use Progress As Disconnected

下位互換性

この機能には下位互換性はありません。

スーパープロバイダーと変更通知

このリリースの JTAPI に含まれるスーパープロバイダー関連の機能拡張では、主に次の点が変更されています。

プロバイダーがオープンされた後に Cisco Unified Communications Manager Administration で「Superprovider privilege」が無効に設定されると、そのことが CTI 変更通知イベントを通じて JTAPI に通知され、オープンされているデバイスのうち制御リストにないものがすべてクリーン アップされます。

JTAPI は、この変更を「CiscoProviderCapabilityChangedEvent」を使用してアプリケーションに通知します。この新しいイベントはフラグが変更されたときに発行され、フラグが有効になったか無効になったかを示します。制御リストにないデバイスがスーパープロバイダー モードでオープンされて制御リストに追加された場合、JTAPI はそのデバイスを自身の制御リストに追加します。

- このフラグが有効な「CiscoProviderCapabilityChangedEvent」を通常のアプリケーションが受信した場合は、スーパープロバイダー特権が付与されていることを意味しているので、このアプリケーションは自身の制御リストにないデバイスの取得を開始できます。
- スーパープロバイダー アプリケーションが、Superprovider フラグが無効な「CiscoProviderCapabilityChangedEvent」を受信した場合は、このアプリケーションのスーパープロバイダー特権が解除されたことを意味します。その後、次の一連のイベントが発生します。
  - Provider Out of Serivce（OOS）イベントがアプリケーションに配信され、そのアプリケーションが取得 / オープンしたデバイスがすべてクローズされます。
  - 取得 / オープンされたデバイスのうち制御リストにないすべてのデバイスについて CiscoTermRemovedEv がアプリケーションに配信されます。
  - JTAPI が通常のユーザとして CTI への再接続に成功すると、Provider inService イベントがアプリケーションに配信されます。
  - デバイスと回線の情報がアプリケーションに配信されます。
  - プロバイダーが OOS に移行する前にオープンしていたすべての制御対象デバイスについて CiscoTermCreatedEv がアプリケーションに配信されます。
- Cisco Unified Communications Manager Administration で「パーク DN モニタリング」フラグが変更された場合、JTAPI は「CiscoProviderCapabilityChangedEvent」を使用してアプリケーションに通知します。
  - このフラグが有効なイベントを受信したアプリケーションは、パーク DN を制御するための登録機能を実行します。
  - このフラグが無効に設定されているイベントをアプリケーションが受信した場合、JTAPI は再び「CiscoProviderCapabilityChangedEvent」を使用してアプリケーションに通知し、すべてのパーク DN アドレスをクローズします。
- Cisco Unified Communications Manager Administration で「change calling party number」フラグが変更されると、JTAPI はそのことを「CiscoProviderCapabilityChangedEvent」を使用してアプリケーションに通知します。
  - このフラグが有効なイベントを受信したアプリケーションは、発信者番号を変更できるようになります。
  - このフラグが無効なイベントを受信したアプリケーションは、発信者番号を変更できません。

  アプリケーション側でも、このフラグが無効な場合は発信者番号を変更しないようにする必要があります。

- スーパープロバイダーが制御リストにないデバイスをオープン / 取得した後に Cisco Unified Communications Manager Administration でそのデバイスが削除された場合、JTAPI は端末オブジェクトをクローズして、そのデバイスの CiscoTermRemovedEvent をアプリケーションに送信します。

インターフェイスの変更

スーパープロバイダーと変更通知の拡張の一部として、JTAPI では次の API をアプリケーションに公開しています。その結果、スーパープロバイダーのための JTAPI 実装と特定のプロバイダー機能の処理が変更されています。このリリースの JTAPI に含まれるスーパープロバイダー関連の機能拡張としては、JTAPI の QBE インターフェイス、JTAPI の動作に関する変更、アプリケーションに公開される新しい API があります。

JTAPI は、CiscoProviderCapabilityChangedEv を次の形式でアプリケーションに配信します。アプリケーションでは、JTAPI から送信されるこの新しいイベントを受信し、処理できるようにしてください。

```
public interface CiscoProviderCapabilityChangedEv {
    public CiscoProviderCapabilities getCapability ();
}
```

CiscoProviderCapabilities には、プロバイダーの発信者変更特権を設定するための次の新しいメソッドがあります。

```
public boolean canModifyCallingParty();
public void setCanModifyCallingParty(boolean value);
```

CiscoProviderCapabilityChangedEv は、適切なフラグ値とともにアプリケーションに配信されます。

その後、次の一連のイベントが発生します。

- JTAPI は、プロバイダー OOS イベントをアプリケーションに送信し、デバイス / 回線 OOS を制御リスト内のオープンされているデバイスと回線に送信します。
- 次に、JTAPI は CTI への再接続を試みます。
  - 再接続に成功した場合、JTAPI はプロバイダー inService イベントを送信し、以前にオープンされていた制御リスト内のすべてのデバイスを再びオープンします。
  - 再接続に失敗した場合、JTAPI はプロバイダーをシャットダウンし、ProviderClosedEvent を送信します。
- スーパープロバイダー特権が追加された場合、JTAPI は CiscoProviderCapabilityChangedEv を適切なフラグ値とともにアプリケーションに送信します。
  - MonitorParkDN フラグが有効になっている場合、JTAPI はパーク DN 監視フラグを true に設定して CiscoProviderCapabilityChangedEv を送信します。
  - MonitorParkDN フラグが無効になっている場合、JTAPI はパーク DN 監視フラグを false に設定して CiscoProviderCapabilityChangedEv を送信します。

    また、JTAPI はすべてのパーク DN アドレスをクローズし、CiscoAddrRemovedEv をアプリケーションに配信します。
- ModifyCgPn フラグが変更された場合、JTAPI はプロバイダー オブジェクト内でフラグを設定します。このフラグがリダイレクトのシナリオの中で調べられ、それに応じてアプリケーションによる発信者の変更権限が許可または禁止されます。

  また、JTAPI は CgPn を変更するためのフラグを設定して CiscoProviderCapabilityChangedEv を配信します。

CiscoProvider インターフェイス

boolean　`hasSuperproviderChanged()`

スーパープロバイダー特権が変更されたかどうかをアプリケーションに知らせます。

boolean　`hasModifyCallingPartyChanged()`

ModifyCgPn 特権が変更されたかどうかをアプリケーションに知らせます。

boolean　`hasMonitorParkDNChanged()`

パーク DN 監視特権が変更されたかどうかをアプリケーションに知らせます。

下位互換性

この機能には下位互換性はありません。

代替スクリプトのサポート

一部の機種の IP フォンでは、電話機上で設定可能なロケールに対応するデフォルト スクリプト以外の代替言語スクリプトがサポートされています。たとえば、日本語ロケールには 2 つの表記スクリプトがあります。一部の機種では Katakana スクリプト（デフォルト）しかサポートされていませんが、その他の機種ではデフォルト スクリプトと Kanji スクリプト（代替スクリプト）の両方がサポートされています。アプリケーションは電話機に表示用のテキスト情報を送信できるため、電話機がどの代替スクリプトをサポートしているかを知る必要があります。

新しい getAltScript() メソッドは、監視対象のデバイスの代替スクリプト情報を提供します。現在存在する代替スクリプトは日本語ロケールの Kanji だけです。

JTAPI では、代替スクリプト情報を提供する CiscoTerminal 用の新しいメソッドが提供されています。

java.lang.String　`getAltScript()`

現在サポートされている代替スクリプトは日本語ロケールの Kanji だけです。空の文字列が返された場合、代替スクリプトが設定されていないか、端末が代替スクリプトをサポートしていないことを表します。

下位互換性

代替スクリプトの機能は JTAPI の下位互換性に影響を与えることはありません。

半二重メディアのサポート

現在、JTAPI メディア イベントの CiscoRTPInputStarted、CiscoRTPOutputStarted、CiscoRTPInputStopped、および CiscoRTPOutputStopped では、メディアが半二重（受信のみ / 送信のみ）か全二重（受信と送信の両方）かは示されません。

この機能拡張により、JTAPI メディア イベントでこの情報を提供する機能が追加されます。JTAPI では、上記のメディア イベントでメディアが半二重か全二重かを問い合せるインターフェイスが提供されています。

半二重メディアのサポート機能は JTAPI の下位互換性に影響を与えることはありません。

新しいインターフェイス getMediaConnectionMode() が Cisco Unified JTAPI の RTP イベントに追加されています。このインターフェイスはメディアに応じて次の値を返します。

- CiscoMediaConnectionMode.NONE
- CiscoMediaConnectionMode.RECEIVE_ONLY
- CiscoMediaConnectionMode.TRANSMIT_ONLY
- CiscoMediConnectionMode.TRANSMIT_AND_RECEIVE

CiscoRTPInputStarted/StoppedEv は、RECEIVE_ONLY および TRANSMIT_AND_RECEIVE だけを返す必要があります。通常、NONE または TRANSMIT_ONLY が返されることはありません。返された場合は、イベントを無視するか、エラーを記録してください。

CiscoRTPOutputStarted/StoppedEv は、TRANSMIT_ONLY および TRANSMIT_AND_RECEIVE だけを返す必要があります。通常、値 NONE または RECEIVE_ONLY が返されることはありません。返された場合は、イベントを無視するか、エラーを記録してください。

CiscoMediaOpenLogicalChannedEv は、RECEIVE_ONLY および TRANSMIT_AND_RECEIVE だけを返す必要があります。通常、値 NONE または TRANSMIT_ONLY が返されることはありません。返された場合は、イベントを無視するか、エラーを記録してください。

public interface **CiscoRTPInputStartedEv**

int　`getMediaConnectionMode()`

CiscoMediaConnectionMode を返します。

public interface **CiscoRTPOutputStartedEv**

int　`getMediaConnectionMode()`

CiscoMediaConnectionMode を返します。

public interface **CiscoRTPInputStoppedEv**

int　`getMediaConnectionMode()`

CiscoMediaConnectionMode を返します。

public interface **CiscoRTPOutputStoppedEv**

int　　getMediaConnectionMode()

CiscoMediaConnectionMode を返します。

ネットワーク アラート

以前のリリースの Cisco JTAPI（Cisco JTAPI バージョン 1.4(x.y)）では、クラスタ外のアドレスへのコールを開始すると、遠端のアドレスに CallCtlConnNetworkReachedEv イベントと CallCtlConnNetworkAlertingEv イベントが配信されていました。

最近のバージョンの Cisco Unified Communications Manager（4.0 以降）および Cisco Unified JTAPI（2.0）では、これらのイベントは配信されていません。これらのバージョンでは、遠端アドレスの CallCtlConnection が OFFERED 状態から ESTABLISHED 状態に移行しました。以前のバージョンの Cisco Unified JTAPI では、「オーバーラップ送信」がオフの状態でゲートウェイ経由のコールが発信されたときに、遠端アドレスに対して CallCtlConnOfferedEv および CallCtlConnEstablishedEv が配信されていました。CallCtlConnNetworkReachedEv および CallCtlConnNetworkAlertingEv イベントはアプリケーションに配信されませんでした。

Cisco Unified Communications Manager 4.0 および 4.1 では、ネットワーク イベントを受信するために、ゲートウェイに対して設定されているルート パターンの「オーバーラップ送信を許可」フラグ、または jtapi.ini の「AllowNetworkEventsAfterOffered」パラメータをオンにする必要がありました。

Cisco Unified Communications Manager リリース 5.0 では、「オーバーラップ送信を許可」フラグが有効な場合、ゲートウェイの向こう側にある遠端アドレスの ConnCreatedEv、CallCtlConnNetworkReachedEv、CallCtlConnNetworkAlertingEv、および CallCtlConnEstablishedEv がアプリケーションに配信されます。

「オーバーラップ送信を許可」フラグが有効でない場合は、ゲートウェイの向こう側にある遠端アドレスの ConnCreatedEv、CallCtlConnOfferedEv、CallCtlConnNetworkReachedEv、CallCtlConnNetworkAlertingEv、および CallCtlConnEstablishedEv がアプリケーションに配信されます。

（注）　Cisco Unified Communications Manager リリース 5.0 では、「AllowNetworkEventsAfterOffered」は使用できません。上記のイベントは jtapi.ini のパラメータ設定に関係なく配信されます。

下位互換性

この機能には下位互換性はありません。

Linux の自動アップデータ

Linux ベースの JTAPI クライアント マシンでこの機能をサポートするために、アップデータのインターフェイスに次の変更が加えられました。このインターフェイスでは、アプリケーションは、コンポーネント名、プロバイダーの IP アドレス、ユーザ名、パスワードを提供する必要があります。アプリケーションは、コンポーネントをダウンロードする URL を指定する必要はありません。これは、Cisco Unified Communications Manager Administration のリリースごとに URL が異なる場合に、アップデータ アプリケーションに生じる問題を回避するために実現されました。

「Replace()」という名前の新しい API が、コンポーネント インターフェイスの一部として組み込まれています。この API を使用すると、古いコンポーネントを新しくダウンロードしたコンポーネントに簡単に置換できます。次のセクションでは、新しいインターフェイスへの変更後のアップデータの動作を定義しています。新しいアップデータは次のように動作します。

- API 署名は、古いものと同じものを使用する。
- 新しいバージョンの jar ファイルとして、アプリケーションの現在のフォルダに newjtapi.jar を作成する。
- 指定されたクラスパスの component.temp というファイルに現在の jtapi.jar をコピーする。
- 現在の jar ファイルを新しい jar ファイルに置換する。この処理が終わると、現在の jar ファイルは component.temp になり、新しい jar ファイルは jtapi.jar になります。アプリケーションでは、URL の取得を、アプリケーションで URL を指定する方法、または CiscoProvider で提供されている新しいインターフェイスを使用して URL を問い合わる方法のいずれかで行う、古いコンポーネント インターフェイスを使用することもできます。URL 情報を取得する必要がある API については、この機能のインターフェイスの概要に記載しています。この処理は、Unix と Windows の両方でサポートされています。

下位互換性

この機能には下位互換性はありません。

コールの選択状態

機能によって、または手動でコールを選択した場合、Cisco Unified JTAPI により CiscoTermConnSelectChangedEv イベントが送信されます。アプリケーションはこのイベントを受信すると、TerminalConnection.getSelectStatus() を使用して正確なコールの選択状態を取得できます。TerminalConnection.getSelectStatus() を呼び出すと、次の 3 つの状態のいずれかが返されます。

- CiscoTerminalConnection.CISCO_SELECTEDNONE：この選択状態は、コールが選択されていないことを示します。
- CiscoTerminalConnection.CISCO_ SELECTEDLOCAL：この選択状態は、コールが TerminalConnection で選択されていることを示します。
- CiscoTerminalConnection.CISCO_ SELECTEDREMOTE：コールがその共用回線で選択されている場合、Passive TerminalConnection がこの選択状態を受け取ります。

下位互換性

この機能には下位互換性はありません。

JTAPI バージョン情報

Cisco Unified Communications Manager Administration のリリース 5.0 に接続するには、JTAPI クライアントを、Cisco Unified Communications Manager Administration リリース 5.0 に付属した JTAPI の新しいバージョンにアップグレードする必要があります。JTAPI のバージョンは 3.0(X.Y) の形式で提供されます。この、X と Y は、サブリリースごとに異なります。アプリケーションは、JTAPI の古いリリースと接続できません。

コール転送

Cisco Unified JTAPI では、JTAPI 仕様に適合したコール転送機能の設定がサポートされています。Cisco Unified JTAPI 実装では、すべての転送特性はサポートされず、アドレスの FORWARD_ALL 属性だけがサポートされます。CallControlAddress オブジェクトの setForwarding、getForwarding および cancelForwarding のメソッドをアプリケーションによって呼び出すことが可能ですが、CallControlForwarding 命令は FORWARD_ALL のものだけが可能です。

コール パーク

Cisco Unified JTAPI は、ユーザのコール パークとの対話をサポートし、アプリケーションに適切なイベントを報告します。コールが IP フォンからパークされると、パーク アドレスに属する Connection は、Disconnected 状態に移行し、関連する TerminalConnection は、Dropped 状態に移行します。パーク番号に対してキュー状態の新規 Connection が作成されます。

コールのパークされたアドレスだけをアプリケーションが監視している場合、すべての既存の Connection は解除され、TerminalConnections が破棄されてコールは Invalid 状態に移行します。

パークの取得

コールが IP フォンからパークされると、電話機にパーク番号が表示されます。どの端末からでも、パーク番号をダイヤルすればコールのパークを解除できます。コールのパークが解除されると、パークを解除したアドレスに接続する新しいコールが作成されます。Queued 状態にある元のコールのパーク番号に対する CallControlConnection は、Disconnected 状態に移行します。

パーク リマインダ

パークされたコールが指定時間内に取得されないと、コールのパークされたアドレスにリマインダ コールが返され、パーク番号 Connection は Disconnected 状態に移行します。コールは再接続され、確立された状態に移行します。コールのパークされたアドレスに Talking 状態の TerminalConnection が作成されます。

パーク DN モニタ

Cisco Unified JTAPI アプリケーションは、コールがパークまたはパーク解除されたときにイベントを受信するように登録できます。この機能に登録すると、プロバイダー オブザーバに CiscoProvCallParkEv イベントが配信されます。この機能に登録するには、ユーザの Call Park Retrieval Allowed フラグがオンになっている必要があります。このフラグには Cisco Unified Communications Manager Administration のユーザ設定でアクセスできます。この機能に登録すると、クラスタ内のデバイスでコールがパークまたはパーク解除されるたびに、アプリケーションが CiscoProvCallParkEv イベントを受信します。

この機能の登録および登録解除は、次の新しいインターフェイスで行えます。

```
public interface CiscoProvider {
public void registerFeature ( int featureID ) throws
    InvalidStateException, PrivilegeViolationException;
public void unregisterFeature ( int featureID ) throws
    InvalidStateException;
}
```

featureID は CiscoProvFeatureID.MONITOR_CALLPARK_DN です。

CiscoJtapiExceptions

Cisco Unified JTAPI では、アプリケーションに CTI 生成のエラー コードが通知されます。これらのコードは、CTIManager に例外またはエラーがある場合に返されます。CTI で返されたエラー コードは、それぞれアプリケーションに伝搬されます。アプリケーションでは、例外オブジェクトの getErrorCode() メソッドの呼び出しによるエラー コードの抽出、getErrorName() メソッドの呼び出しによる CTI エラー コードの取得、およびメソッド getErrorDescription() の呼び出しによるエラー記述の取得が可能です。

getErrorName(int errorCode) メソッドと getErrorDescription (int errorCode) メソッドは推奨しません。今後のリリースで削除される予定です。Cisco では、アプリケーション ユーザがこれらのメソッドを使用しないことを推奨します。

Cisco Unified JTAPI インストールの国際対応

Cisco Unified JTAPI では、JTAPI のインストールとユーザ プリファレンスの UI に複数の言語がサポートされています。JTAPI が起動すると、インストール用の言語を選択するオプションが表示されます。言語を選択すると、その後のインストールの指示は選択した言語で表示されます。最初のオプションは常に英語です。語句が英語以外の言語に欠落している場合、指示はデフォルトの英語になります。詳細については、第 4 章「Cisco Unified JTAPI のインストール」を参照してください。

コールのクリア

Cisco Unified JTAPI アプリケーションでは、アクティブ コールを破棄せずにファントム コールをクリアできます。CiscoAddress には clearCallConnections メッセージがあり、これにより、Cisco Unified Communications Manager（以前の Cisco Unified Call Manager）上にアクティブ コールがないときにアプリケーションによってコールをクリアできます。

デバイス復旧

Cisco Unified JTAPI では、デバイスの自動復旧がサポートされています。

電話機のデバイス復旧

Cisco Unified IP Phone 7960 などのデバイスについては、リホーム機能がデバイス ファームウェアの一部に組み込まれています。プライマリ Cisco Unified Communications Manager の障害時に、コール上に Communications Manager がなくなると、電話機からバックアップ Cisco Unified Communications Manager への接続が試みられます。この移行は、「CTIManager の障害」（P.3-103）で説明するアウト オブ サービスおよびイン サービスのイベントの形式でアプリケーションに送信されます。

CTI Port や CTI RoutePoint などのファームウェアのない仮想デバイスでは、CTIManager または Cisco Unified JTAPI によってフェールオーバーが実行されます。

CTI RoutePoint

Cisco Unified Communications Manager サーバの障害時には、CTI RoutePoint のデバイス プール管理で定義された Cisco Unified Communications Manager サーバ グループから CTIManager がデバイスを復旧させます。プライマリ Cisco Unified Communications Manager サーバが復旧すると、CTIManager はそのプライマリ Cisco Unified Communications Manager 上のデバイスの復旧を試みます。このリホームは、デバイス上にコールがない場合に行われます（物理デバイスと同様）。

CTIManager の障害時には、Cisco Unified JTAPI がバックアップ CTIManager 上のデバイスを復旧させます。CiscoAddrOutOfServiceEv イベントと CiscoAddrInServiceEv イベントによって、デバイスのアベイラビリティがアプリケーションに通知されます。

CTI Port

アプリケーションによって登録される CTI Port には、電話機に似たメカニズムがあります。CTIPort へのシグナリングを処理している Cisco Unified Communications Manager に障害が起きると、CTIManager はこのデバイス用のデバイス プール管理に従ってそのサービスを復旧させます。CTIManager の障害時には、CTI Port がバックアップ CTIManager に接続されると、Cisco Unified JTAPI によって CTI Port が再登録されます。CTI Port の復旧後、CiscoAddrOutOfServiceEv イベントおよび CiscoTermOutOfServiceEv イベントと対応するイン サービスのイベントが送信されます。

これらのデバイスに対するメディア ストリーミングは、アプリケーションによって制御され、ポートがアウト オブ サービスの場合でもストリーミングは継続します。新しいコールを受け入れる準備が完了するまで、デバイスに対して新しいコールが発行されないように、アプリケーション側で処理します。

ディレクトリ変更通知

アプリケーションでは、デバイス追加、ユーザ制御リストからの削除、および Cisco Unified Communications Manager データベースからのデバイスの削除に関する通知を非同期に要求します。また、アプリケーションは回線変更およびデバイスに関する通知を受信します。この通知は Cisco Unified JTAPI に送信され、それぞれ AddressObserver と TerminalObserver 上の CiscoAddrCreatedEv、CiscoAddrRemovedEv、CiscoTermCreatedEv、および CiscoTermRemovedEv によってアプリケーションに伝搬されます。

（注）　回線変更通知を受信するために、CTIPort と CTIRoutePoint に対して必ずデバイスを登録してください。

呼び出し音の無効化または有効化

CiscoAddress の拡張により、デバイス上のすべての回線に対する呼び出し音のステータスをアプリケーションによって設定できます。管理用ウィンドウやその他のウィンドウから呼び出し音の設定が変更されても、イベントは生成されません。

転送と会議の拡張

JTAPI の転送イベントと会議イベントには、理解しにくい面があります。これは、参加者が一方のコールからもう一方のコールに移動する際に、JTAPI では、一方のコールから通話者を削除し、もう一方のコールに通話者を追加することで、このアクションを表現しているためです。これは、実際には通話者が移行中であるときに、コールから破棄されたという通知がアプリケーション上で受信されるため、混乱を招く場合があります。Cisco Unified JTAPI 実装では、アプリケーションによるこの機能の処理を容易にする、いくつかの追加イベントが定義されています。

転送

転送機能では、1 つのコール（転送コール）の参加者が別のコール（最後のコール）に転送されます。コールの参加者を移動すると、コール転送に関連付けられた Connection が DISCONNECTED 状態に移行し、これらの参加者の新しい Connection が最後のコールに作成されます。同様に、転送コールに関連するすべての TerminalConnection が Dropped 状態に移行し、最後のコール内に作成されます。Cisco の拡張機能では、定義で、転送に関連するイベントの開始と終了をマークします。

CiscoConnection.getConnectionID() メソッドを使用して、新旧の Connection に関する CiscoConnectionID を取得することで、新たに作成された Connection オブジェクトと古い Connection オブジェクトを関係付けることができます。Connection が一致する場合は、CiscoConnectionID.equals() メソッドを使用して比較したときに CiscoConnectionID オブジェクトが一致します。

CiscoTransferStartEv

このイベントでは、転送動作が開始されたことが示され、後に続くイベントはこの動作に関連しています。具体的には、Connection と TerminalConnection は両方とも削除され、転送結果として追加されます。

アプリケーションによって、転送に関与する 2 つのコール（転送コールと最後のコール）、およびこのイベントからの転送コントローラ情報を取得できます。JTAPI アプリケーションで転送コントローラを監視していない場合、このイベントでは転送コントローラ情報を使用できません。

CiscoTransferEndEv

このイベントでは、転送動作が終了したことが示されます。このイベントの受信後、アプリケーションでは、関与するすべての通話者が転送され、すべての Connection と TerminalConnection が最後のコールに移動されたと想定できます。

転送シナリオ

次のシナリオでは、転送に関与する 3 つの通話者をそれぞれ A、B、C で表します。

B が転送コントローラとなるコンサルト転送

コンサルト転送では、アプリケーションでコールを別のアドレスにリダイレクトし、転送者はリダイレクトの前に転送先と「コンサルト（情報交換）」できます。

- コール Call1 で A が B をコールする。
- B が応答し、コール Call2 で C とコンサルトする。
- B はコール Call2 をコール Call1 に転送する。

このタイプの転送を行うには、次の JTAPI メソッドを使用します。

- Call2.setTransferEnable(true)（このオプションのメソッドは、コール オブジェクト内で転送がデフォルトで有効であることを示します。）
- Call2.consult(TermConnB, C)
- Call1.transfer(Call2)

（注）　コンサルト転送時には、Call2.transfer(Call1) ではなく Call1.transfer(Call2) でコールが転送されます。

表 3-5 に、A と B のオブザーバが CiscoTransferStartEv と CiscoTransferEndEv の間に受信するコア イベントを示します。

表 3-5　A と B のオブザーバに対するコア イベント

原因メタ イベント	コール	イベント	フィールド
META_UNKNOWN	Call1	CiscoTransferStartEv	transferredCall=Call2 finalCall=Call transferController= TermConnB
META_CALL_TRANSFERRING	Call1	TermConnDroppedEv B CallCtlTermConnDroppedEv B ConnDisconnectedEv B CallCtlConnDisconnectedEv B	
META_CALL_TRANSFERRING	Call1	ConnCreatedEv C ConnConnectedEv C CallCtlConnEstablishedEv C TermConnCreatedEv C TermConnActiveEv C CallCtlTermConnTalkingEv C	
META_CALL_TRANSFERRING	Call2	TermConnDroppedEv B CallCtlTermConnDroppedEv B ConnDisconnectedEv B CallCtlConnDisconnectedEv B	
META_CALL_TRANSFERRING	Call2	TermConnDroppedEv C CallCtlTermConnDroppedEv C ConnDisconnectedEv C CallCtlConnDisconnectedEv C CallInvalidEv C	
META_UNKNOWN	Call2	CallObservationEndedEv	
META_UNKNOWN	Call1	CiscoTransferEndEv	transferredCall=Call2 FinalCall=Call1 transferController= TermConnB

A が転送コントローラとなる任意転送

任意転送では、コールを作成した方法に関係なく、あるコールを別のコールに転送できます。コンサルト転送とは異なり、最初にコンサルト メソッドを使用してコールの 1 つを作成する必要はありません。

- コール Call1 で A が B をコールする。

- A は Call1 を保留にする。
- コール Call2 で A が C をコールする。
- A は Call1 を Call2 に転送する。

このタイプの転送を行うには、次の JTAPI メソッドを使用します。

- コール Call1 を最後のコール Call2 に転送する Call2.transfer(Call1)、または
- コール Call2 を最後のコール Call1 に転送する Call1.transfer(Call2)

Call1.transfer(Call2) が呼び出されたと想定し、表 3-6 に、A と C のオブザーバが CiscoTransferStartEv と CiscoTransferEndEv の間に受信するコア イベントを示します。

表 3-6　　A と C のオブザーバに対するコア イベント

原因メタ イベント	コール	イベント	フィールド
META_UNKNOWN	Call1	CiscoTransferStartEv	transferredCall=Call2 finalCall=Call1 transferController= TermConnB
META_CALL_TRANSFERRING	Call1	TermConnDroppedEv B CallCtlTermConnDroppedEv B ConnDisconnectedEv B CallCtlConnDisconnectedEv B	
META_CALL_TRANSFERRING	Call1	ConnCreatedEv C ConnConnectedEv C CallCtlConnEstablishedEv C TermConnCreatedEv C TermConnActiveEv C CallCtlTermConnTalkingEv C	
META_CALL_TRANSFERRING	Call2	TermConnDroppedEv B CallCtlTermConnDroppedEv B ConnDisconnectedEv B CallCtlConnDisconnectedEv B	
META_CALL_TRANSFERRING	Call2	TermConnDroppedEv C CallCtlTermConnDroppedEv C ConnDisconnectedEv C CallCtlConnDisconnectedEv C CallInvalidEv C	

会議

JTAPI では、2 つのコールで会議を行うときには、一方のコールのすべての通話者をもう一方のコールに移動することが規定されています。所属していた通話者が移動され、それ以降、無効になるコールは、「マージ」コールまたは「コンサルト」コールと呼びます。マージされた通話者の移動先のコールを、これ以降「最終」コールと呼びます。通話者がマージ コールから最後のコールに移動するとき、アプリケーションでは、すべての通話者がマージ コールから破棄されたことを示すイベント、およびこれらの通話者が最終コールに再現されたことを示すイベントが受信されます。

新たに作成された Connection オブジェクトと古い Connection オブジェクトの相関を取るには、CiscoConection.getConnectionID() メソッドを使用して、すべての新旧の Connection に対する CiscoConnectionID オブジェクトを取得します。Connection が一致する場合は、CiscoConnectionID.equals() メソッドを使用して比較したときに CiscoConnectionID オブジェクトが一致します。

会議サポートは、次のメソッドに存在します。

- javax.telephony.callcontrol.CallControlCall.conference(Call)
- javax.telephony.callcontrol.CallControlCall.getConferenceController()
- javax.telephony.callcontrol.CallControlCall.getConferenceEnable()
- javax.telephony.callcontrol.CallControlCall.setConferenceController(TerminalConnection)
- javax.telephony.callcontrol.CallControlCall.setConferenceEnable(boolean)

Cisco の拡張

Cisco Unified JTAPI 実装では、会議の開始と終了を通知する、CiscoConferenceStartEv と CiscoConferenceEndEv の 2 つのイベントが追加されます。これらのイベントは、会議の起動時と終了時に送信されます。これらのイベントから、最後のコール、マージされた会議（コンサルト）コールおよび 2 つの制御対象の TerminalConnection（HELD 状態と TALKING 状態）へのハンドルが取得されます。

CiscoConferenceStartEv

このイベントは、call1.conference(call2) が呼び出されるか、または IP フォン上で 2 度目に [Conference（会議）] ボタンが押された場合に送信されます。ConferenceStartEv では、マージ プロセスの開始が通知されます。このイベントの後に、Cisco Unified Communications Manager の会議プロセスによって反映される一連のマージ イベントが続きます。

CiscoConferenceEndEv

このイベントは、ConferenceStartEv の送信後、マージ プロセスの終了時に送信されます。これにより、コンサルト（またはマージ）コールの最終電話会議へのマージ完了が通知されます。マージ コールは INVALID 状態になり、ObservationEndedEv がコール オブザーバに送信されます。

CiscoCall.setConferenceEnable()

Cisco Unified JTAPI 実装では、CiscoCall.setConferenceEnable() メソッドと CiscoCall.setTransferEnable() メソッドを使用することにより、会議または転送のどちらの機能を使用してコンサルト コールを開始するかを制御します。どちらの機能も明示的に有効にしていない場合は、デフォルトで転送が使用されます。

会議シナリオ

次のシナリオでは、呼び出し可能な 2 種類の典型的な会議について説明します。

会議制御に B を使用したコンサルト会議

次の一連の手順では、このシナリオを一般的に説明します。

- A が B にコール（Call 1）する。
- B が応答する。

- B は C にコンサルト（Call 2）する。

  setConferenceEnable()

  ```
  call2.consult(tc, C)
  ```

- C が応答する。
- B が会議を開催する。

  ```
  Call1.conference(Call2)
  ```

（注）　コンサルテーション後、元のコールの conference() メソッドを呼び出し、会議を開催する必要があります。コンサルト コール オブジェクト内で会議を呼び出すと、例外がスローされます。

会議制御に B を使用した任意会議

次の一連の手順では、このシナリオを一般的に説明します。

- A が B にコール（Call 1）する。
- B が応答する。
- B はコールを保留する。
- B は C にコール（Call 2）する。
- C が応答する。
- B が会議を開催する。

  `Call1.conference(Call2)` または

  `Call2.conference(Call1)`

会議イベント

表 3-7 に、Call1.Conference(Call2) 呼び出し時の、一連のコア イベント（コール制御および Cisco の拡張）を示します。

表 3-7　　一連のイベント

原因メタ イベント	コール	イベント	フィールド
META_UNKNOWN	Call1	CiscoConferenceStartEv	consultCall = Call2 finalCall = Call1 conferenceController= TermConnB
META_CALL_MERGING	Call1	CallCtlTermConnTalkingEv B	
META_CALL_MERGING	Call1	ConnCreatedEv C ConnConnectedEv C CallCtlConnEstablishedEv C TermConnCreatedEv C TermConnActiveEv C CallCtlTermConnTalkingEv C	
META_CALL_MERGING	Call2	TermConnDroppedEv B CallCtlTermConnDroppedEv B ConnDisconnectedEv B CallCtlConnDisconnectedEv B	

表 3-7　一連のイベント （続き）

原因メタ イベント	コール	イベント	フィールド
META_CALL_MERGING	Call2	TermConnDroppedEv C CallCtlTermConnDroppedEv C ConnDisconnectedEv C CallCtlConnDisconnectedEv C CallInvalidEv C	consultCall=Call2 finalCall=Call1 conferenceController= TermConnB
META_UNKNOWN	Call2	CallObservationEndedEv	
META_UNKNOWN	Call1	CiscoConferenceEndEv	

転送と会議の拡張

コール転送に関与するすべての通話者には、CiscoTransferStartEv と CiscoTransferEndEv が送信されます。会議通話に関与するすべての通話者には、CiscoConferenceStartEv と CiscoConferenceEndEv が送信されます。コール転送では、最初のコールへの Connection の破棄と 2 番目のコールへの Connection の作成の、2 つのイベントも生成されます。Cisco Unified Communications Manager リリース 3.1 では、これらのイベントの順序が変更されています。最初に最後のコールの Connection が作成され、次にコンサルト コールの Connection が破棄されます。

（注）　Cisco Unified Communications Manager リリース 3.0 では、コールの転送に関与するすべての通話者に、これらのイベントが送信されるとは限りませんでした。

（注）　Cisco Unified Communications Manager リリース 3.0 から 3.1 へアプリケーションを移植する際、転送用の CiscoTransferStartEv と CiscoTransferEndEv の間、または会議用の CiscoConferenceStartEv と CiscoConferenceEndEv の間に発生するイベントの順序にアプリケーションが依存しないようにしてください。

メディアのないコンサルト

メディア パスを確立せずに転送用のコンサルト コールが実行されていることを、アプリケーションから Cisco Unified Communications Manager に通知できます。実際の転送の前にエージェントが使用可能かどうかを判別するために、コンサルタント コールが実行されている場合は、中継コールのためにメディア パスを確立する必要はありません。CiscoConsultCall インターフェイスで定義されている consultWithoutMedia メソッドでは、メディア パスを確立せずにコンサルタント コールが作成されます。

（注）　コンサルト コールは、転送だけが可能です。これは会議には使用できません。

メディア終端の拡張

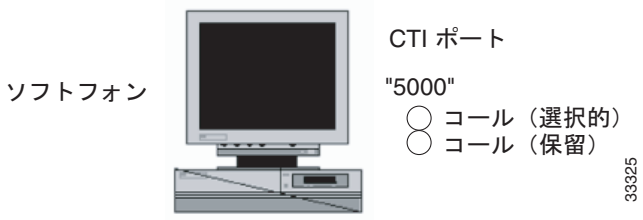
メディア終端機能を利用すると、オーディオやビデオなどのコールのベアラの、アプリケーションによる送信およびキャプチャが可能です。このアクションは、メディアの「再生と録音」または「ソースとシンク」と表現されることもあります。メディア終端は、コールのセットアップやティアダウンの詳細ではなく、コールのエンドポイント間を流れるデータに関与するため、コール制御とは区別されます。

たとえば、Automatic Call Distributor（ACD; 自動呼配送）では、コール制御を使用して使用可能なエージェント間でコールがルーティングされますが、メディアは終端されません。その一方で、Interactive Voice Response（IVR; 対話式音声自動応答）アプリケーションでは、コール制御を使用してコールの応答または接続解除を行い、メディア終端を使用して発信者側に対してサウンド ファイルを再生します。

メディア終端に単独で関与するテレフォニー アプリケーションはありませんが、この機能は常にコール制御と組み合せて使用されます。JTAPI 1.2 では、主にコール制御仕様が記述されており、メディア終端を必要とするアプリケーションには非常に限定的なサポートしか提供されません。Cisco Unified Communications ソリューションのプラットフォームでは、JTAPI 標準よりも高度なメディア終端がサポートされるため、Cisco Unified JTAPI 実装では、JTAPI を拡張してこの機能のフル サポートを追加しています。

Cisco Unified JTAPI では、Computer Telephony Integration（CTI）ポートを使用して、ソフトウェアベースのメディア終端が行われます。これらのポートには、コールの発信または受信に使用可能な 1 本以上の回線（ダイヤル可能な番号）があります。ただし、これらのポートにはメディアのソースとシンクを提供する制御アプリケーションが必要です。アプリケーションは、Cisco Unified Communications Manager を使用して、メディア終端ポートへの関与を登録します。これで、Cisco Unified Communications Manager からアプリケーションにこの仮想デバイスに関連するすべてのイベントが配送されます。Cisco Unified JTAPI では、CTI ポートは CiscoMediaTerminal と表現されます。図 3-6 に CTI ポートの構成を示します。CTI ポートの管理と構成についての詳細は、Cisco Unified Communications Manager Administration の情報を参照してください。

図 3-6　CTI ポートの図



ソフトフォン アプリケーション（たとえば、PC が電話機として動作）を実装する場合は、Cisco Unified JTAPI アプリケーションを使用して CTI ポートを管理します。

Cisco Unified Communications Managerメディア エンドポイント モデル

エンドポイントは、Cisco Unified Communications ソリューションのプラットフォーム上で、IP フォンやゲートウェイなどのメディアを終端するエンティティを表します。1 つのエンドポイントから別のエンドポイントへのコールによって、2 つのエンドポイント間にメディアのフローが生じます。Cisco Unified Communications ソリューションのプラットフォーム内にあるすべてのエンドポイントにおいて、Real-Time Protocol（RTP）を使用して音声データが送信されます。たとえば、Cisco Unified Communications ソリューションの電話機とゲートウェイには、RTP スタックが組み込まれています。また、アプリケーションが Cisco Unified Communications ソリューションのシステム内でエンドポイントとして機能し、メディアを終端することも可能です。すべての Cisco Unified Communications ソリューションのエンドポイントにおいて RTP が使用されるため、アプリケーションにも RTP パケットを送信および受信する機能が必要です。

ペイロードとパラメータのネゴシエーション

ベアラ データおよびペイロードに加えて、各 RTP パケットにはヘッダーが格納されます。このヘッダーは、一連のパケットをメディア ストリームに再構成およびデコードする方法をエンドポイント側で決定する際に役立ちます。ただし、RTP では、あるストリームに対してどのペイロード タイプを使用するかをエンドポイントでネゴシエートする方法は用意されていません。たとえば、オーディオ データは、G.711 標準を使用してエンコードされます。また、RTP では、エンコード済みデータのサンプリング レートや各 RTP パケットで転送されるサンプル数などの固有のペイロード タイプ パラメータとネゴシエートする方法も提供されません。この代わりに、RTP は通常、エンドポイントでこれらのパラメータとネゴシエートする独自の方法を指定できる H.323 などの別のプロトコルと併用されます。このようなネゴシエーションはすべて、エンドポイント間で RTP パケットを送信する前に実行されます。

RTP ストリームに対するペイロードの選択とパラメータのエンコードは、エンドポイントではなく Cisco Unified Communications Manager が担当します。一般的な双方向のオーディオ通話には、次の 5 つの手順があります。

- 初期化
- ペイロード選択
- チャネル割り当ての受信
- 送受信の開始
- 送受信の停止

初期化

始動時には、各エンドポイントから Cisco Unified Communications Manager にそれぞれのメディア機能、つまり、G.711、G.723、G.729a などが通知されます。たとえば、最初に電話機がオンになったとき、または前の接続が切れてから電話機と Cisco Unified Communications Manager が再接続された後に、IP フォンが始動します。エンドポイントでは、1 つのペイロード タイプと別のペイロード タイプのプリファレンスを表現できませんが、パケット サイズなどの各ペイロード タイプの特定のパラメータを指定することは可能です。

エンドポイントで登録された機能リストは、排他的かつ不変です。G.711 と G.723 のサポートが可能とエンドポイントによって示されている場合、G.729a はサポートされないことを意味します。さらに、エンドポイントでサポートされる機能のリストを変更する場合は、エンドポイントを Cisco Unified Communications Manager との接続から解除し、再度初期化する必要があります。

JTAPI アプリケーションでは、Cisco Unified Communications Manager に CiscoMediaTerminal を登録して、この手順を実行します。CiscoMediaTerminal.register() メソッドでは、アプリケーションを使用して、Cisco Unified Communications Manager への登録のためのメディア機能オブジェクトの配列を提供できます。この手順では、Cisco Unified Communications Manager 設定内のデバイス設定に従って、アプリケーションが特定の電話番号に関するすべての発着コールのエンドポイントとして機能することが、Cisco Unified Communications Manager に通知されます。

ペイロード選択

双方向メディア ストリームが 2 つのエンドポイント間に作成される直前、たとえばコールがエンドポイントで応答されるとき、Cisco Unified Communications Manager によってメディア ストリームに対する適切なペイロード タイプ（コーデック）が選択されます。Cisco Unified Communications Manager により、コールに関与する双方のエンドポイントのメディア機能が比較され、使用に適した共通のペイロード タイプとペイロード パラメータが選択されます。

ペイロードを選択する際の基準には、エンドポイントの機能と場所が使用されますが、今後、この選択ロジックに他の基準が追加される可能性があります。エンドポイントは、コールごとのペイロード タイプの選択に動的に関与することはありません。

チャネル割り当ての受信

Cisco Unified Communications Manager では、2 つのエンドポイント間の RTP ストリームで共通のペイロード タイプが検出できる場合、各エンドポイントに対して、論理的な「受信チャネル」（エンドポイントがコールの RTP データを受信する固有の IP アドレスとポート）の作成が要求されます。この要求を受けて、各エンドポイントから Cisco Unified Communications Manager に IP アドレスとポートが返されます。

現在、IP フォンとゲートウェイに限ってこの手順が実行されます。Cisco Unified Communications Manager では、初期化中に固定 IP アドレスとポートを JTAPI アプリケーション側で指定する必要があります。したがって、JTAPI アプリケーションを使用して、同じエンドポイントに対する複数のメディア ストリームを同時に終端できません。アプリケーション側で複数のメディア ストリームを終端する場合、複数のエンドポイントを同時に登録する必要があります。

オープンしている受信チャネル要求に対してエンドポイントからの迅速な応答がない場合、Cisco Unified Communications Manager によってコールの接続が解除されます。JTAPI アプリケーションによる CiscoMediaTerminal の登録時には常に IP アドレスが提供されるため、アプリケーションで制御されるエンドポイントへのコールについては、接続が解除されません。ただし、Cisco Unified Communications Manager で、コールに関与する 2 つのエンドポイント間に共通のペイロード タイプが検出されない場合、Cisco Unified Communications Manager によってコールの接続が解除されます。

送受信の開始

Cisco Unified Communications Manager では、双方のチャネル情報を受信した後に、RTP ストリーム用に選択されたコーデック パラメータと、もう一方のエンドポイントの送信先アドレスが、各エンドポイントに通知されます。この情報は、各エンドポイントに対して、送信開始メッセージと受信開始メッセージの 2 つのメッセージで送信されます。

JTAPI アプリケーションでは、RTP データの送信と受信に必要なコーデック パラメータのすべてを含む CiscoRTPOutputStartedEv イベントと CiscoRTPInputStartedEv イベントを受信します。

JTAPI における QoS ベースライン化作業の一環として、CiscoRTPOutputStartedEv は getPrecedenceValue() API をアプリケーションに提供します。CTI により、値「音声コールの DSCP 値」が JTAPI に提供されます。アプリケーションではこの値を使用して、アプリケーションで開くメディア ストリームに対して DSCP 値を設定できます。

送受信の停止

保留や接続解除などの機能のために RTP ストリームを中断する必要がある場合、RTP データの送信と受信を停止する要求が Cisco Unified Communications Manager から各エンドポイントに対して送信されます。メディア フローの開始と同時に、送信停止と受信停止のメッセージが個別に送信されます。

JTAPI アプリケーションでは、CiscoRTPOutputStoppedEv と CiscoRTPInputStoppedEv を受信します。

Cisco MediaTerminal

JTAPI では、端末オブジェクトは、コールに対する論理エンドポイントを表し、データ（デジタルでエンコードされた音声サンプルなど）の送受信が可能であると想定されます。このため、Cisco Unified IP Phone は、JTAPI の端末として表現されます。ただし、ゲートウェイは、メディア終端を実行しますが、端末としては表現されません。CiscoMediaTerminal が、アプリケーションがメディア終端を担当する特殊なエンドポイントを表します。

CiscoMediaTerminal の使用に関連する手順は次の 4 つです。

- プロビジョニング
- 登録
- オブザーバの追加
- コールの受け入れ

プロビジョニング

CiscoMediaTerminal は実際のハードウェアの IP フォンやゲートウェイを表すことはありませんが、物理端末と同様に Cisco Unified Communications Manager でプロビジョニングされることに注意してください。IP フォンがデバイス ウィザードを使用して Cisco Unified Communications Manager データベースに追加されるのと同様に、同じ方法で CiscoMediaTerminal が追加されます。これにより Cisco Unified Communications Manager はアプリケーションのエンドポイントを電話番号およびコール転送などの他のコール制御プロパティに関連付けることができます。DeviceWizard では、CiscoMediaTerminal というデバイス タイプは存在しません。その代わりに、Cisco Unified Communications Manager には、アプリケーションの登録をサポートする複数のデバイス タイプがあります。JTAPI では、これらの各タイプが CiscoMediaTerminal として公開されます。現在、JTAPI の CiscoMediaTerminal を表すデバイス タイプは CTI ポートだけです。

次に、アプリケーションで制御されるエンドポイントとして使用するために CTI ポートをプロビジョニングする手順を示します。

手順

ステップ 1　Cisco Unified Communications Manager の設定ウィンドウ内で、デバイス ウィザードを使って [デバイス] > [電話] ウィンドウから、CTI ポートのデバイスを追加します。この CTI ポートのデバイス名には、対応する JTAPI の CiscoMediaTerminal の名前を指定します。

ステップ 2　[ユーザ] > [Global Directory] ウィンドウを使用して、新しい CTI ポート デバイスを、アプリケーションが [ユーザ] ウィンドウを使って制御するデバイスのリストに追加します。

詳細については、『*Cisco Unified Communications Manager Administration Guide*』を参照してください。

登録

メディア終端デバイスが Cisco Unified Communications Manager 内で正しくプロビジョニングされると、アプリケーションでは、Provider.getTerminal() メソッドまたは CiscoProvider.getMediaTerminal() メソッドを使用して、対応する CiscoMediaTerminal オブジェクトへの参照を取得できます。2 つのメソッドの違いは、CiscoProvider.getMediaTerminal() では CiscoMediaTerminal だけが返されるのに対し、Provider.getTerminal() では、物理的な IP フォンを表現するオブジェクトなど、プロバイダーに関連するすべての端末オブジェクトが返される点です。

Cisco Unified Communications Manager に対して特定のペイロード タイプの RTP ストリームを終端する意図を通知するには、CiscoMediaTerminal.register() を使用します。CiscoMediaTermina.register() メソッドには、IP アドレス、ポート番号、およびアプリケーションでサポートするコーデックのタイプとコーデック固有のパラメータを示す CiscoMediaCapability オブジェクトの配列を指定します。

IP アドレスとポートは、アプリケーションがメディア ストリームを受信できるアドレスを示します。次のコード例に、CiscoMediaTerminal を登録し、ポート番号 1234 のローカル アドレスにバインドする方法を示します。

```
CiscoMediaTerminal registerTerminal (Provider provider, String terminalName) {
  final int PORT_NUMBER = 1234;
  try {
   CiscoMediaTerminal terminal = provider.getTerminal (terminalName);
   CiscoMediaCapability [] caps = new CiscoMediaCapability [1];
   caps[0] = CiscoMediaCapability.G711_64K_30_MILLISECONDS;
   terminal.register (InetAddress.getLocalHost (), PORT_NUMBER, caps);
  }
  catch (Exception e) {
    return null;
  }
}
```

このコード例を機能させるには、指定したプロバイダーが IN_SERVICE である必要があります。さらに、このコードでは、定数 CiscoMediaCapability.G711_64K_30_MILLISECONDS が使用されることに注意してください。これは実際には、最大 30 ミリ秒の RTP パケット サイズを指定する、CiscoG711MediaCapability オブジェクトへの静的な参照を表します。これと他の共通のメディア形式は、CiscoMediaCapability クラスで定義されています。

CiscoMediaCapability クラスに示されていないメディア ペイロードを指定するには、2 つのオプションがあります。対象となるペイロード タイプが CiscoMediaCapability の既存サブクラスからの単純な派生の 1 つである場合、必要な処理はサブクラスの新規インスタンスの構築だけです。たとえば、最大 60 ミリ秒の RTP パケット サイズの G.711 ペイロードをアプリケーションでサポートできる場合、コンストラクタに 60 ミリ秒を指定するなどして CiscoG711MediaCapability オブジェクトを直接構築できます。

その一方で、対象となるペイロード タイプに一致する CiscoMediaCapability の既存のサブクラスがない場合には、CiscoMediaCapability クラスのインスタンスを直接構築します。CiscoMediaCapability の構築時に指定できる他のパラメータは、30 ミリ秒などの最大パケット サイズだけです。

次のコードに、カスタムのペイロード機能の登録方法を示します。

```
CiscoMediaTerminal registerTerminal (Provider provider, String terminalName) {
  final int PORT_NUMBER = 1234;
  try {
   CiscoMediaTerminal terminal = provider.getTerminal (terminalName);
   CiscoMediaCapability [] caps = new CiscoMediaCapability [1];
   caps[0] = new CiscoMediaCapability (
     RTPPayload.G728,
     30  // maximum packet size, in milliseconds
     );
   terminal.register (InetAddress.getLocalHost (), PORT_NUMBER, caps);
  }
  catch ( Exception e) {
    return null;
  }
}
```

CiscoMediaCapability オブジェクトの構築に使用するペイロード タイプのパラメータは、RTP ヘッダーのペイロード フィールドに対応します。このために、RTPPayload インターフェイスでは、既知のペイロード タイプが定義されています。

オブザーバの追加

RTP データを送受する場所と時間を示すイベントを受信するには、CiscoMediaTerminal に CiscoTerminalObserver を配置します。CiscoTerminalObserver では、新しいメソッドを定義することなく標準の JTAPI TerminalObserver インターフェイスが拡張されます。これは、RTP イベントの受信時にアプリケーションの関与を通知するマーカー インターフェイスを提供するものです。

（注）　これは、CallObserver ではなく TerminalObserver であるため、Terminal.addCallObserver() メソッドではなく Terminal.addObserver() メソッドを使用して追加する必要があります。

さらに、CiscoMediaTerminal に関連付けられたアドレス オブジェクトに CallControlCallObserver を追加します。これにより、CiscoMediaTerminal への呼び出しコールがあるときに、アプリケーションに通知されることが保証されます。呼び出しコールを自動的に受け入れる通常の IP フォンとは異なり、CiscoMediaTerminal は、提供されるコールの受け入れ、接続解除（拒否）、またはリダイレクトを実行します。CallCtlConnOfferedEv は、Terminal オブジェクトではなく Address オブジェクト上に配置された CallControlCallObserver に対してだけ提示されるため、アプリケーションでは、その CallControlCallObserver を正しい場所に配置します。

（注）　CallObserver インターフェイスだけでなく、CallControlCallObserver インターフェイスも確実に実装してください。CallCtlConnOfferedEv はコアの CallObserver インターフェイスだけを実装するオブザーバには配送されません。

コールの受け入れ

着信コールが CiscoMediaTerminal アドレスに到着した場合、TerminalConnection の作成前に CallControlConnection.accept() メソッドを使用してコールを受け入れる必要があります。このプロセスは、発信コールには適用されません。この Connection は、コールが電話番号確認の次へ進行するとすぐに、CallControlConnection.ESTABLISHED 状態になるからです。Connection が受け入れられると、発側の TerminalConnection に応答してメディア フローを開始します。発信側エンドポイントの機能に登録された機能に Cisco Unified Communications Manager を合わせることが可能である場合、アプリケーションによって RTP データの送受信を開始できるように、Cisco Unified Communications Manager からメディア フロー イベントが送信されます。

メディア フロー イベントの受信と応答

2 つのエンドポイント間にメディア ストリームが作成される場合は必ず、Cisco Unified Communications Manager から双方のエンドポイントに送信開始と受信開始のイベントが発行されます。JTAPI では、CiscoRTPOutputStartedEv イベントと CiscoRTPInputStartedEv イベントで送信開始と受信開始のイベントを表します。CiscoRTPOutputStartedEv.getRTPOutputProperties() メソッドでは CiscoRTPOutputProperties オブジェクトが返されます。アプリケーションでは、このオブジェクトから、コールのピア エンドポイントの送信先アドレス、およびペイロード タイプやパケット サイズなどのストリームに関する RTP プロパティを確認できます。同様に、CiscoRTPInputStartedEv.getRTPInputProperties() メソッドでは、アプリケーションに着信ストリームの RTP 特性を通知する CiscoRTPInputProperties オブジェクトが返されます。

メディアが流れている間は常に、CiscoMediaTerminal.getRTPOutputProperties() メソッドと CiscoMediaTerminal.getRTPInputProperties() メソッドから現在の CiscoRTPOutputProperties と CiscoRTPInputProperties も使用できます。CiscoMediaTerminal でメディアの送受信が想定されていない場合、これらのメソッドによって例外がスローされます。

Cisco Unified Communications Manager によって、たとえばコールの接続解除または保留の結果として、アプリケーションによるメディアの送受信が停止される場合は、CiscoRTPOutputStoppedEv イベントと CiscoRTPInputStoppedEv イベントが送信されます。これらのイベントは、2 つのエンドポイント間に存在する現在の RTP メディア ストリームをティアダウンする必要があることを意味します。

着信コール メディア フロー イベントの図

表 3-8 に、アプリケーションで制御されるエンドポイントにコールが提示される場合の Cisco Unified Communications Manager と JTAPI アプリケーション間のダイアログを示します。左の列のイベントはアプリケーションの CallObserver に送信される JTAPI イベント、右の列の要求はアプリケーションで呼び出されるメソッドを表します。

表 3-8　着信メディア フロー イベント

JTAPI イベント	方向	アプリケーションの要求
CallActiveEv ConnCreatedEv ConnProceedingEv CallCtlConnOfferingEv	→	
	←	CallControlConnection.accept ()
CallCtlConnAlertingEv TermConnCreatedEv TermConnRingingEv	→	
	←	TerminalConnection.answer ()
ConnConnectedEv CallCtlConnEstablishedEv TermConnTalkingEv CiscoRTPOutputStartedEv CiscoRTPInputStartedEv	→	
	←	CallControlConnection.disconnect ()
CiscoRTPOutputStoppedEv CiscoRTPInputStoppedEv TermConnDroppedEv CallCtlConnDisconnectedEv	→	

（注）　表 3-8 では、ローカル接続、つまり、アプリケーション エンドポイント向けの JTAPI イベントを示しています。実際の JTAPI メタ イベント ストリームには、発側の状態を記述したイベントが含まれます。

Cisco Unified Communications ソリューションの RTP 実装

Cisco Unified Communications ソリューションのアーキテクチャによって性能が改善されたため、Cisco Unified Communications ソリューションの電話とゲートウェイでは、RTP の一部の機能と、通常、RTP の関連付けられる Real-Time Control Protocol（RTCP）を実装していません。アプリケーションの互換性を確保するには、次の点を考慮する必要があります。

- RTCP はサポートされません。Cisco Unified Communications ソリューションのエンドポイントから RTCP メッセージが送信されず、エンドポイントに送信されるこの種のメッセージはすべて無視されます。
- Cisco Unified Communications ソリューションのエンドポイントでは現在、RTP ヘッダーの「synchronization source」（SSRC）フィールドは使用されていません。アプリケーションによって SSRC フィールドを使用した RTP ストリームを多重化しないでください。多重化すると、電話機とゲートウェイにおいて、メディアを正しくデコードおよび提示できません。

リダイレクト

JTAPI 1.2 では、CallControlConnection.redirect() メソッドの事前条件の 1 つとして、Connection が CallControlConnection.OFFERING または CallControlConnection.ALERTING のどちらかの状態にあることが規定されています。Cisco Unified JTAPI では、CallControlConnection.ESTABLISHED 状態の Connection もリダイレクトできます。

redirect() メソッドには、CiscoConnection インターフェイス内に次のオーバーロードされた形式があります。これにより、コールのリダイレクト中に障害が起きたときに、必要な動作をアプリケーションによって指定し、発信元検索スペースを指定したり、元の着信フィールドをリセットできます。

オーバーロードのリダイレクト メソッドにある次の INT パラメータの 1 つを CiscoConnection インターフェイスから渡すことで、アプリケーションに必要な動作を選択します。

- 障害時のリダイレクト ドロップ：コールが使用中または無効な送信先にリダイレクトされた場合、Cisco Unified Communications Manager では、リダイレクトが失敗した場合にコールを破棄するか、またはリダイレクト コントローラにコールを残す処理のどちらかを実行できます。この後、JTAPI アプリケーションを使用して、別の送信先にコールをリダイレクトするなどの修正処置ができます。リダイレクト モード パラメータには、次のオプションがあります。
  - CiscoConnection.REDIRECT_DROP_ON_FAILURE
  - CiscoConnection.REDIRECT_NORMAL
- コール サーチ スペース：リダイレクトでは、コール サーチ スペースのパラメータを使用して、使用する callingSearchSpace を指定します。アプリケーションでは、発側のコール サーチ スペースまたはリダイレクト コントローラのコール サーチ スペースのどちらかを使用できます。このシナリオには、次のパラメータ オプションがあります。
  - CiscoConnection.CALLINGADDRESS_SEARCH_SPACE
  - CiscoConnection.ADDRESS_SEARCH_SPACE
- 元の着信のリセット：着信アドレス オプションのパラメータを使用して、元の着信フィールドをリセットします。このシナリオには、次のオプションがあります。
  - CiscoConnection.CALLED_ADDRESS_UNCHANGED
  - CiscoConnection.CALLED_ADDRESS_SET_TO_REDIRECT_DESTINATION. このオプションでは、コールがリダイレクトの送信先に到着したときに、フィールドが影響を受けます。

詳細については、com.cisco.jtapi.extensions.CiscoConnection のドキュメントを参照してください。

A が B をコールし、B が C にリダイレクトし、C（リダイレクト先）がプロバイダーによって監視されたアドレスを表さない場合、JTAPI により、C について原因コード Ev.CAUSE_NORMAL で CallCtlConnAlertingEv が提供されます。リリース 5.0 よりも前では、このシナリオの原因コードは CiscoCallEv.CAUSE_REDIRECTED を指定していました。

この変更では、C がプロバイダーによって監視されている場合と監視されていない場合の動作の一貫性が維持されています。

（注）　同じシナリオで C が監視されている場合、5.0 よりも前のリリースでも C について CAUSE_NORMAL で CallCtlConnAlertingEv が提供されており、この動作は変更ありません。

ルーティング

JTAPI でのルーティングには、Cisco Unified Communications Manager 上の CTI ルート ポイントの設定が必要です。複数のコールをこのルート ポイントにキューイングできますが、CTI ルート ポイントのデバイス上に設定できるのは、1 本の回線だけです。

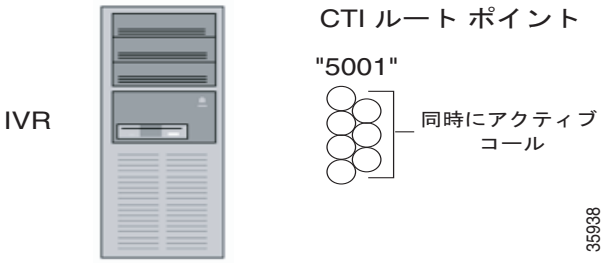
コール センター パッケージで説明されるように、補助ルーティングの JTAPI 実装には、次のアクションが含まれています。

- ルート アドレス上のルート コールバックを登録する
- 多様なルーティング イベント（routeSelect、routeEnd）への応答に対して適切なハンドラを作成する

（注）　CTI ルート ポイントは、同一回線上で同時に無制限の数の着信コールを処理できるデバイスを表します。コールに対しては、javax.telephony.callcenter パッケージのメソッドを使用したルーティング、あるいは javax.telephony.callcontrol パッケージのメソッドを使用した受け入れ、リダイレクト、または接続解除を実行できます。各 CTI ルート ポイントは、Cisco Unified Communications Manager の 1 回線だけで構成できます。1 つの CTI ルート ポイントでサポートされる回線の最大数は 34 です。34 を超える回線をサポートするには、ルート ポイントを追加します。CTI ルート ポイントの構成方法と管理方法の詳細については、『*Cisco Unified Communications Manager Administration Guide*』を参照してください。

図 3-7 に CTI ルート ポイントの構成を示します。

図 3-7　CTI ルート ポイント



Cisco Route Session の実装

コールが RouteAddress で着信されると、この実装では Route Session スレッドが開始され、アプリケーションに RouteEvent が送信されます。次に、このスレッドでは、routeSelect() または endRoute() のどちらかによる RouteEvent に対するアプリケーションの応答時間を計るため、タイマー スレッドが開始されます。アプリケーションが routeSelect (String[] selectedRoutes) で応答した場合、JTAPI ではすべての事前条件が満たされていることを確認してから、配列で指定された最初の送信先へのコールのルーティングが試みられます。送信先が有効かつ利用可能な番号であった場合、コールはルーティングされ、アプリケーションでは RouteUsedEvent と RouteEndEvent が順番に取得されます。送信先が有効かつ利用可能な番号でない場合は、無効、使用中、利用できない送信先などの原因でルーティングにエラーが生じ、アプリケーションでは ReRouteEvent が取得されます。JTAPI では、re-Route Event を送信する前にタイマー スレッドが再開されます。Cisco Unified Communications Manager では、再ルーティングがサポートされないため、ルーティングが失敗した場合、発信者に対してビジー トーンが再生されるか、またはコールが破棄されます。アプリケーションでは、すべての障害インスタンスをクリーン アップしたり、JTAPI に endRoute を送信して RouteSession をクリーン アップできます。アプリケーションが endRoute() に応答しない場合、JTAPI タイマーは再度満了し、JTAPI では、アプリケーションに RouteEndEvent() を送信することで、Route Session がクリーン アップされます。

アプリケーションから selectRoute() メソッドまたは endRoute() メソッドが返される前にルーティング タイマーが満了すると、Cisco Unified Communications Manager によって、登録されていない電話機に対して発呼した場合と同じ処理（ファースト ビジー音の再生）が適用されます。ルート ポイント上に ForwardNoAnswer が設定されている場合、タイマーが満了した時点で、すぐにコールはその番号に転送されます。

コールをルーティングする有効なアドレスを応答できない場合、endRoute とエラーをアプリケーションによって呼び出すこともできます。JTAPI 仕様では、RouteSession インターフェイスで ERROR_RESOURCE_BUSY、ERROR_RESOURCE_OUT_OF_SERVICE、および ERROR_UNKNOWN の 3 種類のエラーが定義されています。endRoute が RouteSession 上で呼び出される場合、この実装では現在、accepts() により RouteAddress でコールが受け入れられ、発信者に対してリングバック音が再生されます。ルート ポイントに対して転送が設定されている場合、Forwarding Timer が満了するとコールが転送されます。

Route Timer の選択

このタイマーは、RouteSelectTimeout=5000 というキーを含む JTAPI.ini コンフィギュレーション ファイルで設定します。単位にはミリ秒を使用します。このタイマーのデフォルト値は、5 秒に指定されていますが、アプリケーションのニーズに応じてタイマーの時間を増減して、Route Session のクリーンアップ効率を改善できます。このタイマーの値は、むやみに大きくしないでください。スレッドとしての各 Route Session は、ルート ポイントへのコールを表すため、これらの Route Session をクリーンアップする必要があります。アプリケーション上でルート イベントの受信と routeSelect/endRoute イベントへの応答の間に深刻な遅延が予測される場合は、アプリケーションによってこのタイマーの時間を適切に延長してください。

Forwarding Timer

現在、システム全体で使用されている Forward on No Answer 用タイマー（つまり、Cisco Unified Communications Manager 上のすべてのデバイスに適用されます）は、Cisco Unified Communications Manager のサービス パラメータ設定により設定できます。このタイマーのデフォルト値は、12 秒に指定されています。今後のリリースには、JTAPI でコールが受け入れられた直後（アプリケーションから endRoute を呼び出したか、またはルーティング タイマーが満了したとき）にルート ポイントに対する転送が機能するように、CTI Route Point 用の別個のタイマーが追加される予定です。

RouteSession の拡張

CiscoRouteSession は、JTAPI 仕様に対する Cisco の拡張の役割を果たします。この拡張の最も重要な点は、基盤となるコール オブジェクトがアプリケーションに公開されることです。CiscoRouteSession.getCall() は CiscoCall を返し、このコールによって、関連するアドレス、Connection などの他のコール モデル オブジェクトが公開されます。この拡張では、アプリケーションに対する追加のエラーも定義されています。

発信者オプションの要約

コールバックがない場合、または routeEvent が RouteSession.routeSelect() または endRoute() で応答されない場合、次の状態になるまで発信者に対して何も送信されません。

- アプリケーション側でルート ポイントの Connection を disconnect() によって解除または reject() によって拒否でき、その結果、発信者に対してビジー トーンが送信される。
- アプリケーション側でコールを受け入れることができ、Forward No Answer が設定されていればこれが作動する。
- アプリケーションによってコールを破棄できる。発信者は受話器を持ったままで、何が起きたかが理解できません。

コールバックを使用すると、アプリケーションによって endRoute() を呼び出した場合、endRoute() が返された後、次の状態になるまで発信者に対してリングバック音が送信されます。

- クライアントがコールを破棄する disconnect() を呼び出す。
- クライアントがコールを redirects() でリダイレクトする。
- scm.ini で設定された Forward on No Answer タイマーが作動し、前の 2 つのオプションがまだ作動していない場合はコールが転送される。
- ルート ポイントに対して転送が設定されていない場合、最初の 2 つのオプションが作動していない場合は発信者に対してリングバック音が送信され続ける。

ルート ポイント使用時の耐障害性

ルート ポイントを使用するアプリケーションで耐障害性を確保するには、2 つの JTAPI アプリケーションを用意し、別個の RouteAddress の登録された 2 つの異なる Cisco Unified Communications Manager にこのアプリケーションを接続する方法があります。たとえば、Application1 では Communications Manager1 を使用して RouteAddress1 を管理します。Application2 では、Communications Manager2 を使用して RouteAddress2 を管理します。Cisco Unified Communications Manager Administration 上では、ルート ポイントが互いにポイントし合うように、これらの CTI ルート ポイントに対する ForwardNoAnswer の設定を管理する必要があります。この例では、RouteAddress1 に FNA=RouteAddess2、RouteAddress2 に FNA=RouteAddress1 を付けます。Communications Manager1 が停止した場合、Application2 が引き継げるようにコールが RouteAddress2 に転送されます。さらに、ProviderShutdown イベントの受信時にそれぞれの適切な Cisco Unified Communications Manager サーバに再接続するように、両方のアプリケーションを設定できます。

冗長性

設定時には、デバイスをデバイス プール内に設定し、静的な Cisco Unified Communications Manager グループに割り当てる必要があります。デバイスは、コール制御のシグナリングを処理する特定の Cisco Unified Communications Manager サーバに登録されます。サーバに障害が起きると、デバイス

はそのグループのバックアップ サーバにフェールオーバーされます。デバイスは、プライマリ サーバがオンライン状態に復旧し、デバイス上のアクティブ コールがなくなるまで待機した後で、プライマリ Cisco Unified Communications Manager サーバにリホームされます。Cisco Unified JTAPI では、バックアップ サーバへの登録中に、一時的なアウト オブ サービスのメッセージを送信することで、アプリケーションにこの移行が通知されます。

クラスタ抽象概念

CTIManager は、クラスタ内にあるすべての Cisco Unified Communications Manager の仮想表現を提供します。Cisco Unified JTAPI アプリケーションは、特定の Cisco Unified Communications Manager ではなく CTIManager と通信します。また、CTIManager によってクラスタ内にある Cisco Unified Communications Manager 間の接続が維持されます。これにより、プロバイダーは CTIManager を使用してクラスタ内のすべてのデバイスを表現できます。図 3-8 は「JTAPI、Cisco Unified Communications Manager、および CTIManager が 1 つのボックス内にある単一ボックス構成」を図示したものです。図 3-9 は「JTAPI を独立したクライアントとして配置した冗長 Cisco Unified Communications Manager と CTIManager」を図示したものです。

クラスタ管理とデバイス プール設定に関する詳細については、Cisco Unified Communications Manager ヘルプの情報を参照してください。

図 3-8　JTAPI、Cisco Unified Communications Manager、および CTIManager が 1 つのボックス内にある単一ボックス構成

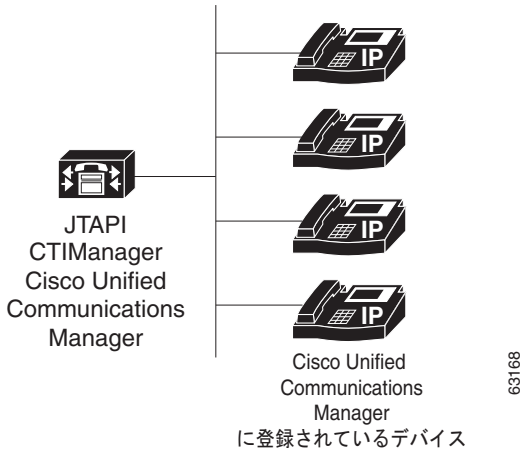
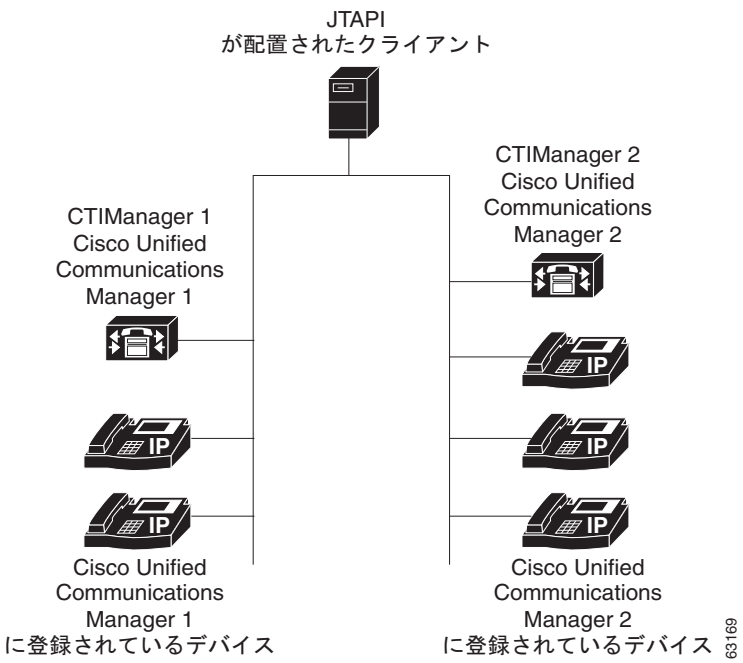


図 3-9　JTAPI を独立したクライアントとして配置した冗長 Cisco Unified Communications Manager と CTIManager



（注）　Cisco Unified Communications Manager の以前のリリースでは、Cisco Unified JTAPI 上で動作するアプリケーションでは、単一の Cisco Unified Communications Manager に登録されたデバイスの制御または監視だけが可能でした。Cisco Unified Communications Manager サーバが停止した場合、Cisco Unified Communications Manager サーバと JTAPI 間の接続が停止し、プロバイダーはシャットダウンします。

Cisco Unified Communications Managerサーバの障害

Cisco Unified Communications Manager に障害が起きた場合、関連デバイスはそのグループにある次の Cisco Unified Communications Manager サーバにリホームされます。このプロセスは、デバイスごとのデバイス プール情報の設定にある Cisco Unified Communications Manager の優先順位付きリストで定義されます。

Cisco Unified Communications Manager サーバの障害時でも、クラスタ内にあるデバイスの部分的な機能停止が生じるだけです。これらのデバイスは、Cisco Unified Communications Manager のフェールオーバーが成功し、セカンダリ Cisco Unified Communications Manager に登録された後、引き続き使用可能になります。

（注）　Cisco Unified IP Phone 7960 などのデバイスでは、デバイス上にアクティブ コールがないときにだけ、セカンダリの Cisco Unified Communications Manager サーバに対するフェールオーバーが行われます。コール中の Cisco Unified Communications Manager サーバの障害では、そのデバイスの観察が停止するだけです。メディア パスはそのまま存在しますが、コール制御は機能しません。

Cisco Unified JTAPI では、CiscoAddrOutOfServiceEv イベントと CiscoTermOutOfServiceEv イベントを使用して、この部分的な機能停止がアプリケーションに伝えられます。Cisco Unified Communications Manager のフェールオーバー時は、JTAPI アプリケーションに対してデバイスが使用

可能になる前に、デバイスは正常にセカンダリ Cisco Unified Communications Manager に登録される必要があります。Cisco Unified JTAPI からは、CiscoAddrInServiceEv イベントと CiscoTermInServiceEv イベントが送信されます。

この間、プロバイダーはイン サービス状態にあります。他の Cisco Unified Communications Manager サーバ上にあるデバイスには、引き続きすべてのコール制御を実行できます。イベントは、個々のアドレスまたは端末のオブザーバ オブジェクトがコールバックされると送信されます。CiscoAddrOutOfServiceEv イベントと CiscoAddrInServiceEv イベントは、AddressObserver を実装しているオブジェクトに送信され、addressChangedEvent() コールバック オブジェクト メソッドを使用してアドレスに追加されます。CiscoTermOutOfServiceEv イベントと CiscoTermInServiceEv イベントは、TerminalObserver インターフェイスを実装しているオブジェクトに送信され、terminalChangedEvent( ) コールバック メソッドを使用して端末に追加されます。

デバイスに現在コールがある場合は、CallObservationEnded メッセージが CallObserver callChangedEvent() のコールバック時に送信され、次に、CiscoAddrOutOfServiceEv メッセージと CiscoTermOutOfServiceEv メッセージが送信されます。

（注） アプリケーションでは、アドレスまたは端末のコール制御関数を呼び出す前に、CiscoAddrOutOfServiceEv、CiscoTermOutOfServiceEv、CiscoAddrInServiceEv、および CiscoTermInServiceEv のイベントの監視および応答を実行する必要があります。このアクションをサポートしない場合、アプリケーション側でシステムの正確な状態を把握できないため、予想外のエラーが生じる可能性があります。

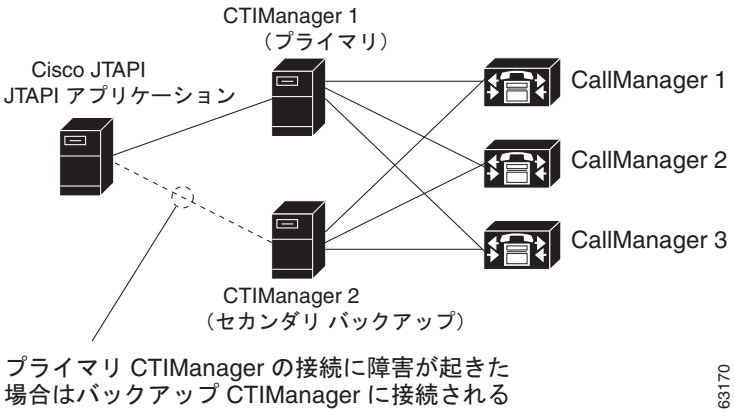
CTI Manager の冗長性

Cisco Unified JTAPI では、CTI Manager 経由での冗長性のための透過型アプリケーションも提供されます。プライマリ CTI Manager に障害が起きると、Cisco Unified JTAPI はバックアップ CTI Manager と自動的に接続され、この再接続がアプリケーションに伝えられます。これにより、1 台の Cisco Unified Communications Manager サーバに接続する代わりに、アプリケーションは CTIManager のセットに接続します。アプリケーションでは、JTAPI を呼び出すときに CTIManager サーバの名前を指定します。

Cisco Unified JTAPI と CTIManager では、両者間の接続切れを検出するために双方向のハートビート信号が維持されます。CTIManager によって、アプリケーションが作動していない状態が検出され、その割り当てられたリソースがクリーン アップされます。図 3-10 はクラスタ内における JTAPI、CTIManager、および Cisco Unified Communications Manager の論理表現を図示したものです。

（注） Cisco Unified JTAPI では、プライマリ CTIManager に正常に接続された後、CTIManager への JTAPI 接続が失敗すると、プライマリまたはバックアップの CTIManager への再接続が交互に試みられます。

図 3-10　クラスタ内における JTAPI、CTIManager、および Cisco Unified Communications Manager の論理表現



CTIManager の冗長性を呼び出す

アプリケーションの起動中に CiscoJtapiPeer の getProvider() メソッドが呼び出されると、Cisco Unified JTAPI では、リストにある最初の CTIManager への接続が試みられ、最初の接続の試みが失敗すると、次の CTIManager への接続が試行されます。リストにあるすべての CTIManager が使用できないか、または接続がすべての CTIManager に拒否された場合、例外がアプリケーションに送信され、それ以上の再接続は試行されません。最初の接続が成功した後、CTIManager または CTIManager への接続に障害が検出されると、Cisco Unified JTAPI では、バックアップまたはプライマリの CTIManager への接続が交互に試みられます。

冗長 CTIManager のリストは、カンマ区切りのリストであり、文字列として CiscoJtapiPeer.getProvider(String providerString) に渡されます。providerString の使用方法は次のとおりです。

- providerString = CTIManager;login=XXX;passwd=YYY;appinfo=ZZZ （非冗長機能）
- providerString = CTIManager1,CTIManager2;login=XXX,passwd=YYY;appinfo=ZZZ （冗長機能）

（注）　appinfo パラメータはオプションなので、アプリケーションでは特定の appinfo パラメータは指定しません。このパラメータは、Cisco Unified JTAPI が JTAPI インスタンス ID とローカル ホスト名から生成します。

また、CiscoJtapiPeer.getServices() メソッドをサポートするために、jtapi.ini ファイル内にさまざまな CTIManager リストを定義できます。Cisco Unified JTAPI では、次の定義を適用できます。

CtiManagers=<CTIManager1>,<CTIManager2>;<CTIManager3>

説明：

<CTIManager1>、<CTIManager2> は、冗長グループを示します。

<CTIManager3> は、非冗長グループを示します。

CTIManager の障害

Cisco Unified JTAPI によって CTIManager との接続の切断が検出されると、アプリケーションにこのサービスの切断が通知されます。該当するオブザーバ上のアプリケーションに次のイベントが送信されます。

- CallObservationEndedEv イベントは、アドレス上にあるすべてのコール オブザーバに送信され、進行中のコールは終了します。これらのコールは物理的には接続されていますが、Cisco Unified JTAPI にはコールの状態変化を送信する機能がないため、アプリケーションによるコールの観察は終了します。
- CiscoAddrOutOfServiceEv イベントが端末上のすべてのアドレスに送信され、CiscoTermOutOfServiceEv イベントが端末に送信されます。
- このプロセスは、プロバイダーのユーザ制御リストにあるすべての端末に対して繰り返されます。（CiscoAddrOutOfServiceEv イベントは、アクティブな AddressObserver があるアドレスにだけ送信され、CiscoTermOutOfServiceEv イベントは、アクティブな TerminalObserver がある端末にだけ送信されます。）
- その後、プロバイダーはアウト オブ サービス状態に設定され、プロバイダー上の ProviderObserver コールバックに ProvOutOfServiceEv イベントが配送されます。

次に、Cisco Unified JTAPI によってリストの次の CTIManager への接続が試みられ、ProvInServiceEv が ProviderObserver に送信されます。あらかじめアプリケーションの制御下で登録されたデバイスは新しい CTIManager に再インストールされます。デバイスが再インストールされると、CiscoAddrInServiceEv イベントと CiscoTermInServiceEv イベントが、それぞれのオブザーバを経由してアプリケーションに送信されます。あらかじめ追加されたすべてのオブザーバは維持されます。デバイス上にコールが存在する場合、コールのスナップショットが、それぞれのコール オブザーバに送信されます。

あらかじめ登録された CTI ポートは同じメディア パラメータで再登録されます。RouteAddress コールバックは以前と同様に維持され、これらのコールは新しい CTIManager 上で復旧されます。ただし、コール スナップショットは RouteAddress に配送されません。

ハートビート

Cisco Unified JTAPI と CTIManager では、CTIManager または JTAPI の障害を検出するためにハートビート信号が維持されます。CTIManager サーバは、双方向のハートビートでハートビート パラメータを制御します。アプリケーションでは、Cisco Unified JTAPI を初期化するときに、希望するサーバ ハートビート インターバルを要求できますが、このインターバルは CTIManager によって上書きされる可能性があります。

アプリケーションでは、jtapi.ini 設定にある DesiredServerHeartbeatInterval を使用して必要なハートビート パラメータを指定します。

Cisco Unified JTAPI では、初期化中にクライアントの要求したハートビート インターバルが指定されます。CTIManager では、クライアント側のハートビート インターバルとサーバ（CTIManager）からハートビートを送信するインターバルが、Cisco Unified JTAPI に対して指定されます。サーバ指定のインターバルの 2 倍の期間にハートビート メッセージが受信されない場合は、クライアントにより接続がティアダウンされます。ハートビートのトラフィックを最小化するには、クライアントからサーバへのすべてのメッセージ、またはサーバからクライアントへのすべてのイベントを、ハートビートの代わりに使用します。

コールの発側の表示名

CiscoCall インターフェイスには、コールの発側と着側の名前を表示できるメソッドが用意されています。アプリケーション上で getCurrentCallingPartyDisplayName() を使用して、発呼側の表示名を取得できます。

JTAPI アプリケーションでは、次のインターフェイスを使用して発側と着側の表示名を取得できます。

{
. .

```
..
    /**
     *This interface returns the display name of the called party in the call.
     *It returns null if display name is unknown.
    */
    public String getCurrentCalledPartyDisplayName();

    /**
     *This interface returns the display name of the calling party.
     *It returns null if display name is unknown.
    */
    public String getCurrentCallingPartyDisplayName();
}
```

表示名は、アドレス オブジェクト内部で保管され、currentCallingAddress と currentCalledAddress が更新されたときに更新されます。コールがアクティブな状態にない場合、またはコールの currentCalling と currentCalled のアドレスが初期化されない場合、NULL が返されます。

（注）　Call.getCurrentCalledAddress() と call.getCurrentCallingAddress() は、電話会議ではサポートされません。また、電話会議では call.getCurrentCalledPartyDisplayName() と call.getCurrentCallingPartyDisplayName() もサポートされません。

SetMessageWaiting

SetMessageWaiting インターフェイスは、アドレスのメッセージ待ちランプまたはインジケータを、アプリケーションで設定するためのメソッドを提供します。このメソッドは、送信先と同じパーティションにあるアドレス上で呼び出します。

次のインターフェイスでは、`destination` で指定したアドレスに対して、メッセージ待ちインジケータを有効または無効にする必要があるかどうかが指定されます。`enable` が true の場合は、メッセージ待機が有効になります（すでに有効になっている場合はそのまま）。`enable` が false の場合は、メッセージ待機が無効になります（すでに無効になっている場合はそのまま）。

```
{
public void setMessageWaiting (java.lang.String destination, boolean enable)
    throws  javax.telephony.MethodNotSupportedException,
            javax.telephony.InvalidStateException,
            javax.telephony.PrivilegeViolationException
}
```

QuietClear

QuietClear は、Cisco Unified Communications Manager の停止、CTIPort を制御しているアプリケーションの停止、または CTIManager の停止などのネットワーク障害が原因で、コールの 2 つの通話者のうち、一方のアドレスがアウト オブ サービスになると、もう一方の端に示されます。この段階では、コールのもう一方の端では、コールの破棄または接続の解除だけが可能です。他の callControl 操作は実行できません。

アウト オブ サービスになった通話者については、アプリケーションには ConnDisconnectedEv や TermConnDroppedEv が送られ、コールのもう一方の端では、ConnFailedEv と CiscoCallEv.CAUSE_TEMPORARYFAILURE の CiscoCause が受信されます。

QuietClear モード時にアプリケーションによって次の機能の起動が試みられると、CiscoJtapiException.CTIERR_OPERATION_FAILER_QUIETCLEAR のエラー コードとともに PlatformException がスローされます。

- コンサルト転送
- コンサルト会議
- ブラインド転送
- 保留
- 保留解除

（注）　アプリケーションは、このモードではコールの破棄だけが可能です。

GetCallInfo

アドレス上の GetCallInfo インターフェイスでは、アドレスに関する CallInfo をアプリケーションによって問い合せることができます。問い合せは、CiscoAddressCallInfo オブジェクトを返します。これには、アクティブまたは保留のコールの数、アクティブまたは保留のコールの最大数、およびこのアドレスでの現行コールのコール オブジェクトに関する情報が含まれます。このインターフェイスは、どんなコールが特定の時刻に特定のアドレスにあるかも示します。

次のインターフェイスを使用して、端末に存在するコールに関する情報を取得します。

```
{ public CiscoAddressCallInfo getAddressCallInfo(Terminal iterminal);
}
```

DeleteCall

DeleteCall インターフェイスでは、アプリケーションが、createCall インターフェイスを使用して作成されたコールを削除できるようにします。このメソッドはコールを受け取り、プロバイダーがイン サービスでない場合、またはコールがアイドル状態ではない場合に InvalidStateException をスローします。DeleteCall により、コールが Invalid 状態に移行されます。

次のインターフェイスが CiscoProvider に追加されます。

```
{ public void deleteCall( Call call ) throws InvalidStateException;
}
```

アプリケーションでは、このインターフェイスを使用して createCall を使用して作成されたコールを削除できます。このメソッドはコールを受け取り、プロバイダーがイン サービスでない場合、またはコールがアイドル状態ではない場合に InvalidStateException をスローします。DeleteCall により、コールが Invalid 状態に移行されます。

コールを正常に削除するには、createCall を使用してアプリケーションでコールを作成し、コールをアイドル状態にする必要があります。

GetGlobalCallID

GetGlobalCallID では、CiscoCallID 上で、コールの nodeID および Global Call ID（GCID）を取得するインターフェイスを提供します。これは内部コール オブジェクトにある GCID 情報を公開します。

次のメソッドが、CiscoCallID インターフェイスに追加されます。

```
{   /**
    * returns the callmanager nodeID of the call
    */
public int getCallManagerID();
```

```
    /**
    * returns the GlobalCallID of the call
    */
public int getGlobalCallID ();
}
```

RTP イベントの GetCallID

GetCallID では、発側や着側などの任意のコール情報にアクセスし、アプリケーション側からコールの RTP イベントにリンクできるように、RTP イベント上のインターフェイスを提供します。

CTIManager からの RTP イベントで受信される callLegID は、クライアント側の ICCNCall の決定に使用されます。このコールは、JTAPI 層に渡され、CiscoCallID の取得元から CiscoCall が判定されます。この情報は、アプリケーションに配送される RTP イベントを構築するために使用されます。

次のインターフェイスが、`CiscoRTPInputStoppedEv`、`CiscoRTPInputStartedEv`、`CiscoRTPOutputStoppedEv`、および `CiscoRTPOutputStartedEv` に追加されます。

```
{ public CiscoCallID getCallID();
}
```

XSI オブジェクト パス スルー

アプリケーションでは、JTAPI と CTI のインターフェイス経由で XML オブジェクトを電話機に渡すことができます。XML オブジェクトには、IP フォン サービス機能から使用可能な電話機上の表示の更新、ソフトキーの更新、有効化、無効化および他のタイプの更新を含めることができます。これにより、電話機への独立した接続を維持することなく、JTAPI および CTI のインターフェイス経由で、アプリケーションから IP フォン サービス機能にアクセスできます。

CiscoTerminal メソッド

アプリケーションでは、CiscoTerminal インターフェイス メソッドを使用して、バイト形式の XSI オブジェクトを Cisco Unified IP Phone に送信できます。このインターフェイスでは、ペイロードが 2000 バイトのデータに制限されます。

CiscoTerminal は CiscoTerminal.REGISTERED 状態であり、そのプロバイダーは Provider.IN_SERVICE 状態である必要があります。正常な応答があった場合は、プッシュされたデータが電話に到着したことを示します。ただし、アプリケーションでは、プッシュ要求による CiscoIPPhoneResponse オブジェクトを含めて、電話から返送される XML を受信することはできない点に注意してください。アプリケーションの要求が成功しなかった場合は、PlatformException がスローされます。データ長が 2000 バイトを超える要求は、すべて拒否されます。

```
public String sendData ( String terminalData ) throws InvalidStateException,
MethodNotSupportedException;
```

この機能をアプリケーションで利用するには、事前に端末に関する TerminalObserver を追加する必要があります。

認証とメカニズム

電話の IP アドレスを必要とする HTTP POST 要求を電話機の Web サーバに送信すると、オブジェクトのプッシュが実行されます。Web サーバは、その要求を解析し、Cisco Unified Communications Manager に返された HTTP により要求を認証して、これを実行し、要求の成功または失敗を示す XML 応答をアプリケーションに返します。

XSI では、Skinny Client Control Protocol（SCCP）によって IP フォン サービス オブジェクトが電話に直接送信されます。JTAPI クライアントは信頼されており、電話の IP アドレスを必要としないので、電話では要求の認証は行われません。実際の XML 内容の詳細については、『*Cisco IP Phone Services Application Development Notes*』を参照してください。

Cisco VG248 および ATA 186 アナログ電話ゲートウェイ

Cisco Unified JTAPI は、Cisco VG248 および ATA 186 アナログ電話ゲートウェイに接続されているアナログ電話の制御をサポートします。Cisco VG248 および ATA 186 アナログ電話ゲートウェイをユーザ制御リストに追加すると、これらのデバイスをアプリケーションで制御できます。

アプリケーションには、他の IP フォンと同様の方法でデバイスに関するイベントが送信されます。また、アプリケーションでは、コールを発信できるほか、API による応答要求を除くその他の機能を呼び出すこともできます。発信が実行されるのは、デバイスが物理的にオフフック状態にある場合だけです。

デバイスに対する API からのコールにアプリケーションが応答できません。アプリケーションが VG248 および ATA 186 端末に対する TerminalConnection で () を応答しようとすると、CiscoJtapiException.COMMAND_NOT_IMPLEMENTED_ON_DEVICE というエラーを示す PlatformException が送出されます。コールに応答するには、受話器を手で持ち上げる必要があります。それによって、API による転送、会議、ブラインド転送、パークなど、その他のコール制御機能を呼び出せるようになります。

DN あたりの複数コール

DN あたりの複数コールは、1 回線（DN）上での複数コールと、これらのコール機能の操作をサポートする機能を表します。Cisco Unified Communications Manager リリース 4.0(1) よりも前では、最大 2 つのコールしかサポートされていませんでした。Cisco JTAPI は、1 回線あたり複数のコールをサポートします。これにより、同一回線上で複数のコールを扱い、この回線上で機能を操作することが可能になります。

DN あたりの複数コールによる、インターフェイスやメッセージ フローの変更はありません。

共用回線のサポート

共用回線は、複数端末での同じ DN のアピアランスを表します。CiscoJtapi は、共用回線をサポートします。共用回線では、同じ DN を持つ他の共用回線端末上で別のコールがアクティブになっていても、共有 DN 端末の制御、特定の共有 DN 端末にあるコールの保留と別の共有 DN 端末からの保留解除、2 本の共用回線間での発呼、特定の共用回線端末からのコール開始を、アプリケーション側から実行できます。

共用回線には、次のインターフェイスがあります。

- CiscoAddress.getInServiceAddrTerminals()：アドレスがイン サービス状態にある端末の配列を返します。

  ```
  Terminal {} getInServiceAddrTerminals();
  ```

- CiscoAddrOutOfService.getTerminal()：アウト オブ サービスの端末を返します。

  ```
  Terminal getTerminal();
  ```

- CiscoAddrInService.getTerminal()：イン サービスの端末を返します。

  ```
  Terminal  getTerminal();
  ```

- CiscoConnection.setRequestController(TerminalConnection tc)：Connection に関連付けられた terminalConnection をアプリケーション側で選択して、パーク、リダイレクト、または接続解除操作を実行できるようにします。SharedLine のシナリオで複数の TerminalConnection がアクティブになっている場合に、これが必要になります。
- CiscoConnection.getRequestController()：アプリケーションが要求コントローラとして設定する TerminalConnection を返します。
- CiscoAddrAddedToTerminalEv：次の条件が発生すると送信されます。
  - SharedDN を含む user controlList に端末またはデバイスが追加され、アプリケーションにイベントが送信された。つまり、ユーザのアドレスが制御リストにあり、制御リストの同じアドレスに新しいデバイスを追加した場合、このイベントが送信されます。
  - EM（エクステンション モビリティ）ユーザが、SharedDN を含むプロファイルを使用して端末にログインした。この場合、このイベントは、既存のアドレスに新しい端末が追加されたことを通知します。
  - ユーザ制御リスト内のデバイスに新しい SharedDN が追加された。

  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。
- CiscoAddrRemoveFromTerminalEv：次の条件が発生すると送信されます。
  - SharedDN を含む user controlList から端末またはデバイスが削除された。つまり、ユーザの制御リスト内の共有アドレスを使用したデバイスが制御リストから削除された場合、このイベントが送信されます。
  - SharedDN を含むプロファイルが使用されている端末から、EM（エクステンション モビリティ）ユーザがログアウトした。このイベントは、端末のうち 1 台が既存のアドレスから削除されたことをアプリケーションに通知します。
  - ユーザ制御リスト内のデバイスから新しい SharedDN（SharedLine）が削除された。

  getTerminal() インターフェイスは、アドレスから削除される端末を返します。

  getAddress() インターフェイスは、端末が削除されるアドレスを返します。

SharedLine の変更された動作または新しい動作を次に示します。

- CiscoAddress イベントの動作には、次の変更点があります。
  - JTAPI アプリケーションは、共用回線アドレスの複数の CiscoAddrInServiceEv を受信します。アプリケーションは、CiscoAddrInServiceEv.getTerminal() を使用して、アドレスがイン サービスの端末を取得できます。
  - JTAPI アプリケーションは、共用回線アドレスの複数の CiscoAddrOutOfServiceEv を受信します。アプリケーションは、CiscoAddrInServiceEv.getTerminal() を使用して、アドレスがアウト オブ サービスの端末を取得できます。
  - 最初の共用回線がイン サービスになると、アドレスの状態がイン サービスになります（最初の CiscoAddressInServiceEv が受信されたときなど）。
  - 最後の共用回線がアウト オブ サービスになると、アドレスの状態がアウト オブ サービスになります（最後の CiscoAddressOutOfServiceEv が受信されたときなど）。
- 着信コールの場合、共用回線のすべての回線アピアランスが鳴ります。これは、アプリケーションに対しては、1 つのアクティブ コール（callActiveEv）、1 つの Connection（ConnCreatedEv）、および複数の terminalConnection（共用回線あたり 1 件の TermConnCreatedEv）として示されます。

- コールは、すべての端末に対して示されます。コールが呼び出し状態になっている場合、TerminalConnection の状態は呼び出し中です。共用回線が応答すると、terminalConnection は Active 状態に、共用回線上にあるその他の terminalConnection は Passive 状態になり、その時点のすべての共用回線の callControlTerminalConnection は Bridged 状態になります。コールが保留にされると、すべての TerminalConnection は Active 状態に、callControllTerminalConnection は HELD 状態になります。この時点では、どの端末でもコールを取得できます。取得する側の端末では、terminalConnection が Active 状態のまま callControlTerminalConnection が Talking 状態になり、その他のすべての共用端末の terminalConnection は Passive 状態になります。同時に、CallControlTerminalConnection が HELD 状態から Bridged 状態に変わります。
- ある共用回線から、同じ DN を持つ別の共用回線に発呼できます。この場合、コールには 1 つの Connection と複数の TerminalConnection が含まれます。
- ある共用回線から同じ DN を持つ別の共用回線に発呼する場合、事後条件は 1 つの Connection だけになります。
- 2 つの terminalConnection がアクティブになっている共用回線 Connection（割込みなど）の場合、Connection.Disconnect() を実行しても接続解除されません。
- Passive 状態または Bridged 状態の TerminalConnection が 1 つしかない 1 つの SharedDN Connection だけをアプリケーションが監視している場合、この Connection に対してどんな API を呼び出しても PreConditionException が発生します。
- 前述のシナリオと同様に、アプリケーションが監視しているコールのすべての Connection に Passive 状態または Bridged 状態の TerminalConnection しかない場合、このコールに対するすべての API が PreConditionException をスローします（`Call.Drop()` など）。
- 共用回線上に複数の Active 状態の TerminalConnection がある場合、次のシナリオにおいて Call.drop() は CallInValid を返しません。
  - A が A' との SharedLine で、A' がコールに割り込んでいる、A と B の 2 者間コールの場合。

    アプリケーションは A' および B を監視していません。アプリケーションが Call.drop() を発行した場合、A’ の TerminalConnection は Passive 状態になりますが、コールは無効になりません。
  - 前述同様に、A、A'、A"、B が電話会議を行っている場合。

    アプリケーションは A および A' だけを監視するので、Call.drop() を実行してもコールは無効になりません。A および A' の TerminalConnection だけがパッシブになります。
  - A、A'、B、B' が SharedLine アドレスである場合。

    A が B にコールし、B が応答し、A' と B' がコールに割り込みます。アプリケーションは A および B だけを監視します。この場合、Call.drop() により A および B の TerminalConnection は Passive 状態になりますが、コールは無効になりません。
- TerminalConnection が Passive 状態または Bridged 状態、あるいは Passive/InUse 状態にある場合、TerminalConnection() ですべての API が PreConditionException をスローします。Passive または Bridged 状態では、TerminalConnection で API 端末の ConnectionJoin()（割り込みと呼ばれる）だけを使用できます。TerminalConnection は、現在 TerminalConnection Join() をサポートしていません。
- Connection 上に Active 状態または Talking 状態の複数の TerminalConnection がある場合、アプリケーションは一方を終了してから、この Connection に対して Redirect()、Park()、Disconnect() などの API を発行する必要がある可能性があります。Connection.setRequestController（TerminalConnection tc）API を使用して、TerminalConnection を選択できます。
- SharedLine 端末上のコールが保留され、アプリケーションが Connection.Disconnect () を発行した場合、アプリケーションは Connection.setRequestController（TerminalConnection tc）API を使用して特定の TerminalConnection を設定できます。requestcontroller が設定されていない場合、すべての HeldTerminalConnection が破棄され、Connection が解除された状態になります。1 つの

HeldConnection だけが破棄された場合、その他の SharedLines 端末上にコールが残ります。破棄された端末からはコール アピアランスがなくなるため、このコールに対して、この端末が割り込んだり、機能の操作に参加したりすることはできなくなります。

インターフェイスの変更点については、第 6 章「Cisco Unified JTAPI 拡張」を参照してください。共用回線のメッセージ フローを確認するには、付録 A「メッセージ シーケンスの図」を参照してください。

転送と直接転送

転送機能は、コールの転送機能を提供します。

直接転送機能は、回線上にある任意の 2 つのコールを転送する機能を示します。コールのコントローラがドロップし、その他の 2 者がアクティブ状態のままコールに残ることができます。この機能では、1 つの拡張機能がサポートされています。この機能は、コールがどの状態にあっても実行でき、新しい CTI イベントに合わせて再設計することもできます。転送機能には、次の拡張機能があります。

- アプリケーションから、保留中の 2 つのコールを転送する。
- アプリケーションが、OneHeld および OneConnected のコールをあらゆる順序で保持できる。
- アプリケーションから、回線上にある任意の 2 つのコールを転送する。

転送および直接転送の変更されたインターフェイスまたは新しいインターフェイスを次に示します。

- CiscoTransferStarted.getTransferControllers() : SharedLine 用の新しいインターフェイスです。SharedLine が TransferController の場合に、複数の terminalConnection をサポートします。transferController が SharedLine ではないときは、1 つの TerminalConnection だけがリストに含まれます。転送コントローラが監視されていない場合、このメソッドは null を返します。
- CiscoTransferStarted.getTransferController() : 現行インターフェイスであり、通常の転送では現状どおりに動作しますが、SharedLine では動作が異なる場合があります。transferController が SharedLine であるときには、複数の TerminalConnection があります。このメソッドは ACTIVE TerminalConnection を返します。ただし、アプリケーションが ACTIVE TerminalConnection を監視していない場合、このメソッドは PASSIVE TerminalConnection のうち 1 つを返します。
- CiscoTransferEnded isSuccess() : CiscoTransferEnded イベント用の新しいインターフェイスです。転送処理が正常に終了した場合は true を返し、転送に失敗した場合は false を返します。転送は次のイベントにより失敗する可能性があります。
  - CallProcessing が転送を完了する前に、一方がコールをドロップした。
  - CallProcessing が転送の完了を行えない。

JTAPI 転送の変更された動作または新しい動作を次に示します。

- 任意転送において、保留メッセージまたは保留解除メッセージが発生しません。
- 転送要求の事前条件が変更されている場合、コールがどの状態にあってもアプリケーションが転送を発行できます。
- アプリケーションに引数としてアクティブな TerminalConnection が渡されていない場合、Call.consult() は PreConditionException/InvalidArgumentException をスローします。
- コントローラにアクティブな TerminalConnection がない場合、Call.Transfer() は PreconditionException/InvalidArgumentException をスローします。

転送と直接転送のメッセージ フローを確認するには、付録 A「メッセージ シーケンスの図」を参照してください。

会議と参加

会議機能は、1 つのコールで 3 者以上の会議を行う機能を提供します。新しい CTI イベントをサポートするために、CTI 層のイベントが変更され、Cisco Unified JTAPI が拡張されています。

参加機能は、複数のコールを 1 つの電話会議に参加させる機能を提供します。この機能は複数のコールをサポートするようになりました。アプリケーションは、会議に参加するコールの配列を渡す必要があります。

会議と、1 つの電話会議への複数コールの参加処理用に、次の新しいインターフェイスまたは変更されたインターフェイスがあります。

- 次のインターフェイスを使用すると、1 つの電話会議に複数のコールの参加を行うことができます。

  ```
  Call.Conference(Call[] otherCalls)
  ```

  （注） 事前条件として、他のすべての otherCalls がコントローラをコールのレッグとして持つ必要があります。

- 次の新しいまたは変更されたインターフェイスが、CiscoConferenceStartEv にあります。
  - TerminalConnection getHeldConferenceController() : このインターフェイスは 2 つのコールの任意会議だけに有効で、保留されたコールのうち 1 つだけを返します。
  - TerminalConnection[] getHeldConferenceControllers() : このインターフェイスは、複数のコールを参加させる際にすべての保留中のコールを取得します。
  - TerminalConnection getTalkingConferenceController() : このインターフェイスは、会話中の会議コントローラを返します。ただし、会議に参加しているすべてのコールが保留され、会話中の会議コントローラがない場合、このインターフェイスは null を返します。
  - Call getConferencedCall() : このインターフェイスは、会議に参加する多数のコールのうち 1 つだけを返します。3 つ以上のコールがある場合の参加会議に関しては、無意味である可能性があります。
- CiscoConferenceEnded イベントの新しいインターフェイス、isSuccess() :

  このインターフェイスは、会議 が正常に実行されたか失敗したかに応じて、True または False を返します。アプリケーションは、このインターフェイスを使用して、会議が正常に実行されたかどうかを確認できます。次のイベントは会議の失敗として定義されます。
  - アプリケーションが Call.conference(otherCalls[]) 要求を発行し、1 つ以上のコールが会議に参加できなかった場合、この会議は失敗と見なされます。アプリケーションは、getFailedCalls() インターフェイスを使用して、失敗したコールを見つけることができます。
  - 会議ブリッジがなく、会議がまったく開催できなかった場合、アプリケーションは、getFailedCalls() を使用して、会議に参加できなかったコールのリストを取得できます。
  - 会議が開催できる前に会議に参加していた側がドロップした。
- CiscoConferenceEnded イベントのインターフェイス（Call[] getFailedCalls()）は、会議が失敗した際に、会議への参加に失敗したすべてのコールを取得します。

会議の新しい動作または変更された動作を次に示します。

- 任意会議が発生した時点でアプリケーションが確認できるような保留中または保留解除のメッセージはありません。
- 任意会議で、いずれかのコールが通話中状態にある必要があるという事前条件はなくなりました。ただし、他のすべての otherCalls がコントローラをコールのレッグとして持つ必要があります。

- アプリケーションは、2 つ以上の保留中のコールを電話会議に追加できます。finalCall では、コントローラが自動的に Talking 状態に戻ります。
- Call.Conference(otherCalls) 要求にはアクティブ コールを必ず含めてください。会議要求にアクティブ コールが含まれていない場合、要求が失敗します。
- コントローラにアクティブ コールがない場合、Call.Conference(otherCalls) 要求は正常に実行されます。ただし、アクティブ コールが 1 つある場合は、要求に含める必要があります。
- アプリケーションに引数としてアクティブな TerminalConnection が渡されていない場合、Call.consult() は PreConditionException/InvalidArgumentException をスローします。
- コントローラにアクティブな TerminalConnection がない場合、Call.Conference()/Call.Conference(Call[]) は PreconditionException/InvalidArgumentException をスローします。

インターフェイスの変更点については、第 6 章「Cisco Unified JTAPI 拡張」を参照してください。会議と参加のメッセージ フローを確認するには、付録 A「メッセージ シーケンスの図」を参照してください。

割り込みとプライバシー イベント通知

割り込み機能は、別端末のアドレスで確立されたコールに共用アドレスが割り込み処理できる機能です。この機能は、TerminalConnection アドレスが Passive 状態で、CallCtlTerminalConnection が Bridged 状態である場合に有効です。Cisco Unified JTAPI のこのバージョンでは、アプリケーション制御による端末（IP フォン）から手動で機能を有効にすることだけがサポートされます。このリリースでは、API から機能を有効にすることはできません。

プライバシー機能は、他の共有アドレスからコールへの割り込み処理を有効または無効にできる機能です。プライバシーを有効にすると、他の共有アドレスからコールに割り込めません。また、無効にすると、割り込みができます。プライバシーは、端末のプロパティを示します。IP フォンには「プライバシー」ソフトキーがあり、このソフトキーを押すと、プライバシーが有効または無効になります。端末のアクティブ コールの場合、プライバシーを動的に有効または無効にすることができます。コールのプライバシーがオンになっているときは、共有アドレス上にあるコール アピアランスの TerminalConnection は「InUse」状態です。CallProgress 中にプライバシー ステータスが変化した場合、CiscoTermConnPrivacyChangedEvent がアプリケーションに配信されます。

Cisco Unified Communications Manager には 2 種類の割り込み機能があります。「割り込み」と呼ばれる組み込みの会議ブリッジと、「C 割り込み」と呼ばれる共用会議ブリッジ リソースが使用されます。アプリケーションにとっては、割り込みと C 割り込みの間でのインターフェイスの変更はありません。ただし、一部動作の違いがあり、これらは付録 A「メッセージ シーケンスの図」にあるメッセージ フローの図に記載されています。

割り込み、C 割り込みおよびプライバシーには次のインターフェイスがあります。

Interface CiscoTerminalConnection.getPrivacyStatus()

```
booleangetPrivacyStatus()
```

このインターフェイスは、端末上にあるコールのプライバシー ステータスを返します。

Interface CiscoTermConnPrivacyChangedEv

```
javax.telephony.TerminalConnectiongetTerminalConnection()
```

CiscoCall.CAUSE_BARGE という新しい原因コードが、割り込みの CiscoCall に追加されています。

割り込みの結果、SharedLine TerminalConnection または CallCtiTerminalConnection が Active 状態または Talking 状態になると、JTAPI は CiscoCall.CAUSE_BARGE として CallCtiCause を提供します。この原因コードは、割り込み処理中に作成される一時コールをドロップする際に使用される CallCtiEvent でも提供されます。

C 割り込みでは、この原因コードは提供されません。

これらのインターフェイスの詳細については、第 6 章「Cisco Unified JTAPI 拡張」を参照してください。割り込み、C 割り込み、およびプライバシーのメッセージ フローを確認するには、付録 A「メッセージ シーケンスの図」を参照してください。

CallSelect と UnSelect のイベント通知

直接転送または参加などの操作を行うために、電話のコールを選択または選択解除できます。SharedLine ユーザがコールを選択すると、回線を共用している他の TerminalConnection は、Passive 状態になり、CallCtlTermiCallConnection は InUse 状態になります。コールの選択が解除されると、CallCtlTerminalConnection が Bridged 状態になります。Passive または InUse の TerminalConnection については、アプリケーションから API を呼び出せません。（転送/会議などの）機能動作中の CallProcessing でも、Select/UnSelect 操作を実行できます。アプリケーションが RemoteInUse 端末を監視している場合、アプリケーションは次のイベントも受け取ります。

たとえば、A と A' が SharedLine の場合、A がコールを選択すると、A' の CallCtlTerminalConnection は Passive 状態または InUse 状態になります。A がコールの UnSelect を実行した場合、A' の CallCtlTerminalConneciton は Passive 状態または Bridged 状態になります。

CallSelect または UnSelect に関するメッセージ フローを確認するには、付録 A「メッセージ シーケンスの図」を参照してください。

動的 CTIPort 登録

この機能により、コールごとに、またはメディアが確立されるたびに、アプリケーションから IP アドレス（ipAddress）およびポート番号（portNumber）を提供することが可能になります。この機能を使用するには、メディア機能を提示して、アプリケーションでメディア端末を登録する必要があります。このメディア端末でコールが応答されると、アプリケーションに CiscoMediaOpenLogicalChannelEv が送信されます。このイベントは、メディアが確立されるたびに送信されます。アプリケーションはこのイベントに応答し、メディアが確立される IP アドレスとポート番号を示す必要があります。

CiscoMediaTerminal は、アプリケーションによる RTP メディア ストリームの終端を可能にする、特殊な CiscoTerminal を表します。CiscoMediaTerminal は、通常の CiscoTerminal と異なり、物理的なテレフォニー エンドポイントではないため、サードパーティ方式で監視、制御することができます。CiscoMediaTerminal は論理的なテレフォニー エンドポイントを表し、メディアを終端するアプリケーションに関連付けることができます。そのようなアプリケーションには、音声メッセージング システム、Interactive Voice Response（IVR; 対話式音声自動応答）、ソフトフォンなどが含まれます。

（注）　Cisco Unified JTAPI により CiscoMediaTerminal として表現されるのは、CTIPort だけです。

メディアの終端プロセスには 2 つの段階があります。アプリケーションはまず、特定の端末向けのメディアを終端するために、Terminal.addObserver メソッドを使用して、CiscoTerminalObserver インターフェイスを実装するオブザーバを追加します。次に、CiscoMediaTerminal.register メソッドを使用して、その IP アドレス、およびその端末向けの着信 RTP ストリームの送信先となるポート番号を登録します。

コールごとに動的に ipAddress および portNumber を登録するには、アプリケーションは、サポートしている機能だけを提示して登録する必要があります。アプリケーションは、メディアが確立されるたびに送信される CiscoMediaOpenLogicalChannelEv に応答する必要があります。アプリケーションが CiscoMediaOpenLogicalChannelEv に応答する前に何らかの機能が実行された場合、機能が失敗する可能性があります。

Cisco Unified Communications Manager Administration ウィンドウにある Media Exchange Timer に指定された時間内にアプリケーションがこのイベントに応答しない場合、コールが失敗する可能性があります。

インターフェイスの変更点については、第 6 章「Cisco Unified JTAPI 拡張」を参照してください。コールごとの CTIPort 動的登録のメッセージ フローを確認するには、付録 A「メッセージ シーケンスの図」を参照してください。

（注）　ChangeRTPDefaults インターフェイスは CiscoMediaTerminal ではサポートされていません。

コールごとの CTIPort 動的登録には、次の新しいインターフェイスまたは変更されたインターフェイスがあります。

Interface CiscoMediaOpenLogicalChannelEv Extends CiscoTermEv

int　**getpacketSize**()

遠端のパケット サイズを返します（ミリ秒単位）。

int　**getPayLoadType**()

次の定数の 1 つの遠端ペイロード形式を返します。

CiscoRTPHandle　**getCiscoRTPHandle()**

アプリケーションが setRTPParams 要求を呼び出す必要のある CiscoTerminalConnection オブジェクトを返します。

Interface CiscoRTPHandle

int　**getHandle()**

このオブジェクト（現在は Cisco Unified Communications Manager CallLeg ID）の整数表現を返します。

CiscoProvider

CiscoCall　**getCall (CiscoRTPHandle rtpHandle)**

特定の端末に関連付けられた rtpHandle を持つコール オブジェクトを返します。アプリケーションが CallOpenLogicalChannelEv で CiscoRTPHandle を受信した時点で、コール オブサーバが端末に追加されない場合は、CiscoCall が null である可能性があります。

ルート ポイントでのメディア終端

この機能により、コールごとに、またはメディアが確立されるたびに、IP アドレスおよびポート番号を指定することで、ルート ポイントおよびアプリケーションにあるすべてのアクティブ コールを終端することが可能になります。
この機能を使用するには、メディア機能を提示して、アプリケーションでルート ポイントを登録する必要があります。このルート ポイントでコールが応答されると、アプリケーションに CiscoMediaOpenLogicalChannelEv が送信されます。このイベントは、メディアが確立されるたびに送信されます。アプリケーションはこのイベントに応答し、メディアの終端先となる IP アドレスとポート番号を示す必要があります。

CiscoRouteTerminal は、アプリケーションによる RTP メディア ストリームの終端を可能にする、特殊な CiscoTerminal を表します。CiscoRouteTerminal は、通常の CiscoTerminal と異なり、物理的なテレフォニー エンドポイントではないため、サードパーティ方式で監視、制御することができます。CiscoRouteTerminal は論理的なテレフォニー エンドポイントを表し、コールをルーティングしメディアを終端する必要があるアプリケーションに関連付けることができます。CiscoMediaTerminal とは異なり、CiscoRouteTerminal では複数のアクティブ コールを同時に保持できます。通常、CiscoRouteTerminal は、エージェントが次の発信者に応対できるようになるまでの間、コールをキューに格納するために使用されます。

（注） JTAPI により CiscoRouteTerminal として表現されるのは、RoutePoint 端末だけです。

メディアの終端プロセスには 3 つの段階があります。

ステップ 1　アプリケーションが、CiscoRouteTerminal.register メソッドを使用して、メディア機能をこの端末に登録します。

ステップ 2　CiscoTerminalObserver インターフェイスを実装するオブザーバを、アプリケーションが Terminal.addObserver メソッドを使用して追加します。

ステップ 3　CiscoRTPHandle を使用してプロバイダーから CiscoCall オブジェクトを受信するために、アプリケーションは、CiscoRouteTerminal または CiscoRouteAddress に addCallObserver を追加する必要があります。

アプリケーションは、コールごとに CiscoMediaOpenLogicalChannelEv を受信するので、CiscoRouteTerminal の setRTPParams メソッドを使用して、IP アドレスおよびポート番号を提示する必要があります。

CiscoJtapiClient 1.4(x) リリース以前で作成されたすべてのアプリケーションは、メディア終端を必要としない場合、修正して CiscoRouteTerminal.NO_MEDIA_TERMINATION に登録する必要があります。

登録されるメディア機能および registrationType が同じであれば、複数のアプリケーションを同じルート ポイントに登録できます。CiscoRouteTerminal.DYNAMIC_MEDIA_REGISTRATION に登録され、端末オブザーバが追加されたすべてのアプリケーションは、CiscoMediaOpenLogicalChannelEv を受信します。ただし、1 つのアプリケーションだけが setRTPParams を呼び出すことができます。

（注） メディアを終端するアプリケーションは、CallControl パッケージを使用して、コールの応答およびリダイレクトを行う必要があります。コールのルーティングだけを行うアプリケーションは、ルーティング パッケージを使用できます。

（注）　アプリケーションが CiscoMediaOpenLogicalChannelEv に応答する前に何らかの機能が実行された場合、機能が失敗する可能性があることが、アプリケーション側で認識されている必要があります。Cisco Unified Communications Manager Administration ウィンドウにある Media Exchange Timeout パラメータに指定された時間内にアプリケーションがこのイベントに応答しない場合、コールが失敗する可能性があります。

ルート ポイントでのメディア終端用に、次の新しいインターフェイスまたは変更されたインターフェイスがあります

Interface CiscoRouteTerminal Extends CiscoTerminal

boolean　**isRegistered()**

CiscoMediaTerminal が登録された場合、このメソッドは true を返します。それ以外の場合は false を指定します。

boolean　**isRegisteredByThisApp()**

アプリケーションが正常に登録要求を発行した場合、このメソッドは true を返します。また、アプリケーションがデバイスを登録解除するまでの間、true のままになります。これは、CTIManager の不具合によりデバイスがアウト オブ サービス状態にある場合でも有効です。

void　**register** `(CiscoMediaCapability[] capabilities, int registrationType)`

CiscoRouteTerminal は CiscoTerminal.UNREGISTERED 状態で存在する必要があり、プロバイダーは Provider.IN_SERVICE 状態で存在する必要があります。

void　**setRTPParams** `(CiscoRTPHandle rtphandle, CiscoRTPParams rtpParams)`

アプリケーションは、ipAddress および RTP ポート番号を設定して、コールのメディア ストリームを動的に行います。

void　**Unregister()**

CiscoRouteTerminal が登録されており、プロバイダーが Provider.IN_SERVICE 状態であることを確認してください。

Interface CiscoMediaOpenLogicalChannelEv Extends CiscoTermEv

int　**getpacketSize**()

遠端のパケット サイズを返します（ミリ秒単位）。

int　**getPayLoadType**()

次の定数の 1 つの遠端ペイロード形式を返します。

CiscoRTPHandle　**getCiscoRTPHandle()**

アプリケーションが setRTPParams 要求を呼び出す必要のある CiscoTerminalConnection オブジェクトを返します。

Interface CiscoRTPHandle

int　**getHandle()**

このオブジェクト（現在は Cisco Unified Communications Manager CallLeg ID）の整数表現を返します。

CiscoProvider

CiscoCall　**getCall (CiscoRTPHandle rtpHandle)**

特定の端末に関連付けられた rtpHandle を持つコール オブジェクトを返します。アプリケーションが CallOpenLogicalChannelEv で CiscoRTPHandle を受信した時点で、コール オブサーバが端末に追加されない場合は、CiscoCall が null を登録する可能性があります。

これらのインターフェイスの詳細については、第 6 章「Cisco Unified JTAPI 拡張」を参照してください。ルート ポイントでのメディア終端のメッセージ フローを確認するには、付録 A「メッセージ シーケンスの図」を参照してください。

リダイレクト時の元の着信者 ID のセット

Cisco Unified JTAPI アプリケーションは、優先される元の着信側 DN をリダイレクト要求内に指定できます。リダイレクト時の元の着信者 ID のセット機能により、別の宛先への Connection 上にあるコールをリダイレクトすることに加えて、アプリケーションが OriginalCalledID に任意の値を設定できます。これにより、アプリケーションは、コールを別のボイス メールへ直接転送できます。たとえば、A が B にコールし、B がコールを C Voice Mail へ転送する場合、アプリケーションは拡張リダイレクト要求内で、優先される元の着信者に C を、また C Voice Mail プロファイルに宛先を指定できます。この要求を使用すると、Cisco Unified Communications Manager の originalCalledParty フィールドに C が指定されたコールが C Voice Mail プロファイルに表示されます。一般的なボイスメール アプリケーションは、originalCalledParty 情報を調べてユーザのボイス メールボックスを特定します。

この機能は、元の着信者を変更してコールを特定の通話者へリダイレクトするあらゆるアプリケーションで使用できます。

（注）　この機能では、lastRedirectedAddress も、リダイレクト要求に指定された preferredOriginalCalledParty に変更されます。

次の callControlConnection インターフェイスが、リダイレクト時の元の着信者 ID のセットに適用されます。

Interface CiscoConnection Extends callControlConnection With Additional Cisco Unified Communications Manager-Specific Capabilities

`javax.telephony.Connection`　**redirect (java.lang.String destinationAddress, int mode, int callingSearchSpace, java.lang.String preferredOriginalCalledParty)**

このメソッドは CallControlConnection.redirect() メソッドをオーバーロードします。

インターフェイスの詳細については、第 6 章「Cisco Unified JTAPI 拡張」を参照してください。リダイレクト時の元の着信者 ID のセットのメッセージ フローを確認するには、付録 A「メッセージ シーケンスの図」を参照してください。

シングル ステップ転送

このインターフェイスを使用すると、アプリケーションが特定のアドレスにコールを転送できます。Cisco Unified JTAPI は、JTAPI 1.2 仕様の定義に従って引き続きこのインターフェイスをサポートします。ただし、アプリケーションに配信されるイベントが Cisco Unified JTAPI の以前のバージョンから変更されています。

Cisco Unified JTAPI の以前のバージョンでは、このインターフェイスが使用されると、元のコールは保留状態になり、転送コントローラと宛先の間で新しいコールが作成されました。転送が正常に終了すると、転送コントローラ上の両方のコールが IDLE 状態になります。転送に失敗した場合、元のコールは HELD 状態のままになり、アプリケーションがコールを取得します。転送処理の開始時および完了時に、CiscoTransferStart および終了イベントがアプリケーションに配信されます。

アプリケーションには次のような変更があります。

- 新規コールが作成されない。
- CiscoTransferStartEv および CiscoTransferEndEv がアプリケーションに送信されない。
- 転送操作が失敗した場合も元のコールの状態が維持される。

このインターフェイスの事前条件および事後条件に変更はありません。

シングル ステップ転送に関するメッセージ フローを確認するには、付録 A「メッセージ シーケンスの図」を参照してください。

API の自動アップデート

Cisco Unified Communications Manager を上位バージョンにアップグレードした場合、Cisco Unified Communications Manager の新しいバージョンとは API の互換性がないことに注意する必要があります。アプリケーションが期待どおりに動作できるように、API を互換バージョンにアップグレードしていることを確認してください。API はクライアント サーバにローカルでインストールされるため、複数のマシンをアップグレードする必要があります。クライアント アプリケーションの数が少ない場合は、Cisco Unified Communications Manager Administration に接続し、Cisco Unified Communications Manager 互換プラグインのダウンロードとインストールを行うことで、簡単に実施できます。

クライアント アプリケーションが複数ある場合、この機能は、アプリケーションが始動時に HTTP 要求を使用して自身の ID を Web サーバに提示し、必要な JTAPI API のバージョンに関する応答を受信するための機能を提供します。

アプリケーションのクラスパスにあるローカルのバージョンと、サーバから取得できるバージョンが比較され、アップグレードが必要かどうかが確認されます。これにより、Cisco Unified Communications Manager に合わせてアプリケーションが jtapi.jar コンポーネントをリフレッシュでき、アプリケーションが自動アップデート可能な jtapi.jar を集中展開することができます。

この機能の実行に必要な API は、updater.jar 形式でパッケージ化されています。jtapi.jar および updater.jar のパッケージには、バージョンの比較に使用できる標準マニフェストが含まれています。

（注）　この機能では、JTAPI Preferences、JTAPITestTools、Updater.jar、javadoc コンポーネントはアップデートされません。アプリケーションにこれらのコンポーネントが必要な場合は、Cisco Unified Communications Manager プラグイン ページから JTAPI をインストールしてください。自動アップデートは、JTAPI リリース 2.0 以降をサポートしています。

詳細については、第 4 章「Cisco Unified JTAPI のインストール」を参照してください。

API の自動アップデートには、次の新しいインターフェイスまたは変更されたインターフェイスがあります。

Class com.cisco.services.updater.ComponentUpdater

Component　**queryLocalComponentVersion** (java.lang.String componentName, java.lang.String path)

IOException、IllegalArgumentException をスローします。

Component　**queryServerComponentVersion** (java.lang.String componentName, java.lang.String urlString)

IOException、IllegalArgumentException をスローし、HTTP クエリーをサーバに送信して、リモート サーバにインストールされているコンポーネントのバージョンを判別します。

Interface com.cisco.services.updater.Component

int　**compareTo** (Component otherComponent)

Component　**fetchFromServer**()

HTTP によりコンポーネントをサーバから取得し、temp.jar というファイル名を付けてローカル ファイル システム上のローカル ディレクトリに書き込みます。

java.lang.String　**getBuildDescription ()**

「a.b(c.d) Release」形式の文字列「Release」にあたるバージョンを返します。

int　**getBuildNumber** ()

a.b(c.d) 形式の「d」にあたるバージョンを返します。

java.lang.String　**getLocation ()**

コンポーネントの位置（文字列形式）。

int　**getMajorVersion** ()

a.b(c.d) 形式のバージョン「a」にあたるバージョンを返します。

int　　**getMinorVersion** ()

a.b(c.d) 形式のバージョン「b」にあたるバージョンを返します。

java.lang.String　　**getName ()**

コンポーネント名を返します。

int　　**getRevisionNumber** ()

a.b(c.d) 形式の「c」にあたるバージョンを返します。

また、JTAPI の自動アップデータ機能により、アプリケーションは Cisco Unified Communications Manager から JTAPI.JAR の最新バージョンを直接ダウンロードできます。

1. アップデータにより、アプリケーションの現在のフォルダに newjtapi.jar ファイルが作成されます。これは、Cisco Unified Communications Manager からダウンロードされた新しいバージョンの jar ファイルを表します。
2. アップデータにより、指定されたクラスパスの component.temp というファイルに現在の jtapi.jar がコピーされます。
3. アップデータにより、現在の jtapi.jar が新しい jtapi.jar ファイルに置換されます。

この処理が終わると、現在の jar ファイルは component.temp になり、新しい jar ファイルは jtapi.jar になります。この処理は、Linux と Windows の両方でサポートされています。

自動アップデータの使用例

```
Command Line : java com.cisco.services.updater.ComponentUpdater <server> <component name>
<login> <passwd>
    Component localComponent, downloadedComponent;
    ComponentUpdater updater = new ComponentUpdater();
    String localPath = updater.getLocalComponentPath(args[1]);
    localComponent = updater.queryLocalComponentVersion("jtapi.jar",localPath);
    localComponent.copyTo("component.temp");
    String provString = args[0] + ";login=" + args[2] + ";passwd=" + args[3];

    CiscoJtapiPeer peer = (CiscoJtapiPeer) (JtapiPeerFactory.getJtapiPeer(null));
    CiscoJtapiProperties tempProp = ( (CiscoJtapiPeerImpl) (peer)). getJtapiProperties();
    tempProp.setLightWeightProvider(true);

    Provider provider = peer.getProvider(provString);
    String url = ( (CiscoProvider) (provider)).getJTAPIURL(); provider.shutdown();
    Component serverComponent = updater.queryServerComponentVersion("jtapi.jar", url);

    downloadedComponent = serverComponent.fetchFromServer();
    int retVal = downloadedComponent.replaces(localComponent);
```

「replaces」API は既存の JTAPI バージョンを新しいバージョンに置き換えます。

（注） アップデータは JTAPI.JAR ファイルだけをアップデートし、JTAPI プラグインにバンドルされている他のサンプル アプリケーションや Cisco JTAPI のドキュメンテーションはアップデートしません。これらの別コンポーネントを入手するには、アプリケーションがプラグインを Cisco Unified Communications Manager からダウンロードしてインストールする必要があります。

デバイス タイプ名の処理に関する変更

現在、TSP ではデバイス タイプ名がデバイス タイプに応じてハードコーディングされています。新しいデバイス タイプが追加された場合は、新しいデバイス タイプ名をサポート対象デバイスのリストに手動で追加する必要があります。CTI はデバイス タイプ名を CTI 自身のキャッシュに取得して格納することがないため、TSP ではこの情報を CTI から取得できません。TSP では、手動による操作がないまま新しいデバイス タイプが追加されたときに、デバイス タイプ名を更新する必要があります。

CTI から送られ、deviceInfo 構造体に格納されているデバイス タイプ名の受信が処理されるように QBE インターフェイスの変更が JTAPI に加えられました。deviceInfo は JTAPI ではまったく使用されておらず、アプリケーションにも公開されません。QBE インターフェイスだけが次のように変更されました。

```
public DeviceRegisteredEvent ( String deviceName, int deviceType, boolean
allowsRegistration, int deviceID, boolean loginAllowed, UnicodeString userID, boolean
controlled, int reasonInt, int registrationType, int unicodeEnabled,int locale,

// added for deviceTypeName change
String devTypeName) {

public DeviceUnregisteredEvent ( String deviceName, int deviceType, boolean
allowsRegistration, int deviceID, UnicodeString userID, boolean controllableBool, int
reasonInt , int locale,
//added for devtypename support
String devTypeName) {
```

CiscoTerminal Filter と ButtonPressedEvents

JTAPI 2.0 リリースよりも前では、Cisco Unified JTAPI アプリケーションは端末イベントを直接制御できませんでした。端末オブザーバに適切なフィルタを設定することで、アプリケーションがボタン押下イベントを受信できるようになりました。アプリケーションで RTP イベントを取得するために、コール オブザーバを追加する必要はなくなりました。

CiscoTermEvFilter を使用して setButtonPressedEv が有効にされている場合、電話機で数字ボタンが押されたときに、アプリケーションは CiscoTermButtonPressedEv を受信します。

CiscoTerminal Filter と ButtonPressedEvent には、次の新しいインターフェイスまたは変更されたインターフェイスがあります

CiscoTerminal

void　　**setFilter (CiscoTermEvFilter terminalEvFilter)**

TerminalObserver に配信されるイベントをアプリケーションがより詳細に制御できるようにします。

CiscoTermEvFilter

boolean　**getButtonPressedEnabled()**

端末のボタン押下イベントに関して、有効または無効のステータスを取得します。デフォルトでは、無効に指定されています。

boolean　**getDeviceDataEnabled()**

端末のデバイス データ イベントに関して、有効または無効のステータスを取得します。デフォルトでは、無効に指定されています。

boolean　**getRTPEventsEnabled()**

端末の RTP イベントに関して、有効または無効のステータスを取得します。デフォルトでは、無効に指定されています。

void　**setButtonPressedEnabled** (boolean enabled)

端末のボタン押下イベントを有効または無効に設定します。

void　**setDeviceDataEnabled** (boolean enabled)

端末のデバイス データ ステータス イベントを Enable または Disable に設定します。

void　**setRTPEventsEnabled (boolean enabled)**

端末の RTP イベントを有効または無効に設定します。

CiscoTermButtonPressedEv

int　**getButtonPressed** ()

インターフェイスの変更点については、第 6 章「Cisco Unified JTAPI 拡張」を参照してください。CiscoTerminal Filter と ButtonPressedEvent のメッセージ フローを確認するには、付録 A「メッセージ シーケンスの図」を参照してください。

発信側番号の変更

この機能は、ルート ポイントから SelectRoute API の発信者 DN を変更することを可能にします。アプリケーションは、selectRoute API で発信側番号変更の配列を渡すことができます。発信側番号変更の配列の長さは、選択されたルートの長さと等しくすることができます。対応する routeSelected インデックスに発信側番号変更の要素がない場合、または要素が null である場合、このルート選択要素には発信者番号変更が設定されません。

2 種類の新しいインターフェイスである getModifiedCallingAddress () と getModifiedCalledAddress () が、発信者または着信者の変更された番号を返すコール オブジェクトに公開されます。変更がない場合、これらのインターフェイスは、getCurrentCallingAddress () インターフェイスおよび getCurrentCalledAddress () インターフェイスと同じ値を返す可能性があります。アプリケーションは、selectRoute API を使用してルート ポイントの制御と発信者番号の変更を行っているだけである場合、

getModifiedCallingAddress インターフェイスの発信者アドレス変更を取得できない可能性があります。アプリケーションは、なんらかの発信者または着信者を制御している場合、発信者番号の変更後にコール制御イベントを受信すれば、正しい値を取得できます。

新たに getRouteSelectedIndex () インターフェイスが、新しい CiscoRouteUsedEvent クラスで公開されました。このクラスは、RouteUsedEvent の拡張で、選択されたルートのインデックスを提供します。このメソッドにアクセスするには、アプリケーションは RouteUsedEvent から CiscoRouteUsedEvent へキャストする必要があります。

例

```
routeSelected[0] = 133555
routeSelected[1] = 144911
routeSelected[2] = 143911
routeSelected[3] = 5005

modifiedCallingNumber[0] =null
modifiedCallingNumber[1] =9721234567
modifiedCallingNumber[2] =9721234568
modifiedCallingNumber[3] =null
```

ルーティングで routeSelected[0] または routeSelected[3] が選択された場合、発信者番号の変更が適用されない可能性があります。

この機能は、特定のユーザの Cisco Unified Communications Manager Administration にある [発信側番号の変更] チェックボックス（デフォルトでは [いいえ]）を管理者が有効にした後にだけ使用できます。これが設定されていない場合、RouteSession.CAUSE_PARAMETER_NOT_SUPPORTED という原因の RerouteEvent がアプリケーションに送信されます。発信者番号を変更するアプリケーションでは、着信者の表示名が影響されることと、それに続く発信者または着信者の機能の対話が不整合が生じる場合があることに注意する必要があります。

発信者番号の変更には、次の新しいインターフェイスまたは変更されたインターフェイスがあります。

CiscoRouteSession

void　　**selectRoute** (java.lang.String[] routeSelected, int callingSearchSpace, String[] modifiedCallingNumber)

このインターフェイスは、アプリケーションが発信者番号を routeSelected アドレスに変更できるようにします。対応する routeSelected 要素に modifiedCallingNumber 要素がない場合、特定の routeSelected 要素にコールがルーティングされた際に発信者番号が変更されません。

CiscoCall

javax.telephony.Address　**getModifiedCalledAddress ()**

アプリケーションが selectRoute API を使用して発信者を変更した場合、このインターフェイスは変更された着信側アドレスを返します。ただし、アプリケーションが、発信者番号を変更するルート ポイントを制御しているだけである場合、この情報は正確ではない可能性があります。発信者番号の変更が行われない場合は、getCurrentCalledAddress インターフェイスと同様に機能します。これは、通常、発信側番号が変更された後に機能が呼び出された場合に getCurrentCalledAddress とは異なります。

javax.telephony.Address　**getModifiedCallingAddress ()**

アプリケーションが selectRoute API を使用して発信者を変更した場合、このインターフェイスは変更された発信側アドレスを返します。ただし、アプリケーションが、発信者番号を変更するルート ポイントを制御しているだけである場合、この情報は正確ではない可能性があります。発信者番号の変更が行われない場合は、このインターフェイスは getCurrentCallingAddress インターフェイスと同様に機能します。

CiscoRouteUsedEvent

int　**getRouteSelectedIndex** ()

このメソッドは、コールのルーティング先となるルートの配列インデックスを返します。

インターフェイスの変更点については、第 6 章「Cisco Unified JTAPI 拡張」を参照してください。発信者番号の変更のメッセージ フローを確認するには、付録 A「メッセージ シーケンスの図」を参照してください。

CTI ポートおよびルート ポイントの AutoAccept のサポート

この機能は、CTIPort およびルート ポイントのアドレスの AutoAccept を有効または無効にする機能をアプリケーションに提供します。アドレスの AutoAccept ステータスが変更されると、Cisco Unified JTAPI はイベントを提供してアプリケーションに変更を通知します。

（注）　ルート ポイントでサポートされる回線の最大数は 34 です。

CiscoAddress オブジェクト内に提供された新しい setAutoAcceptStatus() インターフェイスを使用すると、AutoAccept をオンまたはオフに設定できます。同様に、CiscoAddress オブジェクト内に提供された getAutoAcceptStatus() インターフェイスを使用すると、アプリケーションはアドレスの現在の AutoAccept ステータスを照会できます。

アドレスの AutoAccept ステータスが変更されると、アプリケーションは AddressObserver によって CiscoAddrAutoAcceptStatusChangedEv を取得します。このイベントには、AutoAccept ステータスが変更された端末を返す getTerminal() インターフェイスと、AutoAccept がオンまたはオフであるかを示す整数を返す getAutoAcceptStatus() が含まれます。アドレス オブザーバが追加されていない場合、このイベントは提供されません。

次のインターフェイスは、CTIPort および RoutePoint の AutoAccept をサポートしています。

CiscoAddress

int　**getAutoAcceptStatus** (javax.telephony.Terminal terminal)

Ciscoaddress.getAutoAccept(Terminal iterminal) は、端末のアドレスに関する AutoAccept ステータスを返します。

void　**setAutoAcceptStatus** (int autoAcceptStatus, javax.telephony.Terminal terminal)

これにより、アプリケーションは CiscoMediaTerminal や CiscoRouteTerminal のアドレスについて AutoAccept を有効にできます。

CiscoAddrAutoAcceptStatusChangedEv

Public interface: **CiscoAddrAutoAcceptStatusChangedEv**

Extends com.cisco.jtapi.exension.CiscoAddrEv

CiscoAddrAutoAcceptStatusChangedEv イベントは、端末のアドレスの AutoAccept ステータスが変更されるたびにアプリケーションに送信されます。アドレスに複数の端末がある場合、このイベントは個々の端末のアドレスの AutoAccept ステータスごとに送信されます。

このイベントには、次のインターフェイスがあります。

int　**getAutoAcceptStatus** ()

CiscoAddrAutoAcceptStatusChangedEv.getAutoAcceptStatus は、CiscoAddress.AUTOACCEPT_OFF CiscoAddress.AUTOACCEPT_ON 端末上のアドレスの AutoAccept ステータス値を次のように返します。

com.cisco.jtapi.extensions.CiscoTerminal　**getTerminal** ()

このアドレスの AutoAccept ステータスが変更された端末を返します。

インターフェイスの変更点については、第 6 章「Cisco Unified JTAPI 拡張」を参照してください。CTIPort および RoutePoint での AutoAccept のメッセージ フローを確認するには、付録 A「メッセージ シーケンスの図」を参照してください。

CiscoTermRegistrationFailed イベント

CiscoMediaTerminal または CiscoRouteTerminal の登録が非同期で失敗すると、アプリケーションにこのイベントが提供されます。登録が失敗すると、通常、アプリケーションは CiscoRegistrationFailedException を取得します。ただし、登録要求が正常に行われても、CTI によって登録が拒否される可能性もあります。このイベントは、登録要求が正常に行われても、登録が拒否された場合に提供されます。このイベントを受信するには、アプリケーションに TerminalObserver が必要です。このイベントを受信したときに提供されるエラー コードによっては、アプリケーションが新しいパラメータを使用して再登録する必要があります。

次に、返されるエラーと、解決するためにアプリケーションが行う必要のある処置の一覧を示します。

エラー メッセージ　CiscoTermRegistrationFailedEv.MEDIA_CAPABILITY_MISMATCH

説明　端末がすでに登録されているため、登録できない。同じメディア機能で再度登録を実行してください。

推奨処置　同じ機能で再登録してください。

エラー メッセージ　CiscoTermRegistrationFailedEv.MEDIA_ALREADY_TERMINATED_NONE

説明　端末がメディア終端タイプ「none」ですでに登録されているため、登録できない。

推奨処置　メディア終端タイプ「none」で再登録してください。

エラー メッセージ　CiscoTermRegistrationFailedEv.MEDIA_ALREADY_TERMINATED_STATIC

説明　端末が静的メディア終端ですでに登録されているため、登録できない。静的登録の場合、再度登録を実行できません。

推奨処置　端末が UnRegister 処理されるまで待機してください。

エラー メッセージ　CiscoTermRegistrationFailedEv.MEDIA_ALREADY_TERMINATED_DYNAMIC

説明　端末が動的メディア終端ですでに登録されているため、登録できない。

推奨処置　動的メディア終端で再登録してください。

エラー メッセージ　CiscoTermRegistrationFailedEv.OWNER_NOT_ALIVE

説明　端末を登録しようとすると、登録が競合状態になる。

推奨処置　端末を再登録してください。

このイベントには次のインターフェイスが定義されています。

int　　**getErrorCode**（）

この例外の errorCode を返します。

メッセージ フローの変更はありません。

SelectRoute インターフェイスの拡張

SelectRoute インターフェイスが拡張されて、PreferredOriginalCalledNumber パラメータと PreferredOriginalCalledOption パラメータを取るようになりました。これにより、コールがルーティングされるときに、OriginalCalled 値を指定された PreferredOriginalCalledNumber に再設定することが可能になりました。このインターフェイスは、PreferredOriginalCalledNumber、PreferredOriginalCalledOption のリストを取り、それらを RouteSelected リストに対応付けます。RouteSelected リストのインデックス I にある Route にコールがルーティングされる場合、インデックス I の PreferredOriginalCalledNumber と PreferredOriginalCalledOption が使用されます。パラメータのさまざまな値によって、アプリケーションは次のように動作します。

（注）　x の下で、コールがルーティングされるインデックスを選択してください。たとえば、コールが Route n へルーティングされる場合の x 値は n に等しくなります。インデックス x の PreferredOriginalCalledOption が無効または範囲外である場合は、JTAPI によってデフォルトの CiscoRouteSession.DONOT_RESET_ORIGINALCALLED に設定されます。また、PreferredOriginalCalledOption が null である場合、すべてのルーティングが CiscoRouteSession.DONOT_RESET_ORIGINALCALLED オプション付きで処理されます。

PreferredOriginalCalledOption[x] が CiscoRouteSession.RESET_ORIGINALCALLED に設定された場合

- RouteSelected リストに Route R1、R2 ..Rn が含まれる場合、preferredOriginalCalled リストに O1、O2、… On が含まれます。R1 が使用可能な場合、R1 へコールがルーティングされて、OriginalCalledNumber が O1 に設定されます。R1 がビジーで R2 が使用可能な場合、R2 へコールがルーティングされて、OriginalCalledNumber が O2 に設定され、以降同様に続きます。
- RouteSelected リストに Route R1、R2 ..Rn が含まれる場合、preferredOriginalCalled リストに O1、O2、… Om が含まれます（m < n）。R1 が使用可能な場合、R1 へコールがルーティングされて、preferredOriginalCalled が O1 に設定されます。R1 がビジーで R2 が使用可能な場合、R2 へコールがルーティングされて、OriginalCalledNumber が O2 に設定され、m になるまで、以降同様に続きます。Route m+1 からは、Rm+1 が使用可能な場合、Rm+1 へコールがルーティングされて、OriginalCalledNumber が Rm+1 に設定され、以降同様に続きます。最終的に、Rn が使用可能な場合、コールが Rn へルーティングされ、OriginalCalledNumber が Rn " に設定されます。
- RouteSelected リストに Route R1、R2 ..Rn で preferredOriginalCalled リストが NULL のとき、R1 が使用可能な場合、R1 へコールがルーティングされて、OriginalCalledNumber が R1 に設定されます。R1 がビジーで R2 が使用可能な場合、R2 へコールがルーティングされて、OriginalCalledNumber が R2 に設定され、以降同様に続きます。

PreferredOriginalCalledOption[x] が CiscoRouteSession.DONOT_RESET_ORIGINALCALLED に設定された場合

- RouteSelected リストに Route R1、R2 ..Rn が含まれる場合、preferredOriginalCalled リストに O1、O2、..On が含まれます。この場合、利用可能なルートの 1 つにコールがルーティングされ、OriginalCalledNumber は変更されません。
- RouteSelected リストに Route R1、R2 ..Rn が含まれる場合、preferredOriginalCalled リストに O1、O2、… Om が含まれます。m < n では、利用可能なルートの 1 つにコールがルーティングされ、OriginalCalledNumber は変更されません。

- RouteSelected リストに Route R1、R2 ..Rn が含まれ、preferredOriginalCalled リストが NULL である場合、利用可能なルートの 1 つにコールがルーティングされ、OriginalCalledNumber は変更されません。

（注）　OriginalCalled が PreferredOriginalCalled に設定されると、LastRedirectingParty 番号も PreferredOriginalCalled に再設定されます。

SelectRoute インターフェイスの拡張には、次の新しいインターフェイスまたは変更されたインターフェイスがあります。

int　**selectRoute** (java.lang.String[] routeSelected, int callingSearchSpace, java.lang.String[] preferredOriginalCalledNumber, int[] preferredOriginalCalledOption)

コールをルーティング可能な 1 つ以上の宛先を選択します。

PreferredOriginalCalledOption には次のいずれかの値を指定できます。

static int　**DONOT REESET_ORIGINALCALLED**

PreferredOriginalCalledOption のオプション パラメータ値。OriginalCalled を再設定しないことを指定します。

static int　**REESET_ORIGINALCALLED**

PreferredOriginalCalledOption のオプション パラメータ値。OriginalCalled を preferredOriginalCalledNumber に再設定します。

インターフェイスの変更点については、第 6 章「Cisco Unified JTAPI 拡張」を参照してください。SelectRoute インターフェイスの拡張のメッセージ フローを確認するには、付録 A「メッセージ シーケンスの図」を参照してください。

コールのプレゼンテーション インジケータ

コールのプレゼンテーション インジケータ（PI）は、エンド ユーザに対して Calling/Called/CurrentCalling/CurrentCalled/LastRedirecting の名前および番号を隠したり見せたりする機能をアプリケーションに提供します。JTAPI には、通話者の PI 値を取得する CiscoCall の関数があります。この PI 情報を使用して、通話者の情報をエンド ユーザに提示します。これらの関数は、true または false の値を返します。値が「true」であれば表示が「許可」状態であり、「false」であれば「制限」状態であることを示します。

電話会議では、CiscoCall のこのインターフェイスは正しい値を返しません。アプリケーションは、コール内のすべての Connection を繰り返して確認し、Connection の作成先アドレスに関連付けられた PI 値を取得する必要があります。CiscoConnection で提供されるインターフェイスは、getAddressPI( ) です。

CiscoCall の次の新しいインターフェイスは PI 値を取得します。

CiscoCall

boolean **getCalledAddressPI()**

getCalledAddressPI に関連付けられた PI を返します。true が返された場合、アプリケーションはアドレス名を表示します。false が返された場合、アプリケーションはアドレス名を表示しません。

boolean **getCallingAddressPI()**

getCallingAddressPI に関連付けられた PI を返します。true が返された場合、アプリケーションはアドレス名を表示します。false が返された場合、アプリケーションはアドレス名を表示しません。

boolean **getCurrentCalledAddressPI()**

CurrentCalledAddressPI に関連付けられた PI を返します。true が返された場合、アプリケーションはアドレス名を表示します。false が返された場合、アプリケーションはアドレス名を表示しません。

boolean **getCurrentCalledDisplayNamePI()**

CurrentCalledDisplayNamePI に関連付けられた PI を返します。true が返された場合、アプリケーションはアドレス名を表示します。false が返された場合、アプリケーションはアドレス名を表示しません。

boolean **getCurrentCallingAddressPI()**

getCurrentCallingAddressPI に関連付けられた PI を返します。true が返された場合、アプリケーションはアドレス名を表示します。false が返された場合、アプリケーションはアドレス名を表示しません。

boolean **getCurrentCallingDisplayNamePI()**

getCurrentCallingDisplayNamePI に関連付けられた PI を返します。true が返された場合、アプリケーションはアドレス名を表示します。false が返された場合、アプリケーションはアドレス名を表示しません。

boolean **getLastRedirectingAddressPI()**

getLastRedirectingAddressPI に関連付けられた PI を返します。true が返された場合、アプリケーションはアドレス名を表示します。false が返された場合、アプリケーションはアドレス名を表示しません。

CiscoConnection の次のインターフェイスは、Connection に関連付けられたアドレスの PI 値を取得します。

CiscoConnection

boolean **getAddressPI()**

Connection の作成先アドレスに関連付けられた PI を返します。true が返された場合、アプリケーションはアドレス名を表示します。false が返された場合、アプリケーションはアドレス名を表示しません。

メッセージ フローの変更はありません。

Progress 状態から Disconnect 状態への変換

ヨーロッパの PSTN で、API を通じて未割り当ての電話番号にコールが発信された場合、アプリケーションは、原因コードが CiscoCallEv.CAUSE_UNALLOCATEDNUMBER に設定された ConnFailedEv イベントを受信します。米国の PSTN の場合は、まったくイベントを受信しないことがあります。

欧州の PSTN を経由した場合と米国の PSTN を経由した場合の動作を一貫性のあるものとし、同時に下位互換性の問題にも対処するために、UseProgressAsDisconnectedDuringErrorEnabled という新しいサービス パラメータが JTAPI バージョン 1.4(3.21) で jtapi.ini ファイルに追加されています。このパラメータを有効（1 = 有効、0 = 無効、デフォルトは無効）にすると、両方の場合にアプリケーションが ConnFailedEv を受信します。

デバイス ステート サーバ

デバイス ステート サーバは、端末のすべてのアドレスの累積的な状態情報を提供します。これらのイベントは、端末イベントとして配信されます。これらのイベントを受信するには、TerminalObserver を追加する必要があります。

次の状態があります。

- IDLE：端末のどのアドレスにもコールが存在しない場合、その端末の DeviceState は IDLE と見なされ、Cisco Unified JTAPI からアプリケーションに `CiscoTermDeviceStateIdleEv` が送信されます。
- ACTIVE：端末のいずれかのアドレスに発信コール（CTI の状態が、Dialtone、Dialing、Proceeding、Ringback または Connected）または着信コール（CTI の状態が Connected）が存在する場合、その端末の DeviceState は ACTIVE と見なされ、Cisco Unified JTAPI からアプリケーションに `CiscoTermDeviceStateActiveEv` が送信されます。
- ALERTING：端末のアドレスに発信コール（CTI の状態が Dialtone、Dialing、Proceeding、Ringback、または Connected）または着信コール（CTI の状態が Connected）が存在し、その端末の 1 つ以上のアドレスに未応答の着信コール（CTI の状態が、Offering、Accepted、または Tinging）が存在する場合、その端末の DeviceState は ALERTING と見なされ、Cisco Unified JTAPI からアプリケーションに `CiscoTermDeviceStateAlertingEv` が送信されます。
- HELD：端末のいずれかのアドレスですべてのコールが保留中（CTI の状態が OnHold）の場合、その端末の DeviceState は HELD と見なされ、Cisco Unified JTAPI からアプリケーションに `CiscoTermDeviceStateHeldEv` が送信されます。

新規イベント

- CiscoTermDeviceDeviceStateIdleEv
- CiscoTermDeviceStateActiveEv
- CiscoTermDeviceStateAlertingEv
- CiscoTermDeviceStateHeldEv

新規インターフェイスと変更されたインターフェイス

`public int getDeviceState()` は、端末のデバイス状態を返します。

CiscoTermEvFilter の次の新規インターフェイスは、デバイス状態の設定と取得を行います。

void　**setDevideStateActiveEvFilter**(boolean filterValue)

CiscoTermDeviceStateActiveEv フィルタを有効または無効にします。デフォルトでは、無効に設定されています。

void　**setDeviceStateAlertingEvFilter**(boolean filterValue)

CiscoTermDeviceAlertingEv フィルタを有効または無効にします。デフォルトでは、無効に設定されています。

void　**setDeviceStateHeldEvFilter**(boolean filterValue)

CiscoTermDeviceHeldEv フィルタを有効または無効にします。デフォルトでは、無効に設定されています。

void　**setDeviceStateIdleEvFilter**(boolean filterValue)

CiscoTermDeviceIdleEv フィルタを有効または無効にします。デフォルトでは、無効に設定されています。

boolean　**getDeviceStateActiveEvFilter**()

CiscoTermDeviceStateActiveEv フィルタの状態を取得します。

boolean　**getDeviceStateAlertingEvFilter**()

CiscoTermDeviceStateAlertingEv フィルタの状態を取得します。

boolean　**getDeviceStateActiveEvFilter**()

CiscoTermDeviceStateAlertingEv フィルタの状態を取得します。

boolean　**getDeviceStateActiveEvFilter**()

CiscoTermDeviceStateAlertingEv フィルタの状態を取得します。

インターフェイスの変更に関する詳細については、第 6 章「Cisco Unified JTAPI 拡張」を参照してください。

Forced Authorization Code と Client Matter Code

Forced Authorization Codes（FAC）は、外線、市外通話、国際通話など、特定のクラスの着信番号（DN）にコールを発信する前に有効な認証コードの入力をユーザに要求します。認証情報は、Cisco Unified Communications Manager CDR データベースに記録されます。

Client Matter Codes（CMC; クライアント マター コード）は、コールを発信する前にユーザがコードを入力できるようにします。クライアント マター コードを使用すると、発信したコールに会計コードや請求コードを割り当てることができます。クライアント マター コードの情報は、Cisco Unified Communications Manager CDR データベースに記録されます。

サポートされるインターフェイス

Cisco Unified JTAPI では、次のインターフェイスで FAC と CMC をサポートしています。

- Call.Connect()
- Call.Consult()
- Call.Transfer(String)
- Connection.redirect()
- RouteSession.selectRoute()

Call.Connect() と Call.Consult()

アプリケーションがこれらのインターフェイスのいずれかを使用して、FAC、CMC、またはその両方を必要とする DN にコールを発信すると、CiscoToneChangedEv が CallObserver に配信されます。このイベントには、その DN でどちらのコードが必要とされているかについての情報も格納されています。このイベントが FAC_CMC 機能によって配信された場合でも、getCiscoCause() インターフェイスは CiscoCallEV.CAUSE_FAC_CMC を返します。getTone() インターフェイスは、ZIPZIP トーンが再生されていることを示す CiscoTone.ZIPZIP を返します。

CiscoToneChangedEv を受信した後、アプリケーションは connection.addToAddress インターフェイスを使用して、適切なコード（末尾に # を付ける）を入力する必要があります。コードの数字と末尾の # は、T302 タイマーで指定されている文字間タイマー値内に 1 文字ずつ入力することも、T302 タイマー値内に全文字を入力することもできます。T302 タイマーの値は、Cisco Unified Communications Manager Administration で設定します。

FAC と CMC の両方が必要な場合

両方のコードを必要とする DN の場合、最初のイベントは常に FAC に適用され、2 番目のイベントは CMC に適用されます。ただし、2 つのコードをシャープ記号（#）で区切ることにより、両方のコードを同じ要求で送信することもできます。最初の要求で送信されるコードによっては、2 番目のイベントが省略されます。

アプリケーションは両方のコードを同時に送信できますが、次の例に示すように、両方のコードの最後に # を付ける必要があります。

```
connection.addToAddress("1234#678#")
```

ここで、1234 は FAC を表し、678 は CMC を示します。

この場合、アプリケーションは 2 番目の CiscoToneChanged を受信しません。

最初の CiscoToneChangesEv に、getWhichCodeRequired()=CiscoToneChanged.FAC_CMC_REQUIRED と getCause()=CiscoCallEv.CAUSE_FAC_CMC の両方が含まれます。

この場合、次のようなケースが考えられます。

- アプリケーションが FAC と CMC を同じ connection.addToAddress(code1#code2#) 要求で送信する。この場合、アプリケーションは 2 番目の CiscoToneChangedEv を受信しません。
- アプリケーションが最初の connection.addToAddress(code#1) 要求で FAC だけを送信する。この場合、アプリケーションは 2 番目の CiscoToneChangedEv で getWhichCodeRequired()=CiscoToneChangedEv.CMC_REQUIRED を受信します。
- アプリケーションが最初のコードの一部だけ、または最初のコードの全部と 2 番目のコードの一部だけを送信する（末尾に # が付いていない場合、そのコードは不完全と見なされ、T302 タイマーの時間が経過すると、システムがコードの検証を試みます）。コードが不完全な場合、アプリケーションは 2 番目の CiscoToneChangedEv で getWhichCodeRequired()=CiscoToneChangedEv.CMC_REQUIRED および getCause()=CiscoCallEv.CAUSE_FAC_CMC を受信します。

PostCondition タイマー

PostCondition タイマーは、connection.addToAddress インターフェイスを呼び出してコードを送信するたびにリセットされます。FAC および CMC の末尾には # を付ける必要があります（たとえば、Connection.addToAddress(「1234#」) のように入力します。この場合、1234 が FAC と見なされます）。すべてのコードが入力されている場合は、T302 タイマーの時間が経過すると、コールが発信されます。すべてのコードが入力されていない場合は、リオーダー音が再生されます。このケースでは、コールが発信された場合でも、アプリケーションが postConditionTimeout を含む PlatformException を受信することがあります。これを回避するには、JTAPI Preferences を使用して、PostCondition のタイムアウト値を大きくする必要があります。

アプリケーションが call.connect() または call.consult() を使用してコールを開始して、PostCondition タイムアウトの制限時間内に Cisco Unified IP Phone から FAC または CMC（# を含む）が入力されなかった場合、実際にはコールが発信されても、postCondition タイムアウトを含む platformException を受信することがあります。これを回避するには、JTAPI Preferences を使用して、PostCondition のタイムアウト値を大きくする必要があります。

共用回線

開始側の通話者が共用回線を使用している場合は、アプリケーションが connection.addToAddress インターフェイスを使用して追加の番号を渡す前に、setRequestController を使用してアクティブな terminalConnection を設定する必要があります。

無効または欠落したコード

T302 タイマーの有効期限が超過する前に入力されたコードが無効な場合またはコードが入力されなかった場合は、コールが拒否されて callCtlCause 原因コードに CiscoCallEv.CAUSE_FAC-CMC が設定されます。

Call.transfer(String) と Connection.redirect()

FAC と CMC をサポートするために、これらのインターフェイスに 2 つの文字列パラメータ（facCode と cmcCode）が追加されました。これらのコードのデフォルト値は、null 値を表します。

コードを必要とする DN に対してこれらの要求が発行された場合、CiscoToneChangedEv は配信されません。FAC、CMC、またはその両方を必要とする DN へ条件的にリダイレクトされるコールは、いずれかのコードが正しくない場合でも拒否されず、接続が維持されます。

RouteSession.selectRoute()

2 つの追加の文字列パラメータの配列（facCode と cmcCode）が FAC と CMC をサポートしています。各 routeselect 要素に DN のコードを指定できます。コードが不要な DN に対しては null 値を指定する必要があります。これらのコードのデフォルト値は、null 値です。

最初の routeselected 要素に適切なコードが含まれていない場合は、配列に含まれる次の要素が試されます。すべての要素が正しくない場合は、reRouteEvent がアプリケーションに送信されます。

（注） FAC または CMC コードを必要とする DN への転送（Call Forward）はサポートされていません。Address API を使用して転送番号をこのような DN に設定することはできますが、このような番号へのコール転送は拒否されます。

スーパー プロバイダー（デバイス認証の無効）

通常、システム管理者は JTAPI アプリケーション ユーザを設定する際に、特定の端末（Cisco Unified IP Phone およびデバイス）を各アプリケーション ユーザに関連付けます。そのため、アプリケーション ユーザが制御または監視できる端末は、自分に関連付けられている端末に限定されます。スーパープロバイダー機能を使用すると、Cisco Unified Communications Manager クラスタ内のすべての端末をアプリケーションで制御したり監視することができます。

CiscoProvider の新しいインターフェイスである createTerminal() を使用すると、terminalName を指定して端末を作成できます。JTAPI には、インターフェイスを通じて terminalName を取得する機能が用意されていません。CiscoProvider.createTerminal(terminalName) は、端末を返します。端末がプロバイダー ドメインにすでに存在している場合、JTAPI はその既存の端末を返します。

2 つ目の新しいインターフェイスである CiscoProvider.deleteTerminal() を使用すると、CiscoProvider.createTerminal() インターフェイスで作成された CiscoTerminal オブジェクトを削除できます。端末オブジェクトが存在しない場合や、端末が CiscoProvider.createTerminal() インターフェイスで作成されたものでない場合は、JTAPI が例外をスローします。

また、JTAPI では、CiscoProviderCapabilities の新しいインターフェイスとして canObserveAnyTerminal() も提供されています。アプリケーション ユーザがこのインターフェイスを使用できるようにするには、Cisco Unified Communications Manager Administration のユーザ設定を変更します。アプリケーションはこのインターフェイスを使用して、自身に createTerminal(terminalName) インターフェイスを呼び出す能力があるかどうかを判別できます。このインターフェイスを呼び出す能力がないアプリケーションがこのインターフェイスを呼び出した場合は、JTAPI から PrivilegeViolationException がスローされます。Cisco Unified Communications Manager クラスタ内に存在しない terminalName を指定した場合は、JTAPI から InvalidArgumentException がスローされます。

Q.Signaling（QSIG）パス交換

QSIG パス交換は、QSIG トランクを介して接続されている PBX 間でコールが転送されるときに、リアルタイム プロトコル（RTP）パスを最適化するネットワーク機能です。パス交換の実行中は、アプリケーションからの機能要求が無視されて JTAPI からアプリケーションに例外がスローされる短い期間が発生します。

RTP パスが始端および終端の PBX をつなぐダイレクト パスに最適化されると、コール の Global Call ID が変更されます。JTAPI には、このコールをモニタリングするための新しいインターフェイスが用意されています。

ネットワーク イベント

以前のリリースの Cisco Unified JTAPI では、クラスタ外のアドレスへのコールを開始すると、遠端のアドレスに CallCtlConnNetworkReachedEv イベントと CallCtlConnNetworkAlertingEv イベントが配信されていました。

Cisco Unified Communications Manager 4.0 以降では、これらのイベントは配信されません。これらのバージョンでは、遠端アドレスの CallCtlConnection が OFFERED 状態から ESTABLISHED 状態に移行します。アプリケーションは、遠端アドレスの CallCtlConnOfferedEv および CallCtlConnEstablishedEv を受信します。CallCtlConnNetworkReachedEv および CallCtlConnNetworkAlertingEv イベントはアプリケーションに配信されません。ネットワーク イベントを受信するには、ゲートウェイに対して設定されているルート パターンの「オーバーラップ送信を許可」フラグをオンにする必要があります。

これらのイベントの配信をアプリケーションで制御できるようにするために、jtapi.ini の新しいパラメータ AllowNetworkEventsAfterOffered が導入されています。[オーバーラップ送信を許可] フラグをオンに設定できないアプリケーションでは、この jtapi.ini パラメータを使用して発信コールのネットワーク イベントを受信できます。

このパラメータをオンにするには次の手順を実行します。

ステップ 1　jtprefs を実行して、適切なオプションを選択します。Cisco Unified JTAPI がデフォルト ディレクトリにインストールされている場合は、c:¥winnt¥java¥lib に jtapi.ini ファイルが作成されます。jtapi.ini ファイルがすでに存在する場合は、jtprefs を実行しなくてもこのファイルを直接編集できます。

ステップ 2　Add AllowNetworkEventsAfterOffered=1 をファイルの最後に追加して、ファイルを保存します。

ステップ 3　Cisco Unified JTAPI を再インストールするたびに上記の手順を繰り返します。

AllowNetworkEventsAfterOffered フラグが有効になると、アプリケーションは遠端アドレスに対して、CallCtlConnOfferedEv、CallCtlConnNetworkReachedEv または CallCtlConnNetworkAlertingEv、および CallCtlConnEstablishedEv を受信します。