Apple Push Notificationサービスプログラミングガイド

iPhone > User Experience

2009-05-22

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
U.S.A.

アップルジャパン株式会社
〒163-1450 東京都新宿区西新宿
3 丁目20 番2 号
東京オペラシティタワー
http://www.apple.com/jp/

目次

序章 はじめに 7

この書類の構成 7
関連項目 8

第 1 章 Push Notificationとは 9

ユーザの視点から見たPush Notification 9
デベロッパの視点から見たPush Notification 11

第 2 章 Apple Push Notificationサービス 13

Push Notificationとその経路 13
フィードバックサービス 14
Quality of Service 14
セキュリティアーキテクチャ 15
　サービスーデバイス間の接続信頼 15
　プロバイダーサービス間の接続信頼 16
　トークンの生成と共有 17
　トークンによる信頼（通知） 18
　信頼に関連するコンポーネント 19
通知ペイロード 20
　ローカライズ形式の文字列 22
　JSONペイロードの例 23

第 3 章 プロビジョニングおよび開発 25

サンドボックス環境と製品環境 25
プロビジョニングの手順 25
　SSL証明書と鍵の作成 26
　プロビジョニングプロファイルの作成とインストール 27
　SSL証明書と鍵のサーバへのインストール 28

第 4 章 プロバイダとApple Push Notificationサービスの通信 29

プロバイダの一般的な要件 29
低レベルのインターフェイス 30
フィードバックサービス 31

第 5 章 iPhone OSクライアントの実装 33

カスタム警告音の準備 33

Remote Notificationのための登録　34
プロバイダに現在の言語設定を渡す　35
Remote Notificationの処理　36

改訂履歴　書類の改訂履歴　39

図、表、リスト

第 1 章 **Push Notificationとは　9**

図 1-1　Push Notificationの警告　9
図 1-2　数字付きバッジが表示されたアプリケーションアイコン　10
図 1-3　「View」ボタンが非表示にされた通知警告メッセージ　10

第 2 章 **Apple Push Notificationサービス　13**

図 2-1　プロバイダからクライアントアプリケーションへのPush Notification　14
図 2-2　複数のプロバイダから複数のデバイスへのPush Notification　14
図 2-3　デバイストークンの共有　18
表 2-1　`aps`辞書のキーと値　21
表 2-2　`alert`プロパティの子プロパティ　21

第 4 章 **プロバイダとApple Push Notificationサービスの通信　29**

図 4-1　Push Notificationのバイナリ形式　30
図 4-2　フィードバックタプルのバイナリ形式　32
リスト 4-1　低レベルインターフェイスを使用した通知の送信　30

第 5 章 **iPhone OSクライアントの実装　33**

リスト 5-1　Remote Notificationのための登録　35
リスト 5-2　現在サポートされている言語を取得してプロバイダに送信する　36
リスト 5-3　プロバイダからのデータのダウンロード　36

はじめに

Push Notification（Remote Notificationとも呼ばれる）は、iPhoneおよびiPod touchデバイス上のアプリケーションの更新データがサーバ上にあることをユーザに知らせる機能です。この機能では永続的なIP接続を使用します。デバイスがアプリケーションへの通知を受信したときにそのアプリケーションが実行されていない場合は、警告メッセージ、特徴的な警告音、またはアプリケーションのアイコン上に数字付きのバッジを付けること（または、これらのいずれかの組み合わせ）によって、デバイスはそれをユーザに通知します。その後ユーザがそのアプリケーションを起動すると、アプリケーションはサーバ（プロバイダとも呼ばれる）からデータをダウンロードします。Push NotificationはiPhone OS 3.0で導入された機能で、デスクトップコンピュータ上のバックグラウンドアプリケーションと同じ働きをします（ご承知のとおり、バックグラウンドアプリケーションはこれらの携帯用デバイスでは利用できません）。

注： Push Notificationを使用する際のガイドラインについては、『*iPhone Human Interface Guidelines*』の「Enabling Push Notifications」を参照してください。

この文書では、Push Notificationとは何かについて解説し、この機能をアプリケーションに実装する方法を説明します。クライアント側の実装には、Objective-CおよびCocoa Touchフレームワークに精通していることが前提になります。プロバイダ側の実装には、TLS/SSLおよびストリーミングソケットに関する知識が役立ちます。

この書類の構成

この文書は、次の各章で構成されています。

- 「Push Notificationとは」（9 ページ）では、Push Notificationとは何か、どのようにユーザに提供されるか、また、デベロッパにどのようなメリットと要件をもたらすかについて説明します。
- 「Apple Push Notificationサービス」（13 ページ）では、プロバイダからクライアントアプリケーションに通知を送信したりルーティングする際の中心となるプッシュサービスについて説明します。
- 「プロビジョニングおよび開発」（25 ページ）では、iPhoneデベロッパプログラムのポータルから証明書を取得して開発環境をセットアップするための手順を説明します。
- 「プロバイダとApple Push Notificationサービスの通信」（29 ページ）では、プロバイダとApple Push Notificationサービス間の通信に必要な要件およびインターフェイスについて説明します。
- 「iPhone OSクライアントの実装」（33 ページ）では、Push (Remote) Notificationを登録したり処理するためにiPhone OSシステム上のクライアントアプリケーションが実行しなければならないことについて説明します。

関連項目

次の情報は、Push Notificationの理解および実装の参考になります。

- 『*Security Overview*』では、iPhone OSおよびMac OS Xの両プラットフォームで使われているセキュリティのテクノロジーと手法について説明しています。
- `UIApplication`および`UIApplicationDelegate`のリファレンス文書では、クライアントアプリケーション用のRemote-Notification APIについて説明しています。
- RFC 5246はTLSプロトコルの標準です。

  データプロバイダとApple Push Notificationサービス間のセキュアな通信には、Transport Layer Security (TLS)、またはその前身のSecure Sockets Layer (SSL)に関する知識が必要です。詳細については、これらの暗号プロトコルのオンラインまたは印刷版解説書を参照してください。

Push Notificationとは

iPhoneまたはiPod touch上のアプリケーションは、多くの場合、クライアントサーバモデルに基づくより大きなアプリケーションの一部に過ぎません。クライアント側のアプリケーションはデバイスにインストールされ、サーバ側のアプリケーションはたくさんのクライアントアプリケーションにデータを提供することが主な役割です（このため、サーバ側のアプリケーションは「プロバイダ」と呼ばれます）。クライアントアプリケーションは時々プロバイダに接続して、そのアプリケーション用の更新データをダウンロードします。電子メールアプリケーションやソーシャルネットワーキングアプリケーションは、このようなクライアントサーバモデルの例です。

しかし、ダウンロード可能な更新データがプロバイダにあるときに、アプリケーションがプロバイダに接続されていなかったり、デバイス上で実行すらされていない場合はどうなるでしょうか？アプリケーションはどのようにして更新データのことを知るのでしょうか？Push Notificationがこの悩みを解決します。Push Notificationはプロバイダがデバイスに配信した短いメッセージです。デバイスはこれを受信すると、ダウンロード可能なデータが存在することをクライアントアプリケーションのユーザに通知します。ユーザがこの機能を有効にしていて、アプリケーションが正しく登録されていれば、この通知はデバイス（アプリケーションの場合もある）に配信されます。Apple Push Notificationサービス（「Apple Push Notificationサービス」 （13 ページ）で説明）は、このRemote Notification機能のための基本テクノロジーです。

ユーザの視点から見たPush Notification

iPhoneを使用している様子（電話をかける、インターネットサーフィンをする、音楽を聴く）を想像してください。iPhoneにはチェスのアプリケーションがインストールされており、あなたは遠隔地でプレイしている友人とゲームを始めることにします。あなたは先手を打ちます（これは、ゲームのサーバアプリケーションによって正しく認識されます）。次に、電子メールを読むためにこのクライアントアプリケーションを終了します。その間に、友人があなたの手に反撃します。チェスアプリケーションのサーバはこの手を認識した後、あなたのデバイス上のチェスアプリケーションが接続されていないことを検知して、Apple Push Notificationサービス(APNs)にPush Notificationを送信します。

あなたのデバイス（正確には、デバイス上で実行中のiPhone OS）は、APNsからこの通知を受信します。チェスアプリケーションは現在実行されていないため、iPhone OSは図 1-1に示すような警告を表示します。このメッセージは、アプリケーション名、短いメッセージ、および（この場合は）2つのボタン（「Close」と「View」）で構成されています。

図 1-1 Push Notificationの警告

「View」ボタンをタップすると、チェスアプリケーションが起動してプロバイダに接続し、新しいデータをダウンロードします。そして、チェス盤のユーザインターフェイスを調整して友人の手を表示します（「Close」をタップすると、単にこの警告が消えます）。

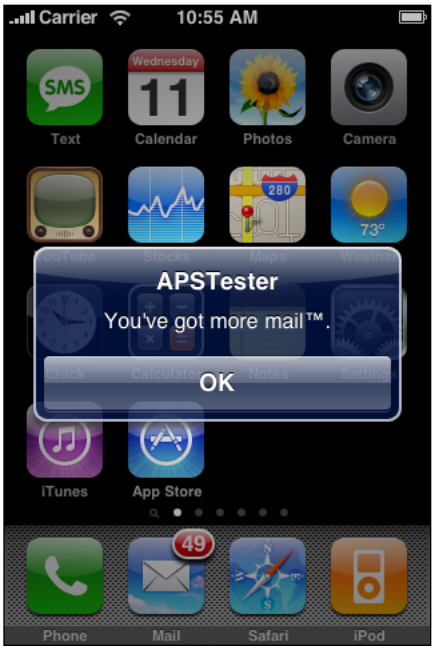
警告メッセージを表示する代わりに（または、警告メッセージに加えて）、iPhone OSはターゲットアプリケーションのアイコンの右上隅に数字付きバッジを表示することもできます。その様子を図1-2に示します。バッジの数字は、通常、アプリケーションのためにサーバに用意されている項目の数を表します。ただし、別の意味を表す場合もあります。バッジの数字はアプリケーション固有のもので、どんな事柄の数を表してもかまいません（たとえば、ダウンロード可能なデータ項目の数、未読の電子メールメッセージ（ただし、メッセージはダウンロード済み）の数など）。

図 1-2　数字付きバッジが表示されたアプリケーションアイコン

警告メッセージや数字付きバッジと一緒に、iPhone OSは、通知を受信したことをユーザに知らせる短い特徴的な警告音を鳴らすこともできます。

クライアントアプリケーションのユーザに警告を行う方法のほとんどはプロバイダが制御します。プロバイダは、警告メッセージ、警告音、数字付きバッジのどの組み合わせをデバイスで実行するかを決定します（ただし、ユーザが制御できる部分もあります）。また、カスタム警告音を指定したり、バッジの数字をインクリメント、デクリメント、または削除することもできます。警告メッセージを指定する場合は、メッセージテキストを作成して、2つではなく1つのボタンを指定することもできます。ボタンを1つにした場合は、図1-3に示すように、「View」ボタンが表示されなくなります。

図 1-3　「View」ボタンが非表示にされた通知警告メッセージ



アプリケーションが実行されているかどうかに関わらず、プロバイダはiPhoneアプリケーションにPush Notificationを送信します。通知が届いたときにアプリケーションが実行されている場合は、アプリケーションデリゲートがその通知を直接処理することもできます。APNsは、デバイス上のアプリケーション宛てにプロバイダから受信した最後の通知を保持しています。したがって、デバイスがオンラインになったときに、その通知をまだ受信していない場合、APNsは保存されている通知をデバイスに配信します。デバイスは、Wi-Fi接続でも携帯電話接続でもPush Notificationを受信します。

重要： Wi-Fi接続がPush Notificationに使用されるのは、携帯電話接続が存在しない場合、またはデバイスがiPod touchの場合だけです。Wi-Fi経由で通知を受信するには、デバイスのディスプレイがオンになっている（つまり、スリープ状態でない）か、デバイスがプラグに接続されていなければなりません。

iPhoneおよびiPod touchデバイスのユーザは、デバイス、またはデバイスにインストールされているアプリケーションがPush Notificationを受信するかどうかを制御できます。アプリケーションの場合は、通知の種類（アイコンのバッジ、警告メッセージ、警告音）を選択的に有効または無効にできます。これらの制限は、「設定(Settings)」アプリケーションの「通知(Notifications)」環境設定で設定します。UIKitフレームワークでは、特定のアプリケーションのユーザ環境設定を検出するプログラミングインターフェイスを提供しています。

デベロッパの視点から見たPush Notification

Push Notificationは、デスクトップシステム上のバックグラウンドアプリケーションと同じ働きをします。現在実行されていないiPhoneアプリケーションに対しては、ユーザが介在することによって間接的に通知が行われます。オペレーティングシステムがアプリケーションの代わりにPush Notificationを受信し、「ユーザの視点から見たPush Notification」 （9 ページ）で説明した方法を1つ以上利用してユーザに警告を出します。警告を受け、ユーザがアプリケーションの起動を選択すると、アプリケーションはプロバイダに用意されているデータをダウンロードします。通知を受信したときにiPhoneアプリケーションが実行されている場合は、選択すればその通知を直接処理することもできます。

注： デスクトップシステムでは、通常、現在実行されていないアプリケーションにダウンロード可能なデータがあることをユーザに通知する手段としてバックグラウンド処理が使われます。しかしiPhoneなどのデバイスでは、パフォーマンスやセキュリティの理由から、バックグラウンドアプリケーションが禁止されています。1度に1つのアプリケーションしか実行できません。

Apple Push Notificationサービス(APNs)は、その名のとおり、デバイスに通知を配信するために「プッシュ型」の設計を使用しています。プッシュ型設計がその反対のプル型設計と異なるのは、通知の受け手（この場合はiPhone OS）が、能動的に更新情報をポーリングするのではなく、受動的に更新情報を検知する点です。プッシュ型設計によって、情報を広範囲かつタイムリーに配信することが可能になり、プル型設計につきもののスケーラビリティの問題もほとんどありません。APNsでは、Push Notificationを実現するために永続的なIP接続を使用します。

Push Notificationのほとんどは、ペイロード（ユーザへの通知方法を指定するAPNs定義のプロパティを含むプロパティリスト）で構成されています。パフォーマンスの理由から、ペイロードは意図的に小さくしています。ペイロードにカスタムプロパティを定義することもできますが、Push Notificationによる配信は保証されていないため、データ送信の目的でRemote Notificationメカニズムを使用してはいけません。ペイロードの詳細については「通知ペイロード」 （20 ページ）を参照してください。

アプリケーションにRemote Notification機能を追加するには、iPhoneデベロッパプログラムのポータルから適切な証明書を取得し、クライアント側とプロバイダ側の各アプリケーションに必要なコードを記述する必要があります。「プロビジョニングおよび開発」（25 ページ）では、プロビジョニングとセットアップの手順について説明します。また、「プロバイダとApple Push Notificationサービスの通信」（29 ページ）および「iPhone OSクライアントの実装」（33 ページ）では、実装の詳細について説明します。

Apple Push Notificationサービスは、不正な動作がないかどうか継続的にプロバイダを監視し、急な活動増加、すばやい接続／切断の繰り返し、およびそれに類似の活動を探します。このような動作を検出した場合、Appleはプロバイダへの通知を試みます。それでも、その動作が続く場合は、そのプロバイダの証明書を失効リストに登録し、それ以降の接続を拒否します。不正または問題となる動作が続くと、プロバイダのAPNsへのアクセスが強制終了される場合もあります。

Apple Push Notificationサービス

Apple Push Notificationサービス（略してAPNs）はPush Notification機能の中核です。これは、iPhoneやiPod touchなどのデバイスに情報を配信するための堅牢で非常に効率的なサービスです。各デバイスは、このサービスとの間で認証済みの暗号化されたIP接続を確立し、この永続的な接続を介して通知を受信します。アプリケーションが実行されていないときに通知が届いた場合、デバイスはそのアプリケーションに更新データがあることをユーザに警告します。

ソフトウェアのデベロッパ（「プロバイダ」）は、サーバソフトウェア内で通知を作成します。プロバイダは、対象となるクライアントアプリケーション宛ての受信データを監視している間、永続的でセキュアなチャネルを介してAPNsに接続します。アプリケーション宛ての新しいデータを受信すると、プロバイダは通知を作成してこのチャネルを介してAPNsに送信します。APNsは、その通知をターゲットデバイスに配信（プッシュ）します。

APNsは単純でありながら効率的で高い処理能力を持つ配信サービスです。さらに、APNsには蓄積交換機能を提供するデフォルトのQuality-of-Serviceコンポーネントが含まれています。詳細については、「Quality of Service」（14 ページ）を参照してください。

「プロバイダとApple Push Notificationサービスの通信」（29 ページ）および「iPhone OSクライアントの実装」（33 ページ）では、プロバイダとiPhoneアプリケーションに固有の実装要件について、それぞれ説明しています。

Push Notificationとその経路

Apple Push Notificationサービスは、特定のプロバイダから特定のデバイスに通知をルーティングします。通知は、デバイストークンとペイロードという2つの主要なデータ部分から構成された短いメッセージです。デバイストークンは電話番号にたとえることができます。そこには、クライアントアプリケーションがインストールされているデバイスをAPNsが見つけるために必要な情報が含まれています。また、APNsはこれを使用して通知のルーティングを認証します。ペイロードは、デバイス上のアプリケーションのユーザに警告する方法を指定するための、JSONで定義されたプロパティリストです。

注： デバイストークンの詳細については、「セキュリティアーキテクチャ」（15 ページ）を参照してください。通知ペイロードの詳細については、「通知ペイロード」（20 ページ）を参照してください。

Remote Notificationのデータの流れは単方向です。プロバイダは、クライアントアプリケーションのデバイストークンとペイロードを含む通知パッケージを作成します。プロバイダは、その通知をAPNsに送信します。そして、APNsがその通知をデバイスに配信（プッシュ）します。

APNsへの認証に成功すると、プロバイダはデータの提供先となるアプリケーションを識別するトピックをAPNsに提供します。このトピックは、現在のところiPhone OSデバイス上のターゲットアプリケーションのバンドル識別子です。

図 2-1 プロバイダからクライアントアプリケーションへのPush Notification

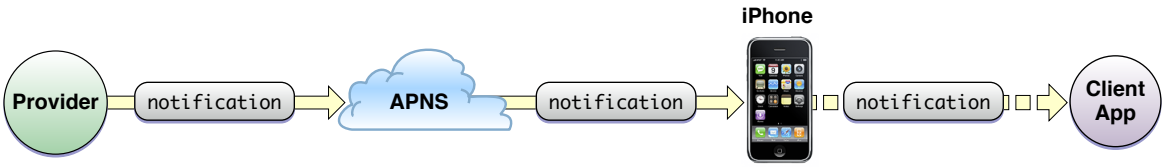
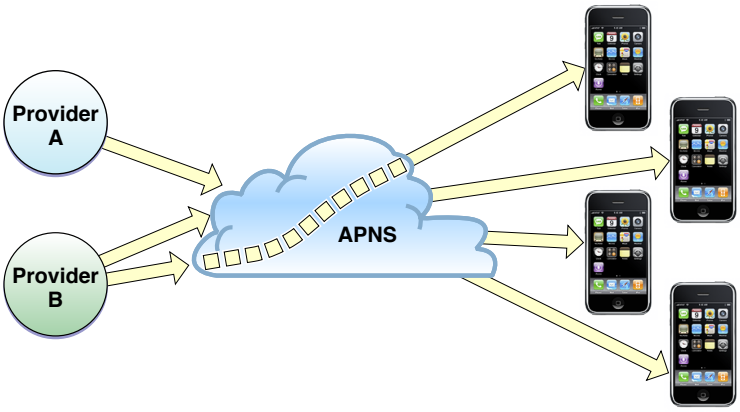


図 2-1は、APNsによってプロバイダとデバイスの間に実現される仮想ネットワークを非常に単純化した図です。ASPsのデバイス側とプロバイダ側には、共に複数の接続ポイントがあります。そして、プロバイダ側の接続ポイントをゲートウェイと呼びます。通常は、複数のプロバイダが存在し、それぞれがこれらのゲートウェイを介して、APNsとの間に1つ以上の永続的でセキュアな接続を確立します。そして、これらのプロバイダは、APNsを経由してクライアントアプリケーションがインストールされている複数のデバイスに通知を送信します。図 2-2は、上よりも少し現実的な図です。

図 2-2 複数のプロバイダから複数のデバイスへのPush Notification



フィードバックサービス

時には、APNsがデバイス上のアプリケーションに通知を配信しようとしても、ターゲットアプリケーションが存在しないためにデバイスに何度も配信を拒否されることがあります。このような状況は、ユーザがそのアプリケーションをアンインストールしている場合にしばしば発生します。このような場合、APNsは、プロバイダが接続しているフィードバックサービスを通してプロバイダに知らせます。フィードバックサービスは、アプリケーションごとに、最近通知の配信に何度も失敗しているデバイスのリストを管理しています。プロバイダは、このデバイスリストを取得してそのデバイスへの通知の送信を停止する必要があります。フィードバックサービスの詳細については「フィードバックサービス」 （31 ページ）を参照してください。

Quality of Service

Apple Push Notificationサービスには、蓄積交換機能を実行するデフォルトのQuality of Service (QoS)コンポーネントが含まれています。APNsが通知を配信しようとしたときにデバイスがオフラインだった場合は、QoSが通知を保存します。QoSが保持するのは、デバイス上のアプリケーションごとに1

つの通知（そのアプリケーション用にプロバイダから受信した最後の通知）だけです。オフラインのデバイスが後で再接続してきたときに、QoSは保存していた通知をデバイスに転送します。QoSは一定期間通知を保存した後、その通知を削除します。

セキュリティアーキテクチャ

プロバイダとデバイス間の通信を可能にするには、Apple Push Notificationサービスが、それらに対して特定のエントリポイントを公開しなければなりません。ただしセキュリティを保証するために、これらのエントリポイントへのアクセスを規制する必要もあります。この目的のためにAPNsは、プロバイダ、デバイス、およびそれらの通信に対して、2つの異なる信頼レベルを要求します。これらは、接続信頼とトークンによる信頼と呼ばれます。

接続信頼は、プロバイダ側のAPNs接続が、通知を配信することをAppleから許可されている認定済みのプロバイダとの接続であることを保証します。デバイス側の接続では、APNsはその接続が正当なデバイスとの接続であることを検証しなければなりません。

エントリポイントでの信頼を確立したら、APNsは正当なエンドポイントだけに通知を配信することを保証しなければなりません。それには、この接続を通してメッセージが送信されるルートを検証して、意図した通知先であるデバイスだけが通知を受信するようにしなければなりません。

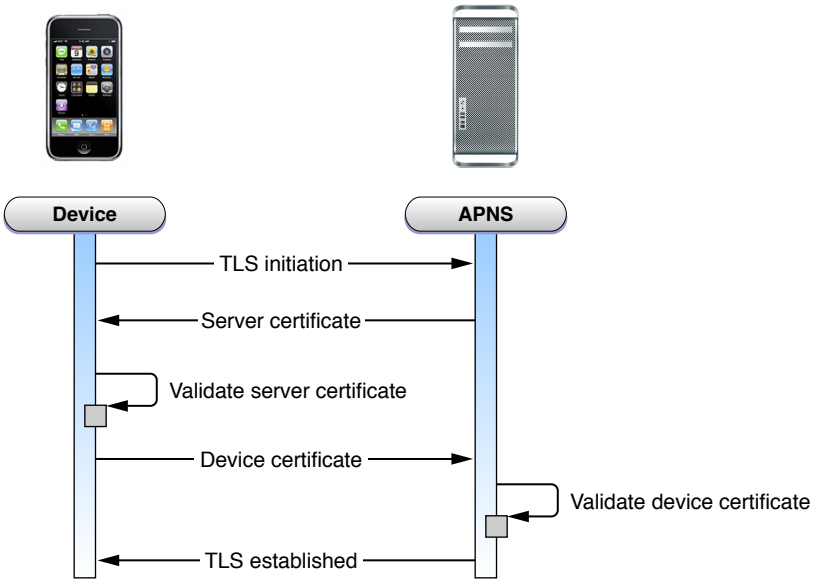
APNsでは、デバイストークンを利用して正確なメッセージルーティングを保証すること（つまり、**トークンによる信頼**）が可能になっています。デバイストークンは、APNsが初めてデバイスに接続したときに、APNsからそのデバイスに渡される不透過なデバイス識別子です。デバイスは、このデバイストークンをプロバイダと共有します。その後は、このトークンがプロバイダからのすべての通知に添付されます。これが、特定の通知のルーティングが正当であることを保証するための基礎になります（たとえて言えば、デバイストークンは、通信先を識別するための電話番号と同じ機能を果たします）。

注： デバイストークンは、`UIDevice`の`uniqueIdentifier`プロパティによって返されるデバイスのUDIDとは別物です。

この後のセクションでは、接続信頼とトークンによる信頼に必要なコンポーネントと、信頼を確立するための4つの手続きについて説明します。

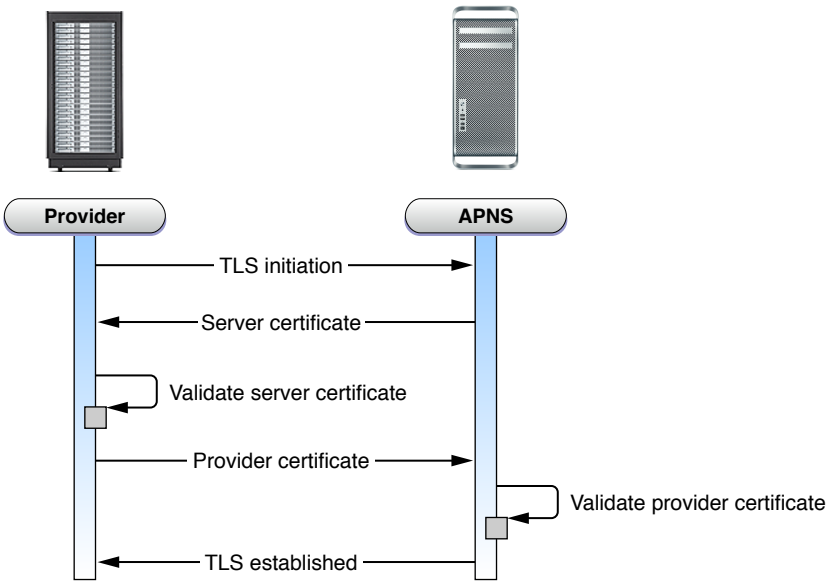
サービスーデバイス間の接続信頼

APNsは、TLSのピアツーピア認証を利用して、接続してきたデバイスの認証を行います（接続信頼のこのステージはiPhone OSによって処理されるので、デベロッパが自身で実装する必要はありません）。この手続きの中で、デバイスはAPNsとのTLS接続を開始し、APNsはサーバ証明書を返します。デバイスはこのサーバ証明書を検証して、APNsにデバイス証明書を送信します。APNsはそのデバイス証明書を検証します。



プロバイダーサービス間の接続信頼

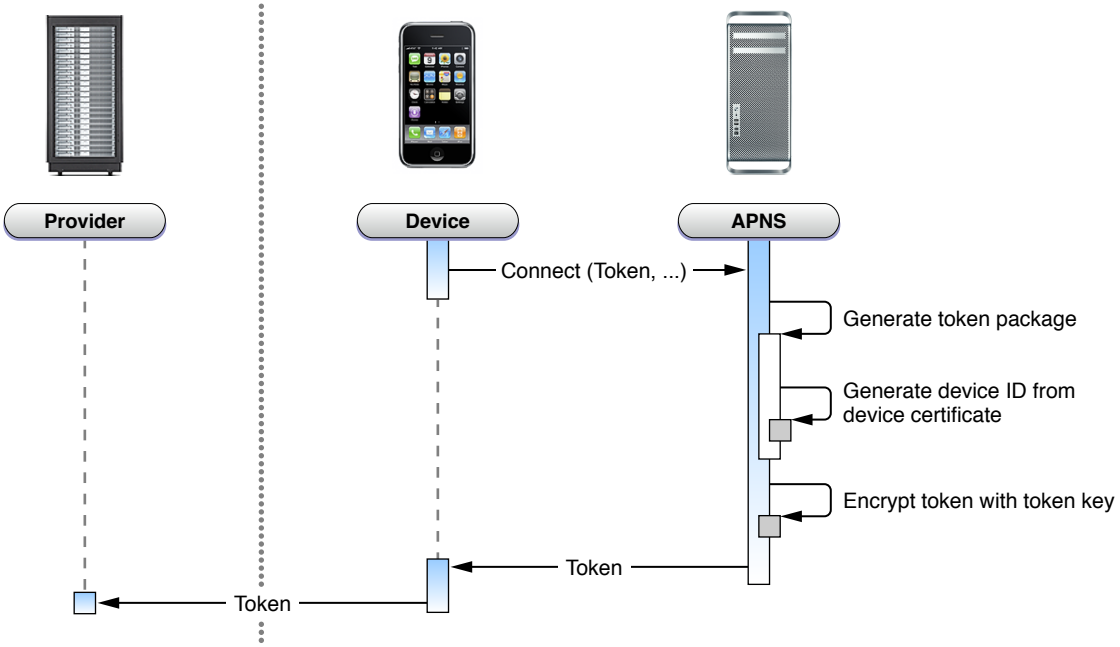
プロバイダとAPNs間の接続信頼も、TLSのピアツーピア認証を利用して確立されます。手続きも「サービスーデバイス間の接続信頼」（15 ページ）で説明したものと同様です。プロバイダはTLS接続を開始し、APNsからサーバ証明書を取得します。そしてそのサーバ証明書を検証します。次に、プロバイダはAPNsにプロバイダ証明書を送信します。今度は、APNsがそのプロバイダ証明書を検証します。この手続きが完了したら、セキュアなTLS接続が確立されます。これでAPNsは、この接続が正当なプロバイダによって確立したことを保証できます。



このプロバイダ接続は、証明書に規定されたトピック（バンドルID）によって識別される、ある特定のアプリケーションへの配信に対してのみ有効です。また、APNsは証明書失効リストを管理しています。プロバイダの証明書がこのリストに存在する場合、APNsはプロバイダの信頼を無効にする（つまり、接続を拒否する）こともあります。

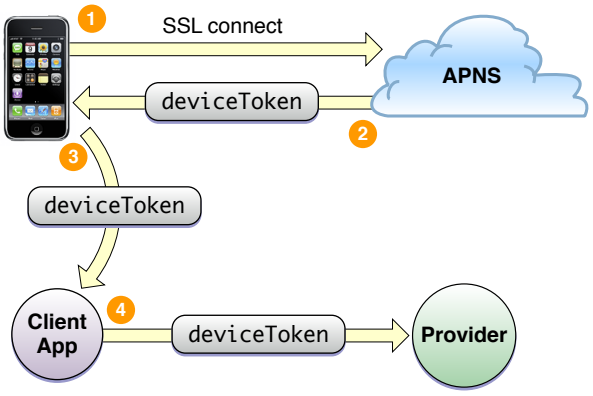
トークンの生成と共有

iPhoneアプリケーションは、Push Notificationを受信するための**登録**をしなければなりません。通常、これは、アプリケーションがデバイスにインストールされた直後に行われます（この手続きについては「iPhone OSクライアントの実装」（33 ページ）で説明します）。iPhone OSは、アプリケーションからの登録要求を受け取ると、APNsに接続してその要求を転送します。APNsは、一意のデバイス証明書に含まれている情報を使用してデバイストークンを生成します。このデバイストークンには、デバイスの識別子が含まれています。次に、トークンキーを利用してデバイストークンを暗号化してデバイスに返します。



デバイスは、このデバイストークンを要求を出してきたアプリケーションに`NSData`オブジェクトとして返します。次に、アプリケーションはこのデバイストークンをバイナリ形式または16進形式でプロバイダに渡さなければなりません。図 2-3は、トークンの生成と共有の順番を示しています。さらに、プロバイダにデバイストークンを供給する際のクライアントアプリケーションの役割も示しています。

図 2-3 デバイストークンの共有

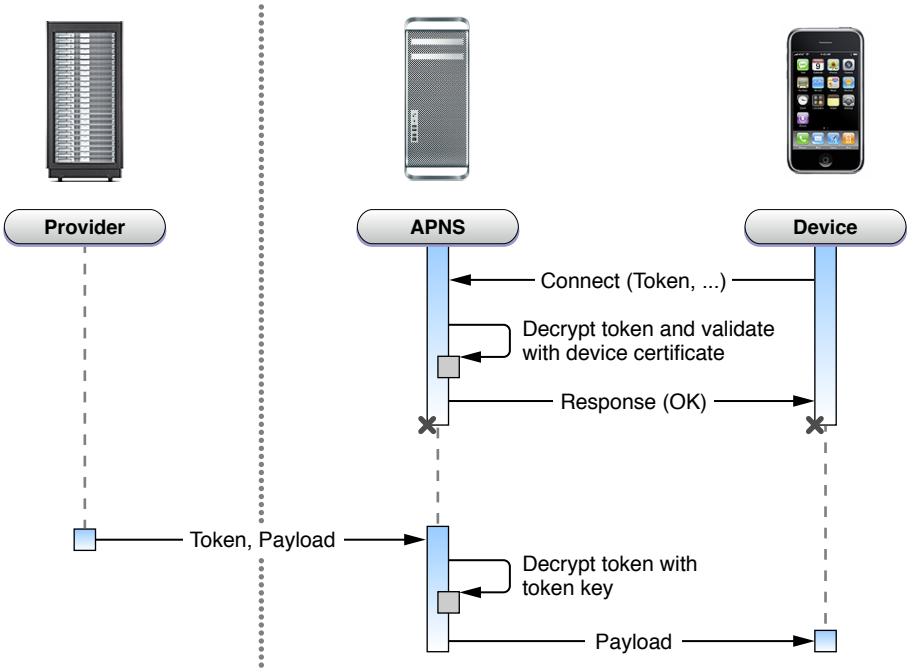


このような形態のトークンによる信頼フェーズによって、後で認証に使われるトークンを生成するのはAPNsだけであることが保証されます。また、APNsは、デバイスから渡されたトークンが、その特定のデバイスのために（そのデバイスのためだけに）以前用意したものと同じトークンであることを自ら確認できます。

トークンによる信頼（通知）

iPhone OSは、APNsからデバイストークンを取得した後は（「トークンの生成と共有」 （17 ページ）で説明）、APNsに接続するたびにそのトークンをAPNsに渡さなければなりません。APNsは渡されたデバイストークンを解読し、そのトークンが接続してきたデバイス用に生成されたものかどうかを検証します。検証のために、APNsは、トークンに含まれているデバイス識別子と、デバイス証明書のデバイス識別子が一致するかどうかを確認します。

デバイスへの配信のためにプロバイダがAPNsに送信するすべての通知には、そのデバイス上のアプリケーションから取得したデバイストークンが添付されていなければなりません。APNsはトークンキーを使用してトークンを解読し、通知が正当であることを確認します。次に、デバイストークンに含まれているデバイスIDを使用して、通知の配信先となるデバイスを判別します。



信頼に関連するコンポーネント

APNsのセキュリティモデルをサポートするために、プロバイダとデバイスは、特定の証明書、認証局(CA)証明書、またはトークンを所有していなければなりません。

- **プロバイダ**：各プロバイダには、一意のプロバイダ証明書と、APNsとの接続を検証するために使用する秘密暗号鍵が必要になります。この証明書はAppleから提供されますが、プロバイダが発行する特定のトピック（クライアントアプリケーションのバンドルID）を識別できなければなりません。通知のたびに、プロバイダはターゲットデバイスを識別するためのデバイストークンをAPNsに渡さなければなりません。プロバイダは、APNsサーバから提供された公開サーバ証明書を使用して、接続先のサービスを検証することもできます。
- **デバイス**：iPhone OSは、APNsから渡された公開サーバ証明書を使用して、接続先のサービスを認証します。iPhone OSは、一意の秘密鍵と証明書を持っており、これを使用してサービスへの認証を行いTLS接続を確立します。iPhone OSは、デバイスの起動中にこの秘密鍵とデバイス証明書を取得して、キーチェーンに格納します。また、iPhone OSは、特定のデバイストークンも保持しています。これは、サービス接続の処理中に受け取ります。登録済みの各クライアントアプリケーションは、このトークンをコンテンツプロバイダに送信する役割を担っています。

APNsサーバも、プロバイダとデバイスの接続とIDを検証するために、必要な証明書、CA証明書、および暗号鍵（秘密鍵と公開鍵）を持っています。

通知ペイロード

各Push Notificationと一緒にペイロードが送信されます。ペイロードは、クライアントアプリケーションにダウンロード可能なデータが存在することをユーザに警告する方法を指定します。通知ペイロードに許される最大サイズは256バイトです。Apple Push Notificationサービスでは、この上限を超える通知は拒否されます。通知の配信は「ベストエフォート型」のため、保証されないことを忘れないでください。

通知ごとに、プロバイダはRFC 4627に厳密に準拠したJSON辞書オブジェクトを作成しなければなりません。この辞書には、キー`aps`で識別されるもう1つの辞書が含まれている必要があります。`aps`辞書には、次のアクションを指定する1つ以上のプロパティが含まれています。

- ユーザに表示する警告メッセージ
- アプリケーションアイコンに付けるバッジの数字
- 警告音

注： 1つの通知で、警告メッセージ、アイコンバッジ、および警告音を組み合わせることができますが、Push Notificationがヒューマンインターフェイスに与える影響を考慮すべきです。たとえば、警告音付きの警告メッセージが頻繁に表示されると、ユーザはそれを便利ではなく邪魔だと感じるかもしれません。ダウンロードすべきデータが重要ではない場合は、特にそう感じるでしょう。

通知が届いたときにターゲットアプリケーションが実行されていない場合は、警告メッセージ、警告音、またはバッジが再生または表示されます。アプリケーションが実行されている場合は、iPhone OSがその通知を`NSDictionary`オブジェクトとしてアプリケーションデリゲートに渡します。この辞書には、それに対応するCocoaプロパティリストオブジェクト（`NSNull`も含む）が含まれています。

プロバイダは、Appleによって予約されている`aps`名前空間の外部に、カスタムペイロード値を定義することもできます。カスタム値にはJSONの構造体および基本型（辞書（オブジェクト）、配列、文字列、数字、Boolean）を使用しなければなりません。カスタムペイロードデータとして顧客情報を含めるべきではありません。代わりに、（ユーザインターフェイスの）コンテキストや内部基準の設定などの目的にカスタムペイロードデータを使用します。たとえば、カスタムペイロード値を、インスタントメッセージのクライアントアプリケーションで使用する会話識別子にしたり、プロバイダが通知を送信した日時を識別するタイムスタンプにできます。

重要： 通知の配信は保証されないため、ペイロードを使って重要なデータをアプリケーションに送信するために、Remote Notification機能を利用すべきではありません。また、機密データをペイロードに含めてはいけません。Remote Notification機能は、新しいデータが利用できることをユーザに*通知する*ためだけに利用すべきです。

表 2-1は、`aps`ペイロードのキーと値のリストです。

表 2-1 aps辞書のキーと値

キー	値の型	説明
alert	文字列または 辞書	このプロパティが含まれていると、iPhone OSは標準的な警告を表示します。alertの値として、文字列または辞書を指定できます。文字列を指定すると、その文字列は2つのボタン（「Close」と「View」）が付いた警告のメッセージテキストになります。ユーザが「View」をタップすると、アプリケーションが起動します。 alertの値として辞書を指定することもできます。この辞書のキーの説明は、表 2-2 （21 ページ）を参照してください。
badge	数値	アプリケーションアイコンのバッジとして表示する数字です。このプロパティが存在しない場合は、現在表示されているバッジの数字はすべて削除されます。
sound	文字列	アプリケーションバンドル内のサウンドファイルの名前です。このファイル内のサウンドが、警告音として再生されます。サウンドファイルが存在しないか、値としてdefaultが指定されている場合は、デフォルトの警告音が再生されます。このオーディオは、システムサウンドと互換性のあるオーディオデータフォーマットのいずれかでなければなりません。詳細については「カスタム警告音の準備」 （33 ページ）を参照してください。

表 2-2 alertプロパティの子プロパティ

キー	値の型	説明
body	文字列	警告メッセージのテキストです。
action-loc-key	文字列またはnull	文字列が指定されている場合は、2つのボタン付きの警告が表示されます。これらのボタンの動作については表 2-1で説明しています。ただし、iPhone OSは、この文字列をキーとして使用して現在のローカライズからローカライズ文字列を取得し、「View」の代わりに右ボタンのタイトルとして使用します。この値がnullの場合は、システムは「OK」ボタンだけが付いた警告を表示します。このボタンをタップすると単純に警告が消えます。詳細については「ローカライズ形式の文字列」 （22 ページ） を参照してください。
loc-key	文字列	現在のローカライズ（ユーザの言語環境によって設定される）に対応するLocalizable.stringsファイル内の警告メッセージ文字列のキーです。このキー文字列を%@指定子や%*n*$@指定子を使用して書式化すると、loc-argsで指定した変数に置き換えることができます。詳細については「ローカライズ形式の文字列」 （22 ページ）を参照してください。
loc-args	文字列の配列	loc-key内の書式指定子の代わりに表示する変数文字列値です。詳細については「ローカライズ形式の文字列」 （22 ページ）を参照してください。

注： iPhoneまたはiPod touchデバイスで、「Close」ボタンと「View」ボタンの両方を持つ警告内にメッセージテキストをそのまま表示する場合は、直接alertの値として文字列を指定します。辞書がbodyプロパティしか持たない場合は、alertの値として辞書を指定しないでください。

ローカライズ形式の文字列

ローカライズされた警告メッセージを表示するには2つの方法があります。1つは、通知を作成するサーバでテキストをローカライズする方法です。それには、デバイスで現在選択されている言語設定をサーバが検出しなければなりません（「プロバイダに現在の言語設定を渡す」（35 ページ）を参照）。もう1つは、クライアントアプリケーションが、サポート対象の各ローカライズ用に翻訳した警告メッセージ文字列をバンドルに格納しておく方法です。プロバイダは、通知ペイロードのaps辞書で、loc-keyプロパティとloc-argsプロパティを指定します。（アプリケーションが実行されていない場合）デバイスは通知を受信すると、これらのaps辞書プロパティを使用して、現在の言語にローカライズされた文字列を検索し書式化します。そして、それをユーザに表示します。

ここでは、2番目の方法が動作する仕組みをもう少し詳しく説明します。

iPhone OS上のアプリケーションは、画像、サウンド、テキストなどのリソースをサポート対象の各言語用に国際化できます。国際化によって、これらのリソースは集められて、2つの部分（言語コードと.lprojという拡張子）から構成される名前（たとえば、fr.lproj）のバンドルのサブディレクトリに配置されます。プログラムで表示されるローカライズ文字列は、Localizable.stringsというファイルに保存されています。このファイルの各エントリにはキーとローカライズ文字列値が含まれています。この文字列には、変数値に置き換え可能な書式指定子を含めることができます。アプリケーションが特定のリソース（たとえば、ローカライズ文字列）を要求すると、ユーザによって現在選択されている言語にローカライズされたリソースを取得します。たとえば、設定言語がフランス語の場合、警告メッセージ用の文字列の値はアプリケーションバンドルのfr.lprojディレクトリ内のLocalizable.stringsから取得されます（iPhone OSは、NSLocalizedStringマクロを利用してこの要求を出します）。

注： action-loc-keyプロパティの値が文字列の場合も、この一般的なパターンに従います。この文字列は、現在選択されている言語のローカライズディレクトリ内のLocalizable.stringsへのキーです。iPhone OSは、このキーを使用して警告メッセージの右側のボタン（“アクション”ボタン）のタイトルを取得します。

わかりやすいように例を考えてみましょう。プロバイダは、警告プロパティの値として次のような辞書を指定しています。

```
"alert" : { "loc-key" : "GAME_PLAY_REQUEST_FORMAT", "loc-args" : [ "Jenna",
"Frank"] },
```

デバイスはこの通知を受信すると、"GAME_PLAY_REQUEST_FORMAT"をキーとして使用し、現在の言語の.lprojディレクトリ内のLocalizable.stringsファイルから、それに関連付けられた文字列値を検索します。現在のローカライズに次のようなLocalizable.stringsエントリがあるとします。

```
"GAME_PLAY_REQUEST_FORMAT" = "%@ and %@ have invited you to play Monopoly";
```

デバイスは、「Jenna and Frank have invited you to play Monopoly」というメッセージを含む警告を表示します。

書式指定子%@のほかに、%*n*$@書式指定子を使用して、位置を指定して文字列変数を置き換えることができます。*n*は、loc-args内の置換対象の配列値のインデックス（1から始まる）です（パーセント記号（%）を表す%%指定子もあります）。たとえば、Localizable.stringsのエントリが次のようになっているとします。

```
"GAME_PLAY_REQUEST_FORMAT" = "%2$@ and %1$@ have invited you to play Monopoly";
```

デバイスは、「Jenna and Frank have invited you to play Monopoly」というメッセージを含む警告を表示します。

loc-keyプロパティとloc-argプロパティを使用した通知ペイロードの完全な例については、「JSONペイロードの例」の後半の例を参照してください。iPhone OSの国際化の詳細については、『*iPhone Application Programming Guide*』の「The Core Application」を参照してください。国際化についての一般的な情報は、『*Internationalization Programming Topics*』を参照してください。文字列の書式化については、『*String Programming Guide for Cocoa*』の「Formatting String Objects」で解説されています。

注： loc-keyプロパティとloc-argsプロパティ（および、たいていのalert辞書）は、本当に必要な場合にだけ使用してください。これらのプロパティの値は、特に長い文字列の場合、パフォーマンスが向上するどころか帯域幅を使い果たす恐れがあります。ほとんどとは言わないまでも多くのアプリケーションでは、これらのプロパティは必要ありません。メッセージ文字列はユーザから与えられるため、暗黙のうちに「ローカライズされている」からです。

JSONペイロードの例

通知のペイロード部分に関する次の例は、表 2-1で示したプロパティの実際の使い方を示しています。キー名に「acme」を含むプロパティは、カスタムペイロードデータの例です。この例には、読みやすくするために空白文字や改行文字が含まれていますが、パフォーマンス向上のためには、プロバイダは空白文字や改行文字を省くべきです。

例1：次のペイロードには、デフォルトの警告ボタン（「Close」と「View」）付きの警告メッセージに対応した簡単でお勧めの形式のaps辞書が含まれています。この例では、alertの値として、辞書ではなく文字列を使用しています。このペイロードにはカスタム配列プロパティも含まれています。

```
{
    "aps" : { "alert" : "Message received from Bob" },
    "acme2" : [ "bang",  "whiz" ]
}
```

例2：この例のペイロードでは、aps辞書を使用して、左側には「Close」ボタン、右側の"アクション"ボタンにはローカライズされたタイトルが表示される警告メッセージをデバイスに要求しています。ここでは、「Play」と同じ意味のローカライズ文字列を取得するために、現在選択されている言語のLocalizable.stringsファイルのキーとして、「PLAY」を使用しています。また、このaps辞書は、アプリケーションアイコンに付けるバッジに5と表示するように要求しています。

```
{     "aps" : {         "alert" : { "body" : "Bob wants to play poker",
"action-loc-key" :"PLAY" },         "badge" : 5,     },     "acme1" : "bar",
   "acme2" : [ "bang",  "whiz" ] }
```

例3：この例のペイロードでは、デバイスが「Close」と「View」の両方のボタンが付いた警告メッセージを表示するよう指定しています。また、アプリケーションアイコンに9という数字のバッジを付け、通知の配信時にバンドルされている警告音を鳴らすことも要求しています。

```
{
    "aps" :{
        "alert" : "You got your emails.",
        "badge" : 9,
        "sound" : "bingbong.aiff"
    },
    "acme1" : "bar",
    "acme2" : 42
}
```

例4：この例のペイロードのおもしろい点は、アプリケーションバンドルからローカライズ形式の文字列を取得し、適切な場所にある変数文字列値（loc-args）に置き換えるために、alert辞書の子プロパティloc-keyとloc-argsを使用していることです。また、カスタムサウンドを指定していますし、カスタムプロパティも含まれています。

```
{
    "aps" :{
       "alert" :{ "loc-key" : "GAME_PLAY_REQUEST_FORMAT", "loc-args" : [ "Jenna",
 "Frank"] },
        "sound" : "chime",
    },
    "acme" : "foo",
}
```

例5：次の例は、空のaps辞書を示しています。badgeプロパティがないため、アプリケーションアイコン上のバッジに付けられた数字は削除されます。acme2カスタムプロパティは2つの整数の配列です。

```
{
    "aps" :{
    },
    "acme2" : [ 5,  8 ]
}
```

パフォーマンスを向上させるために、ペイロードを通知に含める前に、すべての空白文字と改行文字をペイロードから削除することを忘れないでください。

プロビジョニングおよび開発

サンドボックス環境と製品環境

クライアントサーバアプリケーションのプロバイダ側を開発してデプロイするには、iPhoneデベロッパプログラムのポータルからSSL証明書を取得する必要があります。1つの証明書は、バンドルIDで識別される1つのアプリケーションに限定されています。また、証明書はそれぞれ、次の2つの開発環境のいずれか1つに限定されています。それぞれの開発環境は固有のIPアドレスを持っています。

- **サンドボックス**：サンドボックス環境は、プロバイダアプリケーションの初版の開発とテストに使用されます。サーバユニットの数は少ないものの、製品環境と同じサービスセットが提供されます。サンドボックス環境は、シミュレートされたエンドツーエンドテストを可能にする仮想デバイスとしての役割も果たします。

  サンドボックス環境には、gateway.sandbox.push.apple.comのポート2195からアクセスできます。

- **製品**：製品版のプロバイダアプリケーションをビルドするときは製品環境を使用します。製品環境を使用するアプリケーションは、Appleの信頼性要件を満たしている必要があります。

  製品環境には、gateway.push.apple.comのポート2195からアクセスできます。

サンドボックス（開発）環境用と製品環境用に、別々の証明書を取得する必要があります。この証明書はPush Notificationを受信するアプリケーションの識別子に関連付けられます。この識別子にはアプリケーションのバンドルIDが含まれています。これらの環境のいずれかのプロビジョニングプロファイルを作成すると、必要な資格が自動的にこのプロファイルに追加されます。これには、Push Notificationに固有の資格（`<aps-environment>`）も含まれます。この2つのプロビジョニングプロファイルは、DevelopmentとDistributionと呼ばれます。Distributionプロビジョニングプロファイルは、アプリケーションをApp Storeに投稿するための必要条件です。

どの環境で作業をしているかは、Xcodeでコード署名IDを選択すると判別できます。「iPhone Developer: *Firstname Lastname*」という証明書／プロビジョニングプロファイルのペアが表示された場合は、サンドボックス環境にいます。「iPhone Distribution: *Companyname*」という証明書／プロビジョニングプロファイルのペアが表示された場合は、製品環境にいます。この2つの環境を区別しやすくするために、XcodeでDistributionリリース構成を作成するとよいでしょう。

SSL証明書がプロビジョニングプロファイルに含まれていなくても、この証明書と特定のアプリケーションIDが関連付けられているため、`<aps-environment>`がプロファイルに追加されます。その結果、この資格がアプリケーションに組み込まれて、Push Notificationの受信が可能になります。

プロビジョニングの手順

iPhoneデベロッパプログラムでは、開発チームの各メンバは、Team Agent、Team Admin、およびTeam Memberの3つの役割のいずれか1つを持ちます。これらの役割の違いは、iPhone開発証明書とプロビジョニングプロファイルに関する部分です。Team Agentは、Development（サンドボックス）SSL証明書とDistribution（製品）SSL証明書を作成できる、チーム内で唯一の人物です。Team Admin

とTeam Agentは、DevelopmentとDistributionの両方のプロビジョニングプロファイルを作成できます。Team Memberは、証明書とプロビジョニングプロファイルのダウンロードとインストールのみができます。この後のセクションで説明する手順では、これらの役割を参照します。

注： iPhoneデベロッパプログラムのポータルでは、すべてのiPhoneデベロッパプログラムメンバに『*Program Portal User Guide*』と、証明書の作成とプロビジョニングのすべての側面を説明した一連のビデオを公開しています。この後のセクションでは、APNs固有の手順に焦点を当て、その他の側面については簡単に説明します。ポータルにアクセスするには、iPhoneデベロッパプログラムのメンバは、iPhone Dev Center (http://developer.apple.com/jp/iphone/)にアクセスしてログインし、「Program Portal」ボタンをクリックします。

SSL証明書と鍵の作成

iPhone Dev Centerのプログラムのポータルで、Team AgentはAPNs用のアプリケーションIDを選択します。また、Team Agentは次の手順を実行してSSL証明書を作成します。

1. ウインドウの左側のサイドバーにある「App IDs」をクリックします。

   次のページに、有効なアプリケーションIDが表示されます。アプリケーションIDは、アプリケーションのバンドルIDの前に、Appleが作成した10文字のコードが付加された構成になっています。Team Adminは、バンドルIDを入力しなければなりません。証明のためには固有のバンドルIDを入力する必要があります。“ワイルドカード”付きのアプリケーションIDは使用できません。

2. サンドボックスSSL証明書（Developmentプロビジョニングプロファイルに関連付けられている）の場合はアプリケーションIDを指定して、「Configure」をクリックします。

   このアプリケーションIDに対する証明書を構成するには、Apple Push Notificationサービスの列の下に「Available」と表示されていなければなりません。

3. 「Configure App ID」ページで、「Enable for Push Notification Services」ボックスにチェックし、「Configure」ボタンをクリックします。

   このボタンをクリックするとAPNs Assistantが起動します。このガイドに従って次の一連の手順を実行します。

4. 最初の手順では、キーチェーンアクセスアプリケーションを起動して、CSR（証明書署名要求：Certificate Signing Request）を生成する必要があります。

   アシスタントの指示に従います。CSRの生成が終了したら、証明書アシスタントの「完了」をクリックしてAPNs Assistantに戻ります。

   CSRの生成時に、キーチェーンアクセスは秘密暗号鍵と公開暗号鍵のペアを生成します。秘密鍵はデフォルトでログインキーチェーンに組み込まれます。公開鍵は、CA（認証局）に送信されるCSRに含められます。CAから証明書が戻ってくると、その証明書の項目の1つが公開鍵になっています。

5. 「Submit Certificate Signing Request」ペインで「ファイルを選択」をクリックします。前の手順で作成したCSRファイルに移動し、それを選択します。

6. 「Generate」ボタンをクリックします。

「Generate Your Certificate」ペインが表示されている間に、APNs Assistantはクライアント SSL証明書を構成して生成します。これが正常に終了すると、「Your APNs Certificate has been generated.」というメッセージが表示されます。「Continue」をクリックして次の手順に進みます。

7. 次のペインで、「Download Now」ボタンをクリックして証明書ファイルをダウンロード場所にダウンロードします。その場所に移動し、その証明書ファイル（cerという拡張子を持つ）をダブルクリックしてキーチェーンにインストールします。終了したら、APNs Assistantの「Done」をクリックします。

   このファイルをダブルクリックすると、キーチェーンアクセスが起動します。プロバイダの開発に使用しているコンピュータ上のログインキーチェーンに証明書がインストールされていることを確認します。APNs SSL証明書は通知サーバ上にインストールする必要があります。

これら3つの手順が終了すると、iPhone Dev Centerのポータルの「Configure App ID」ページに戻ります。証明書には緑色の丸と「Enabled」というラベルが付いているはずです。

製品環境用の証明書を作成するには同じ手順を繰り返します。ただし、製品環境の証明書用のアプリケーションIDを選択します。

プロビジョニングプロファイルの作成とインストール

次に、Team AdminまたはTeam Agentは、サーバ側のRemote Notification開発で使用するプロビジョニングプロファイル（DevelopmentまたはDistribution）を作成しなければなりません。このプロビジョニングプロファイルは、アプリケーションのデベロッパとそのデバイスを、承認済みの開発チームと関連付けて、これらのデバイスをテストに使用できるようにするための資産を集めたものです。このプロファイルには、証明書、デバイス識別子、アプリケーションのバンドルID、およびすべての資格（<aps-environment>を含む）が含まれています。すべてのTeam Memberは、アプリケーションの実行とテストを行うデバイスにこのプロビジョニングプロファイルをインストールする必要があります。プロビジョニングプロファイル作成の詳細については、『*Program Portal User Guide*』を参照してください。

プロビジョニングプロファイルをダウンロードしてインストールするために、Team Memberは次の手順を実行する必要があります。

1. プログラムポータルの「Provisioning」ページに移動します。
2. APNsに登録したApp IDを含む新規のプロビジョニングプロファイルを作成します。
3. このプロファイルファイル（mobileprovisionという拡張子を持つ）をダウンロード場所からXcodeまたはiTunesのアプリケーションアイコンにドラッグします。

   または、プロファイルファイルを~/ライブラリ/MobileDevice/Provisioning Profilesに移動します。このディレクトリが存在しない場合は、ディレクトリを作成します。
4. Xcodeの「オーガナイザ(Organizer)」ウインドウで、「Provisioning Profiles」セクションに移動し、このプロファイルをデバイスにインストールします。

プロジェクトをビルドすると、バイナリには秘密鍵を使用して証明書が付けられます。

SSL証明書と鍵のサーバへのインストール

事前に取得しておいたSSL Distribution証明書と秘密暗号鍵を、プロバイダのコードを実行して、サンドボックス版または製品版のAPNsに接続するサーバコンピュータにインストールする必要があります。それには、次の手順を実行します。

1. キーチェーンアクセスユーティリティを開き、左側のペインの「自分の証明書」カテゴリをクリックします。
2. インストールする証明書を検索し、三角形（ディスクロージャトライアングル）を開きます。

   三角形を開くと、証明書と秘密鍵の両方が表示されます。
3. 証明書と鍵の両方を選択して、「ファイル」メニューの「書き出す」を選択します。そして、それらを「個人情報交換(Personal Information Exchange) (.p12)」ファイルとして書き出します。
4. この.p12ファイルを新しいコンピュータにコピーします。
5. Mac OS XおよびMac OS X Serverシステム上で、この.p12ファイルをダブルクリックし、新しいコンピュータ上のキーチェーンにインストールします。

プロバイダとApple Push Notificationサービスの通信

この章では、Apple Push Notificationサービス(APNs)との通信のためにプロバイダが使用するインターフェイスについて説明し、プロバイダに期待されるいくつかの機能について解説します。

プロバイダの一般的な要件

プロバイダは、「低レベルの」（バイナリ）インターフェイスを介してApple Push Notificationサービスと通信します。この低レベルインターフェイスは、バルクデータプロバイダとの高速で大容量のインターフェイスです。このインターフェイスは、バイナリコンテンツと組み合わせたストリームソケット設計を使用しています。この低レベルインターフェイスは非同期です。

製品環境のバイナリインターフェイスは`gateway.push.apple.com`のポート2195から利用できます。サンドボックス（開発）環境のバイナリインターフェイスは`gateway.sandbox.push.apple.com`のポート2195から利用できます。同じゲートウェイまたは複数のゲートウェイインスタンスに、複数の接続を並行して確立することもできます。プロバイダが確立できるゲートウェイ接続数には厳格な制限はありませんが、15個を超えないように制限するべきです。

インターフェイスごとに、TLS（またはSSL）を使用してセキュアな通信チャネルを確立する必要があります。これらの接続に必要なSSL証明書は、iPhoneデベロッパプログラムのポータルから提供されます（詳細については、「プロビジョニングおよび開発」（25ページ）を参照してください）。認証済みのプロバイダであることを証明するには、接続時にピアツーピア認証を使用してこの証明書をAPNsに渡さなければなりません。また、複数の通知にわたってAPNsとの接続を維持しなければなりません（APNsは、接続と切断がすばやく何度も繰り返される場合、それをDoS（サービス運用妨害：denial-of-service）攻撃と見なします）。

プロバイダは、Push Notificationの次の側面を担当します。

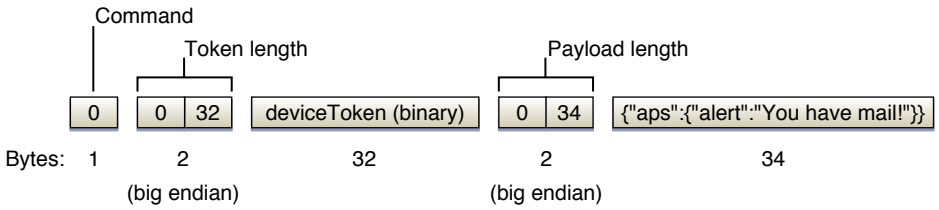
- 通知ペイロードを作成する必要があります（「通知ペイロード」（20 ページ）を参照）。
- アプリケーションアイコンに表示するバッジの数字を提供する必要があります。
- 定期的にフィードバック用のWebサーバに接続して、何度も配信失敗が報告されているデバイスの最新リストを取得すべきです。そして、これらのアプリケーションに関連付けられているデバイスへの通知の送信を停止する必要があります。詳細については、「フィードバックサービス」（31 ページ）を参照してください。

複数の言語で通知メッセージをサポートする場合に、ローカライズされた警告文字列をクライアント側で取得するために`aps`ペイロード辞書の`loc-key`プロパティと`loc-args`プロパティを使用しないときは、サーバ側で警告メッセージのテキストをローカライズする必要があります。それには、クライアントアプリケーションから現在の言語環境を検出する必要があります。「iPhone OSクライアントの実装」（33 ページ）では、この情報を取得するためのアプローチを示します。`loc-key`プロパティと`loc-args`プロパティの詳細については、「通知ペイロード」（20 ページ）を参照してください。

低レベルのインターフェイス

低レベルのインターフェイスは未加工のソケットを採用し、バイナリコンテンツを保持します。また、本質的にストリーミング方式であり、確認応答は行いません。図 4-1は、Remote Notificationパケットの形式を示しています。

図 4-1 Push Notificationのバイナリ形式



デバイストークンとペイロードの長さは、ネットワークのバイト順序（つまり、ビッグエンディアン）になっていなければなりません。さらに、デバイストークンをバイナリ形式にエンコードする必要があります。ペイロードはNULLで終了しては*いけません*。パフォーマンスを最適化するために、TCP/IPのNagleアルゴリズムを使用するか、または明示的に複数の通知を1回の転送で一括送信する必要があります。

リスト 4-1は、この低レベルインターフェイスを使用してAPNsにPush Notificationを送信する関数の例を示しています。この例は、事前に`gateway.push.apple.com`（または`gateway.sandbox.push.apple.com`）にSSL接続してピア交換認証を行っていることを前提としています。

リスト 4-1 低レベルインターフェイスを使用した通知の送信

```
static bool sendPayload(SSL *sslPtr, char *deviceTokenBinary, char *payloadBuff,
 size_t payloadLength)
{
    bool rtn = false;
    if (sslPtr && deviceTokenBinary && payloadBuff && payloadLength)
    {
        uint8_t command = 0; /* コマンド番号 */
        char binaryMessageBuff[sizeof(uint8_t) + sizeof(uint16_t) +
DEVICE_BINARY_SIZE + sizeof(uint16_t) + MAXPAYLOAD_SIZE];
        /* メッセージ形式は |COMMAND|TOKENLEN|TOKEN|PAYLOADLEN|PAYLOAD| */
        char *binaryMessagePt = binaryMessageBuff;
        uint16_t networkOrderTokenLength = htons(DEVICE_BINARY_SIZE);
        uint16_t networkOrderPayloadLength = htons(payloadLength);

        /* コマンド */
        *binaryMessagePt++ = command;

       /* トークン長（ネットワークバイト順序） */
        memcpy(binaryMessagePt, &networkOrderTokenLength, sizeof(uint16_t));
        binaryMessagePt += sizeof(uint16_t);

        /* デバイストークン */
        memcpy(binaryMessagePt, deviceTokenBinary, DEVICE_BINARY_SIZE);
        binaryMessagePt += DEVICE_BINARY_SIZE;
```

```
        /* ペイロード長（ネットワークバイト順序） */
        memcpy(binaryMessagePt, &networkOrderPayloadLength, sizeof(uint16_t));
        binaryMessagePt += sizeof(uint16_t);

        /* ペイロード */
        memcpy(binaryMessagePt, payloadBuff, payloadLength);
        binaryMessagePt += payloadLength;
        if (SSL_write(sslPtr, binaryMessageBuff, (binaryMessagePt -
binaryMessageBuff)) > 0)
            rtn = true;
    }
    return rtn;
}
```

フィードバックサービス

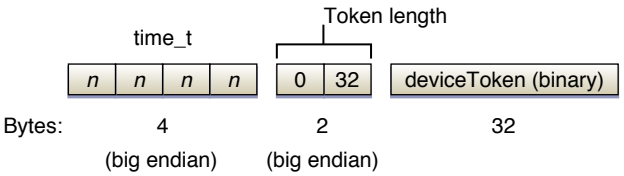
プロバイダがアプリケーションにPush Notificationを配信しようとしたときに、デバイス上にアプリケーションがもう存在しない場合、デバイスはその事実をApple Push Notificationサービスに報告します。このような状況は、ユーザがそのアプリケーションをアンインストールしている場合にしばしば発生します。アプリケーションへの配信に失敗したことがデバイスから報告された場合は、プロバイダがそのデバイスへの通知の送信を停止できるように、APNsは何らかの方法でプロバイダに知らせる必要があります。それによって、不必要なメッセージによるオーバーヘッドが減少し、システム全体のパフォーマンスが向上します。

その目的で、Apple Push Notificationサービスには、配信に失敗したデバイスの、アプリケーションごとのリストを継続的に更新するフィードバックサービスが含まれています。デバイスはバイナリ形式でエンコードされたデバイストークンによって識別されます。プロバイダは、定期的にこのフィードバックサービスに問い合わせて、対象のアプリケーション（トピックによって識別される）に対応するデバイストークンのリストを取得しなければなりません。そして、そのアプリケーションが識別されたデバイスに最近再登録されていないことを確認したら、プロバイダはそのデバイスへの通知の送信を停止すべきです。

フィードバックサービスへのアクセスは、Push Notificationの送信に使われるもとの同様のバイナリインターフェイスを介して行われます。製品環境のフィードバックサービスには`feedback.push.apple.com`のポート2196からアクセスします。サンドボックス環境のフィードバックサービスには`feedback.sandbox.push.apple.com`のポート2196からアクセスします。Push Notificationのバイナリインターフェイスと同様に、TLS（またはSSL）を使用してセキュアな通信チャネルを確立する必要があります。これらの接続に必要なSSL証明書は、通知を送信するために提供されたものと同じです。認証済みのプロバイダであることを証明するには、接続時にピアツーピア認証を使用してこの証明書をAPNsに渡さなければなりません。

接続すると、直ちに転送が始まります。APNsには一切コマンドを送信する必要はありません。読み込むべきデータがなくなるまで、フィードバックサービスによって書き込まれたストリームを読み込みます。受信したデータは次の形式を持つタプルです。

図 4-2 フィードバックタプルのバイナリ形式



`time_t`	デバイス上にアプリケーションがもう存在しないとAPNsが判断した日時を表すタイムスタンプ（4バイトの`time_t`値）。この値（ネットワークのバイト順序になっている）は、1970年からの秒数（UTCに基づく）を表します。 このタイムスタンプを使用して、デバイス上のアプリケーションが、デバイストークンがフィードバックサービスの記録された後に再登録されたかどうかを判断します。再登録されていない場合は、そのデバイスへのPush Notificationの送信を停止する必要があります。
トークン長	デバイストークンの長さ（ネットワークのバイト順序による2バイトの整数値）。
デバイストークン	バイナリ形式のデバイストークン。

注： APNsは、プロバイダがフィードバックサービスをチェックして、デバイス上に存在しないアプリケーションへのPush Notificationの送信を停止する作業をこまめに行っているかどうかを監視します。

iPhone OSクライアントの実装

この章では、クライアント側のRemote Notification機能を実装するために、iPhoneまたはiPod touchアプリケーションで行わなければならない（または行う可能性のある）作業について説明します。これらの通知はクライアント側のAPIでは「Remote Notification」と呼ばれるため、この章ではこの用語を使用します。

注： クライアントアプリケーションは、非同期または二次スレッドで常にプロバイダと通信する必要があります。

カスタム警告音の準備

プロバイダは、iPhone OSがアプリケーションへのRmote Ntificationを受信したときに再生するカスタム警告音を指定できます。このサウンドファイルは、クライアントアプリケーションのメインバンドル内になければなりません。

カスタム警告音はiPhone OSのシステムサウンド機能によって再生されるため、次のいずれかのオーディオデータフォーマットでなければなりません。

- リニアPCM
- IMA4 (IMA/ADPCM)
- μLaw
- aLaw

このオーディオデータは`aiff`、`wav`、または`caf`ファイルとして保存できます。次に、Xcodeで、このサウンドファイルをアプリケーションバンドルの非ローカライズリソースとしてプロジェクトに追加します。

`afconvert`ツールを使用してサウンドを変換することもできます。たとえば、16ビットリニアPCMのシステムサウンドの`Submarine.aiff`をIMA4オーディオのCAFファイルに変換するには、ターミナルアプリケーションで次のコマンドを使用します。

```
afconvert /System/Library/Sounds/Submarine.aiff ~/Desktop/sub.caf -d ima4 -f
caff -v
```

QuickTime Playerでサウンドを開き、「ウインドウ」メニューから「ムービーインスペクタを表示」を選ぶと、そのサウンドのデータフォーマットを判別できます。

Remote Notificationのための登録

アプリケーションのプロバイダから送信されたRemote Notificationを受信するには、アプリケーションはiPhone OS用のApple Push Notificationサービスに登録する必要があります。登録には次の3つのステージがあります。

1. アプリケーションが`UIApplication`の`registerForRemoteNotificationTypes:`を呼び出します。
2. アプリケーションが、デバイストークンを受信するために`UIApplicationDelegate`の`application:didRegisterForRemoteNotificationsWithDeviceToken:`メソッドを実装します。
3. アプリケーションが、デバイストークンを非オブジェクトのバイナリ値としてプロバイダに渡します。

この一連の処理の間に、アプリケーション、デバイス、Apple Push Notificationサービス、およびプロバイダの間で何が行われるかについては、「トークンの生成と共有」（17 ページ）の図 2-3に示しています。

アプリケーションは、起動のたびに登録を行って、現在のトークンをプロバイダに渡す必要があります。アプリケーションは、登録プロセスを開始するために`registerForRemoteNotificationTypes:`を呼び出します。このメソッドのパラメータは、アプリケーション側で受信したい初期の通知の種類（たとえば、アイコンバッジと警告音、ただし警告メッセージはなしなど）を指定する`UIRemoteNotificationType`ビットマスクです。その後は、ユーザが「設定(Settings)」アプリケーションの「通知(Notifications)」環境設定で、有効にする通知の種類を変更できます。また、`enabledRemoteNotificationTypes`メソッドを呼び出して、現在有効になっている通知の種類を取得することもできます。これらの通知の種類が有効になっていない場合は、たとえ通知ペイロードで指定されていても、iPhone OSは、バッジアイコン、警告メッセージの表示、または警告音の再生を行いません。

登録に成功すると、APNsはデバイスにデバイストークンを返します。iPhone OSは、`application:didRegisterForRemoteNotificationsWithDeviceToken:`メソッド内でこのトークンをアプリケーションデリゲートに渡します。アプリケーションは、プロバイダに接続して、このデバイストークンをバイナリ形式にエンコードして渡さなければなりません。トークンを取得する際に問題が発生した場合、iPhone OSは`application:didFailToRegisterForRemoteNotificationsWithError:`メソッドを呼び出してデリゲートに知らせます。このメソッドに渡される`NSError`オブジェクトには、エラーの原因が明確に記述されています。

注： アプリケーションが起動するたびに、デバイストークンを要求してそれをプロバイダに渡すことで、プロバイダが最新のデバイストークンを持つことを保証できます。バックアップの作成元となったデバイスとは異なるデバイスにバックアップを復元した場合は（たとえば、新しいデバイスにデータをインポートした場合）、再び通知を受信するために少なくとも一度はアプリケーションを起動する必要があります。バックアップデータを新しいデバイスに復元した場合や、オペレーティングシステムを再インストールした場合は、デバイストークンが変更されます。さらに、デバイストークンをキャッシュしてプロバイダに渡してはいけません。常に、デバイストークンは必要になったときにその都度システムから取得します。

iPhone OSはアプリケーションが登録済みかどうかを知っているので、アプリケーションを起動するたびに`registerForRemoteNotificationTypes:`を呼び出してもかまいません。その場合、iPhone OSは、通知の種類を設定したり余分なオーバーヘッドを生じさせることなく、ただちにデバイストークンをデリゲートに渡します。

リスト 5-1は、Remote Notificationの登録方法を示した簡単な例です（`SendProviderDeviceToken`は、プロバイダに接続してデバイストークンを渡すためにクライアントで定義されている仮定のメソッドです）。

リスト 5-1 Remote Notificationのための登録

```
- (void)applicationDidFinishLaunching:(UIApplication *)app {
   // その他のセットアップ作業をここで行う
    [[UIApplication sharedApplication]
registerForRemoteNotificationTypes:(UIRemoteNotificationTypeBadge |
UIRemoteNotificationTypeSound)];
}

// デリゲーションメソッド
- (void)application:(UIApplication *)app
didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)devToken {
    const void *devTokenBytes = [devToken bytes];
    self.registered = YES;
    [self sendProviderDeviceToken:devTokenBytes]; // カスタムメソッド
}

- (void)application:(UIApplication *)app
didFailToRegisterForRemoteNotificationsWithError:(NSError *)err {
    NSLog(@"Error in registration.Error: %@", err);
}
```

プロバイダに現在の言語設定を渡す

アプリケーションが、ローカライズされた警告メッセージをクライアント側で取得するために、`aps`辞書の`loc-key`プロパティと`loc-args`プロパティを使用しない場合は、プロバイダが、通知ペイロードに含める警告メッセージのテキストをローカライズする必要があります。ただしそれには、デバイスのユーザが設定言語として選択している言語を検出する必要があります（ユーザはこれを、「設定(Settings)」アプリケーションの「一般(General)」＞「言語環境(International)」＞「言語(Language)」ビューで設定します）。クライアントアプリケーションは設定言語の識別子をプロバイダに送信する必要があります。これは、標準化されているIETF BCP 47の言語識別子（「en」、「fr」など）です。

注： loc-keyプロパティとloc-argsプロパティ、およびクライアント側のメッセージローカライズの詳細については、「通知ペイロード」（20 ページ）を参照してください。

リスト 5-2は、現在選択されている言語を取得して、それをプロバイダに送信する手法を示しています。iPhone OSでは、NSLocaleのpreferredLanguagesで返される配列に1つのオブジェクト（選択されている言語を識別する言語コードをカプセル化したNSStringオブジェクト）が含まれています。UTF8Stringは、この文字列オブジェクトをUTF8エンコードのC文字列に変換します。

リスト 5-2　現在サポートされている言語を取得してプロバイダに送信する

```
NSString *preferredLang = [[NSLocale preferredLanguages] objectAtIndex:0];
const char *langStr = [preferredLang UTF8String];
[self sendProviderCurrentLanguage:langStr]; // カスタムメソッド
}
```

アプリケーションは、ユーザが現在のロケールを何か変更するたびに設定言語をプロバイダに送信することもできます。それには、NSCurrentLocaleDidChangeNotificationという名前の通知を検知し、通知処理メソッド内で、設定言語の識別コードを取得してそれをプロバイダに送信します。

設定言語がアプリケーションでサポートしていない言語の場合、プロバイダは広く普及している代替言語（英語、スペイン語など）にメッセージテキストをローカライズしなければなりません。

Remote Notificationの処理

一般に、データの取得をプロバイダに依存するiPhoneおよびiPod touchアプリケーションには、1つの共通点があります。アプリケーションは起動した直後にプロバイダに接続し、保留中のデータを取得します。通常、データがダウンロードされている間は進捗インジケータが表示されます。また、アプリケーションアイコンに関連付けられているバッジ番号をリセットします。リスト5-3は、この手順を簡単に示したものです。これは、applicationDidFinishLaunching:メソッド内でデリゲートが実装しています。

リスト 5-3　プロバイダからのデータのダウンロード

```
- (void)applicationDidFinishLaunching:(UIApplication *)application {
    // その他のセットアップ作業をここで行う
    [self startDownloadingDataFromProvider];  // カスタムメソッド
    application.applicationIconBadgeNumber = 0;
    // ...
}
```

アプリケーションが実行されていないときにデバイスに届いたほとんどのRemote Notificationに対して、このプロシージャで十分です。iPhone OSはアプリケーションアイコンにバッジを付けたり、警告音を鳴らしたり、または警告メッセージを表示します。ユーザは、アプリケーションをタップして起動することによってそれに応答します。または、警告メッセージが表示された場合、ユーザは「View」ボタンをタップしてアプリケーションを起動します。

ただし、applicationDidFinishLaunching:が適切な実装場所でない場合が2つあります。

- 通知が届いたときにアプリケーションが実行中の場合
- 通知ペイロードにアプリケーションが使用できるカスタムデータが含まれている場合

前者の場合は（iPhone OSがRemote Notificationを受信したときにアプリケーションが実行中の場合）、すぐにデータをダウンロードするには、`UIApplicationDelegate`の`application:didReceiveRemoteNotification:`メソッドを実装する必要があります。ダウンロードが終了したら、必ずアプリケーションアイコンからバッジを削除してください（アプリケーションが頻繁にプロバイダに新しいデータがないかどうかをチェックしている場合は、このメソッドを実装する必要はないかもしれません）。

後者の場合は、通知ペイロードにアプリケーションで使用できるカスタムデータが含まれているので、たとえばそれを使用して、初期のユーザインターフェイスをセットアップするときのコンテキストを提供できます。どちらの場合も、`application:didReceiveRemoteNotification:`（通知を受信したときにアプリケーションが実行中だった場合）と`application:didFinishLaunchingWithOptions:`(アプリケーションが実行されていない場合)の両方を実装できます。どちらのメソッドの場合も、それが呼び出されたときにiPhone OSはペイロード辞書を渡します。プロバイダからデータをダウンロードしたり、アプリケーションアイコンのバッジを削除するだけでなく、デリゲートはこの辞書からカスタムプロパティを取得して必要に応じてそれを適用します。

重要： この文書のほかの章で説明したように、顧客データや機密データを送信するために、通知ペイロードにカスタムプロパティを定義してはいけません。Remote Notificationの配信は保証されていません。カスタムペイロードプロパティの適切な使い方の一例は、メッセージがクライアントにダウンロード済みの電子メールアカウントを表す文字列です。アプリケーションは、この文字列をダウンロードインターフェイスに組み込むことができます。カスタムペイロードプロパティのもう1つの例は、プロバイダが最初に通知を送信した日時を表すタイムスタンプです。クライアントアプリケーションは、この値を使用してその通知がどのくらい古いかを測定できます。

書類の改訂履歴

この表は「*Apple Push Notification サービスプログラミングガイド*」の改訂履歴です。

日付	メモ
2009-05-22	Wi-Fiおよび登録頻度についての注の追加、サンドボックス用のゲートウェイアドレスを追加しました。いくつかの説明を明確化し、拡張しました。
2009-03-15	プロバイダがApple Push Notificationサービスを使用してクライアントアプリケーションにPush Notificationを送信する仕組みを説明した文書の初版。

書類の改訂履歴