Technical Note 03-03
FedEx Web サービスクライアント

By Frank Chang, 4D Inc. Technical Support
Technical Note 03-03

（原題: Building a Web Services Client using 4D: The Fed Ex® Soap serve）

概要

4D バージョン 2003 の強力な新機能のひとつは Web サービスを利用し、SOAP 規格に対応している他のアプリケーションと XML で通信ができるというものです。SOAP では、各アプリケーションがデータをどのように扱うかに関係なく、互いに通信することが可能になります。データやメソッドなどがアプリケーションの言語で書かれていたとしても、別のプログラミング言語を使用する他のアプリケーションと対話することができます。クライアントとしては、メソッドをコールし、受け取ったデータを処理することだけが問題になります。サーバとしては、メソッドを用意し、公開することだけが問題です。

Web サービスクライアントの実践

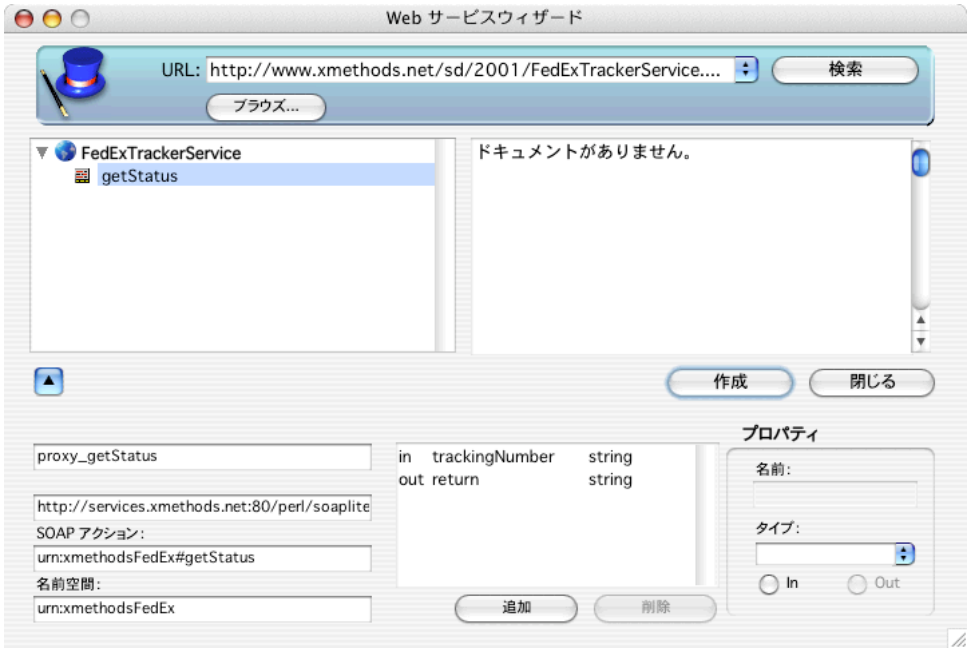
サンプルデータベースでは、Fed Ex®の SOAP サーバをコールしています。同サーバで公開されているメソッドは、どの SOAP クライアントでも利用することができますが、まずはそのサービスの存在を発見しなければなりません。サービスを発見するには Web Services Description Language を使用します。WSDL は、Web サービスの場所や動作をクライアントに説明するために XML 形式で提供されます。

サービスを発見したなら、Web サービス経由で SOAP サーバのサービスをコールするためのドライバである RPC つまり Remote Procedure Call プロジェクトメソッドを 4D 2003 で作成することができます。

RPC の作成までの過程は次のとおりです。4D 2003 のデザインモードで、ツールメニューから Web サービスウィザードを選択し、URL のフィールドに Fed Ex®の Web サービス WSDL アドレスを入力して検索ボタンをクリックします。

http://www.xmethods.net/sd/2001/FedExTrackerService.wsdl

画面の上部には利用できるメソッドの名前が表示され、クリックすると画面下部にそのプロパティが表示されます。getStatus メソッドの場合、文字列タイプのパラメータをひとつだけとることが分かります。これは追跡したい荷物の識別番号に相当します。また、メソッドが実行されると $0 に文字列タイプの変数が返されることも分かります。作成ボタンをクリックすると、RPC メソッドが自動的に作成されます。



サンプルデータベースについて

ストラクチャにはテーブルがふたつあり、[Shipments]では追跡番号、説明、状態が管理されます。このテーブルは、[Shipment History]とシーケンシャルに生成される ID 番号でリレートしており、[Shipment History]には[Shipments]の各レコードについて、状態の履歴が登録されています。[Shipments]のレコードで状態が更新されるたびに、[Shipment History]には新しいレコードが追加されます。

スタートアップでは、メールアドレスと SMTP サーバの情報、更新頻度の指定を促されます。この設定に基づいてアプリケーションは Web サービスを定期的にコールし、更新が検出されれば電子メールでそのことを通知します。

Start ボタンをクリックするか、メニューの項目 Start Services を選択することによって、実際の動作が始まります。停止するには Stop ボタンか Stop Services メニューを使用します。New ボタンをクリックすれば、新しい識別番号を追跡に加えることができます。Update ボタンは、更新を直ちに確認するために使用します。Delete ボタンで削除できるのは、状態が Delivered になっている項目だけです。

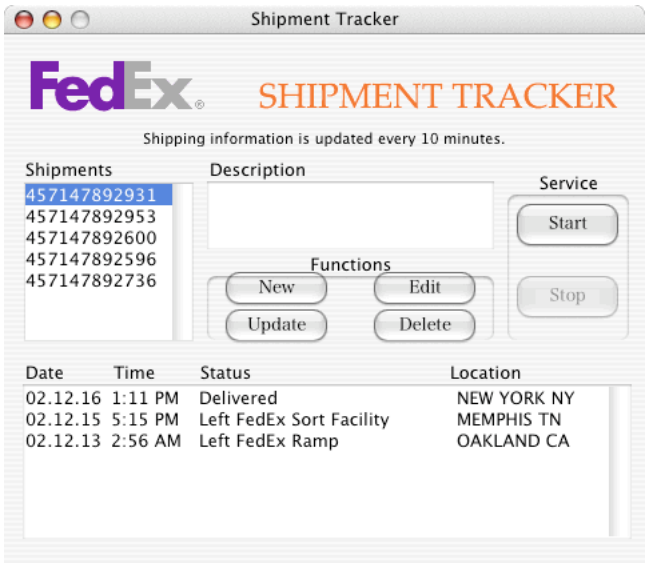
サンプルメソッドについて

Ship_Update

サービスを利用中(<>Ship_Quit_b が False)であれば、メソッド Ship_UpdateLoop を別プロセスで起動します。同メソッドは[Shipments]のカレントセレクションについて追跡情報を取得するためにループします。処理の内容は、各レコードの delivery status フィールドに左右されます。それではじめに Delivered 以外のレコードをクエリし、それぞれについて Ship_GetTrackingInfo メソッドをコールします。

Ship_GetTrackingInfo

主な働きは、Web サービスウィザードで作成されたプロキシメソッドをコールし、その返り値を解析することです。プロキシメソッドからの返り値は Ship_GetTrackingInfo のローカル変数に代入されます。この値は、そのレコードについての直前の情報と比較されます。直前の情報は、Shipment ID で[Shipment History]をクエリし、結果を ID フィールドの降順で並び替えた一番上のレコードです。変化がなければ、何も行なわれません。結果が異なれば、[Shipment History]に新規レコードが作成され、通知メールが送信されます。このとき ID フィールドの値は増加されるので、次回からこのレコードが直前の情報とみなされます。結果が delivered であれば、メソッドは$0 に 1 を返し、それは Ship_Update の変数$vres に代入されます。$vres が 1 であれば delivery status が True に更新されてレコードが更新されるので、次回から Ship_Update のクエリに含まれなくなります。



getStatus

4D によって自動生成されたプロキシメソッドにエラーハンドリングを追加したものです。メソッドをコールする時には$1 に追跡番号を渡します。この値は SET WEB SERVICE PARAMETER コマンドによって Web サービスの変数 trackingNumber にバインドされます。同コマンドには、第一引数としてサービスの URL を渡します。URL から分かるように、FedEx の SOAP サーバでリクエストを処理するのは Perl cgi スクリプトです。第二引数は SOAP アクションヘッダ、第三引数は実行するメソッドの名前、最後の引数が名前空間です。GET WEB SERVICE RESULT コマンドは、は SET WEB SERVICE PARAMETER とは逆に SOAP サーバから返された結果を 4D の変数(この場合は$0)にバインドします。

追加されたエラーハンドリングはふたつあります。ひとつ目はネットワーク接続の問題に対応し、ふたつ目はバインドに関する問題を扱います。このようなエラーハンドリングがないと getStatus はエラーを生成せずに実行を中断してしまいます。