ライセンス認証（アクティベーション）
実装ソリューション

◆認証 Web サービス
◆認証管理アプリケーション
◆配布アプリケーション用認証ライブラリ（DLL）

認証レスキュー!.NET

ユーザーズガイド

（3.0.3）

NEWTONE
株式会社ニュートン

－ 目次 －

●概要

「認証レスキュー！.NET（以降、認証レスキュー！）」は、貴社のパッケージアプリケーションにライセンス認証（アクティベーション）機能を付加して運用するためのソリューションです。
認証レスキュー！では、次の 3 つのソリューションを提供します。

1.Web サーバー用 PC に対するソリューション

インストーラが SQL Server 2014 Express のインストールや認証レスキュー！用のデータベースの設定、IIS の設定を行い、認証 Web サービス、環境設定をインストールします。
稼働後は認証 Web サービスが常時動作し、インターネットを経由したお客様（エンドユーザ）PC 上の貴社アプリケーションからの認証処理のリクエストに自動的に応答します。
また、同様にインターネット経由で、貴社の認証業務用社内 PC での認証管理システムからの認証状況の確認や新しいパッケージ製品の認証キーの作成などのリクエストも処理します。

2.認証業務用社内 PC に対するソリューション

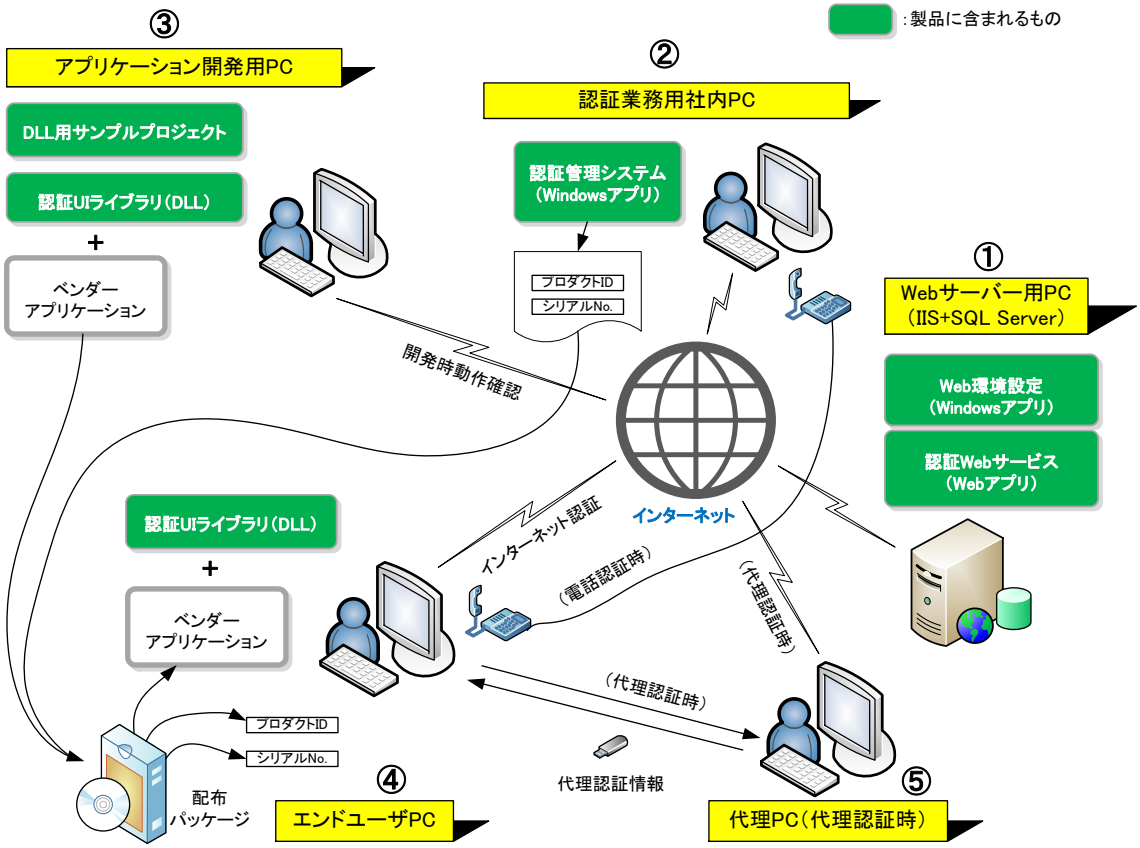
社内 PC で使用する「認証管理システム」をインストールします。認証管理システムでは出荷前製品の認証キーの作成やプロダクト ID やシリアル No.のラベル印刷、お客様（エンドユーザ）がインターネット経由で認証登録した記録などを閲覧できます。

3.「アプリケーション開発用 PC」へのソリューション

アプリケーションに認証登録用 UI（ユーザインターフェース）の機能を付加するための認証 UI ライブラリ（DLL）とサンプルプログラムをインストールします。この認証 UI ライブラリ（DLL）を利用することで貴社のアプリケーションにライセンス認証（アクティベーション）機能を簡単に実装することができます。

この「認証レスキュー！」を導入することで驚くほど早く確実に、貴社パッケージアプリケーションにライセンス認証機能を実装でき、ライセンス不正利用による損害を防止できます！

●運用イメージ



主な運用手順は次の通りです。具体的な手順は、後述します。（丸付き数字は上図の各 PC です。）

1.認証レスキュー！のインストールを行う

①Web サーバー用 PC、②認証業務用社内 PC、③アプリケーション開発用 PC にそれぞれインストールを行います。

2.インストールしたアプリケーションを使用して必要な各設定を行う

①Web サーバー用 PC と②認証業務用社内 PC で設定します。

3.インストールしたアプリケーションを使用して必要な各処理を行う

パッケージソフト出荷のための認証キー（プロダクト ID やシリアル No.など）を作成し、プロダクト ID とシリアル No.ラベル印刷を行います。②認証業務用社内 PC が対象です。

4.インストールした認証 UI ライブラリ（DLL）を利用し貴社アプリケーションに認証機能を実装する

③アプリケーション開発用 PC が対象です。

5.貴社アプリケーションをお客様（エンドユーザ）へ配布する

④エンドユーザ PC が対象です。上記 4 で完成した貴社アプリケーションと上記 3 で発行したプロダクト ID とシリアル No.をお客様（エンドユーザ）へ配布します。

6.お客様（エンドユーザ）がライセンス認証を実行する

④エンドユーザ PC または⑤（エンドユーザの）代理 PC で、お客様（エンドユーザ）が貴社アプリケーションの認証 UI を使用してライセンス認証を行い、その内容が貴社 Web サーバーなどの認証レスキュー！用のデータベースに記録されます。
その内容は、②認証業務用社内 PC の「認証状況」処理などで確認できます。

●認証レスキュー！で利用できるライセンスの種類

認証レスキュー！では、次表のライセンスを利用できます。

ライセンス形態	ライセンス種類	ライセンス認証方法			ライセンス期限	
		インターネット認証	代理（オフライン）認証	電話認証	無期限	有効期限
PC 固定型	無期限ライセンス	○	○	○	○	×
	有効期限ライセンス	○	○	×	×	○
フローティング型	フローティングライセンス	○	×	×	○	○

「ライセンス形態」について

・PC 固定型
エンドユーザの PC1 台ごとに登録が必要なライセンスです。エンドユーザは、登録したライセンスを解除することで他の PC に再度、登録することができます。また、5 ライセンスや 10 ライセンスといったマルチライセンスの登録も可能です。

・フローティング型
エンドユーザで貴社のアプリケーションを同時に使用できる最大の PC 台数を設定するライセンスです。エンドユーザは、最大 PC 台数以内であればどの PC でも貴社のアプリケーションを使用できます。
このライセンスは、エンドユーザの PC が常時インターネットに接続できる環境が必要ですが、世界中のどこにある PC でも、同じフローティングライセンス内での利用が可能です。

後述の以下のリンクもご覧ください。

有効期限機能の利用方法

フローティングライセンスの利用方法

●主な注目機能一覧

◆従来機能
・エンドユーザのプロキシサーバー環境に対応
・PC クラッシュ時の認証解除機能
・オフライン時電話認証機能
・認証 UI ライブラリ（DLL）の提供
・Web サービス用「環境設定」アプリの提供
・認証業務用「認証管理システム」アプリの提供
・Microsoft クラウド Azure 対応
・Amazon クラウド AWS 対応
・代理認証機能
・試用期間機能
・有効期限機能
・プロダクト ID 桁数自由設定機能
・シリアル No.桁数自由設定機能
・MAC アドレス識別情報付加
・CPU 情報識別情報付加
・認証状況の Excel データ出力
・ログ表示の Excel データ出力
・認証状態オンライン確認機能
・認証状況およびログにエンドユーザの IP アドレスを記録

◆拡張機能
・フローティングライセンス機能
・認証状況およびログにエンドユーザの PC 名を記録

◆便利機能
・各種データ閲覧時の検索条件設定機能
・認証キー自動ナンバリング機能
・認証キー一覧編集機能
・データベースのバックアップ（スケジュール化可）および、復元機能
・サンプルデータベース切替お試し機能
・DLL 用サンプルプロジェクトの提供

◆セキュリティ関連
・環境設定でのログインダイアログ指定機能
・DLL での貴社独自の暗号化情報設定機能
（認証レスキュー！を利用した他社とは別の暗号化）
・DLL はアセンブリ署名により偽装対策済み
・DLL は逆コンパイル対策の難読化済み
・SQL Server の SQL インジェクション対策済み（Web サービス）
・Web サービスのブラウザ表示隠ぺい化

●インストールされる内容

Web サーバー用 PC	・認証 Web サービス（Web アプリケーション） ・Web 環境設定（Windows アプリケーション） ・SQL Server 2014 Express + Management Studio	ユーザーズガイドなど
認証業務用 PC	・認証管理システム（Windows アプリケーション）	
アプリケーション開発用 PC	・認証 UI ライブラリ（DLL） ・サンプルプロジェクト（既定 UI 系サンプル、カスタム UI 系サンプル、ASP.NET 系サンプル） .NET Framework 4.5（Visual Basic 2015 用、C# 2015 用、Visual C++ 用） .NET6（Visual Basic 2022 用、C# 2022 用）	

●動作環境

・「Web サーバー用 PC」へのインストールが対応している OS
日本語 Microsoft Windows Server 2022/2019/2016/2012 R2/2012
日本語 Microsoft Windows 11(Home は除く)/10(Home は除く)/8.1(Pro 以上)/8(Pro 以上)
※上記 Windows OS 上の対応暦:グレゴリオ暦(西暦)のみ

・「Web サーバー用 PC」へのインストールが対応している IIS
IIS 10.0/8.5/8.0

・「認証業務用社内 PC」へのインストールによる「認証管理システム」が対応している OS
日本語 Microsoft Windows Server 2022/2019/2016/2012 R2/2012
日本語 Microsoft Windows 11/10/8.1/8
※上記 Windows OS 上の対応暦:グレゴリオ暦(西暦)のみ

・「アプリケーション開発用 PC」へのインストールによる「認証 UI ライブラリ(DLL)」が対応している OS
Microsoft Windows Server 2022/2019/2016/2012 R2/2012
Microsoft Windows 11/10/8.1/8

[DLL による OS および暦対応表]

DLL	【既定 UI 系、カスタム UI 系 DLL】 **Newtone.NR.FW45.dll / Newtone.NR.NET6.dll**	【ASP.NET 系 DLL】 **Newtone.NR.ASPNET.dll**
Windows 対応 OS	日本語版および英語版推奨	日本語版のみ
対応 OS 上の対応暦	・ChineseLunisolarCalendar/中国の太陰太陽暦 ・GregorianCalendar/グレゴリオ暦(西暦) ・HebrewCalendar/ヘブライ暦 ・HijriCalendar/回教暦 ・JapaneseCalendar/和暦 ・JapaneseLunisolarCalendar/日本の太陰太陽暦 ・JulianCalendar/ユリウス暦 ・KoreanCalendar/韓国暦 ・KoreanLunisolarCalendar/韓国の太陰太陽暦 ・PersianCalendar/ペルシャ暦 ・TaiwanCalendar/台湾暦 ・TaiwanLunisolarCalendar/台湾の太陰太陽暦 ・ThaiBuddhistCalendar/タイ仏暦 ・UmAlQuraCalendar/ウムアルクラ暦	・GregorianCalendar/グレゴリオ暦(西暦)のみ

・認証 UI ライブラリ(DLL)を利用するための開発環境(すべて日本語版のみ)
Visual Studio 2022/2019/2017/2015
・認証 UI ライブラリ(DLL)が動作する対応 Framework
・.NET Framework 4.8 / 4.7.2 / 4.7.1 / 4.7 / 4.6.2 / 4.6.1 / 4.6 / 4.5.2 / 4.5 1 / 4.5
・.NET 8/7/6.0

●処理・設定一覧

認証レスキュー！では運用に必要な各処理の他に、ライセンス認証に関連する細かな設定が指定できます。
認証レスキュー！で利用できる処理や設定の一覧を示します。

◆Web サーバー用 PC

・Web サービスの設定
・環境設定へのログイン設定
・データベースの指定
　NRD 規定/NRD サンプルデータ
　/任意接続文字列（マイクロソフト社クラウド Microsoft Azure など）
・データテーブル新規作成処理
・データベース更新処理
・NR 登録ライセンスの管理
・データベースのバックアップ処理（スケジュール化可能）
・データベースの復元

◆認証業務用社内 PC 「認証管理システム」

・Web サービスのアクセス設定
・プロキシサーバーの設定
・環境設定へのログイン設定
・認証キー作成（自動ナンバリング）処理
・認証キー作成（表形式）処理
・認証キー作成（個別）処理
・認証キー編集処理
・認証キー削除（表形式）処理
・認証キー削除（個別）処理
・ラベル印刷処理
・認証状況の表示/出力処理
・ログの表示処理

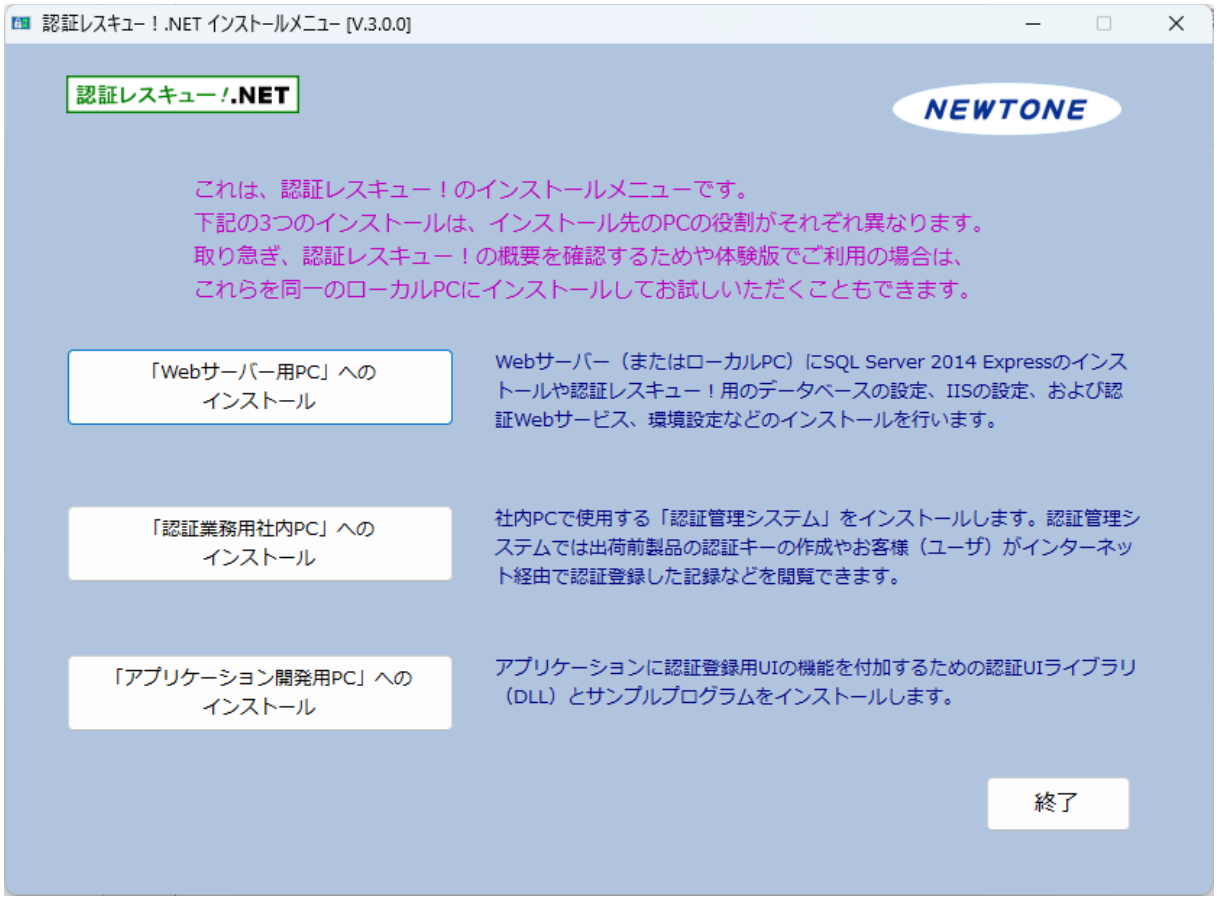
◆アプリケーション開発用 PC 「認証 UI ライブラリ（DLL）」の利用

＜付属サンプルプロジェクト＞
.NET Framework 4.5（Visual Basic 2015 用）：既定 UI 系サンプル、カスタム UI 系サンプル
.NET6（Visual Basic 2022 用）：既定 UI 系サンプル、カスタム UI 系サンプル
.NET Framework 4.5（C# 2015 用）：既定 UI 系サンプル、カスタム UI 系サンプル
.NET6（C# 2022 用）：既定 UI 系サンプル、カスタム UI 系サンプル
.NET Framework 4.5（Visual Basic 2015 用）：ASP.NET 系サンプル
.NET Framework 4.5（C# 2015 用）：ASP.NET 系サンプル
.NET Framework 4.5（Visual C++ 2015 用）：既定 UI 系サンプル、カスタム UI 系サンプル

＜プロパティ＞および＜メソッド＞の各機能（既定 UI 系、カスタム UI 系、ASP.NET 系）

●インストール

（パッケージの場合は）ディスク内のルートまたは（ダウンロードなどの場合は）解凍したフォルダにある「NRInstallMenu.exe」を実行してください。次のような画面が表示されます。



これは、認証レスキュー！のインストールメニューです。
3 つのインストールボタンがあり、インストール先の PC の役割がそれぞれ異なります。
取り急ぎ、認証レスキュー！の概要を確認するためや体験版でご利用の場合は、これらを同一のローカル PC にインストールしてお試しいただくこともできます。

1.「Web サーバー用 PC」へのインストール

Web サーバー（またはローカル PC）に SQL Server 2014 Express のインストールや認証レスキュー！用のデータベースの設定、IIS の設定、および認証 Web サービス、環境設定などのインストールを行います。

なお、Microsoft クラウド Azure の仮想マシンや Amazon クラウド AWS の仮想サーバーを利用する場合は、後述の「Microsoft Azure で認証レスキュー！を利用する方法」、「Amazon AWS で認証レスキュー！を利用する方法」をご覧ください。

このインストールを選択するとさらに次の Web インストーラの画面が表示されます。

ここでは、「データベースのインストール」と「IIS 設定と Web サービスのインストール」を行います。

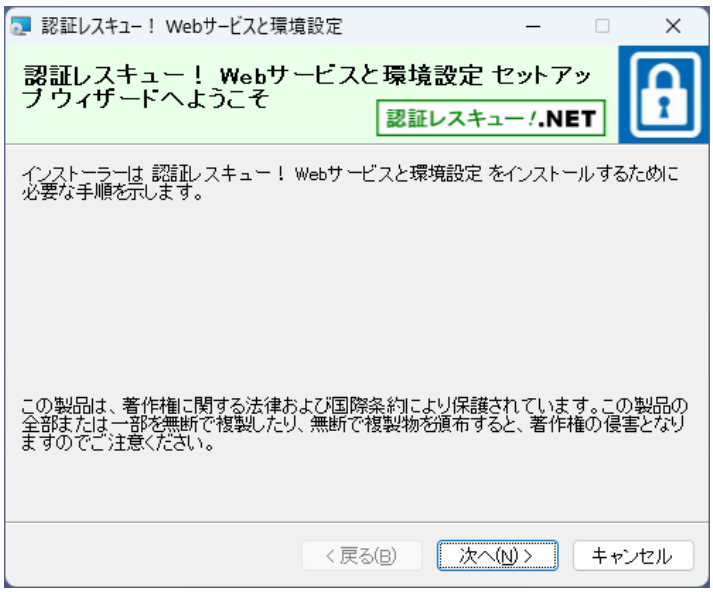
・データベースのインストール

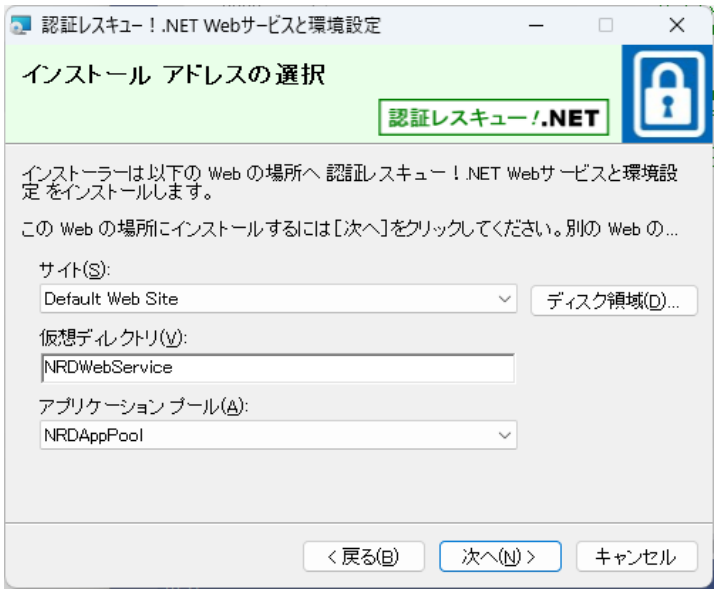
認証レスキュー！用のデータベース（SQL Server Express 日本語版）のインストールを行います。また、認証レスキュー！用のインスタンスと（データを除く）データベースも作成します。既に同じバージョンの SQL Server Express がインストールされている場合は、認証レスキュー！に必要な設定だけを行います。

・IIS 設定と Web サービスのインストール

認証 Web サービスのインストールを行います。Internet Information Services（IIS）が有効になっている必要がありますが、通常は、認証レスキュー！のインストーラが自動的に IIS を有効にして、IIS の不足している機能も自動的にインストールします。
「IIS 設定と Web サービスのインストール」のフレーム内で有効になっているボタンを選択するとインストールが始まります。選択できるボタンが無い場合は、現在の OS が対応外の可能性がありますのでご確認ください。

インストールを開始して IIS の設定が済むと確認画面の表示後、次のような「Web サービスと環境設定」のインストーラが起動されます。

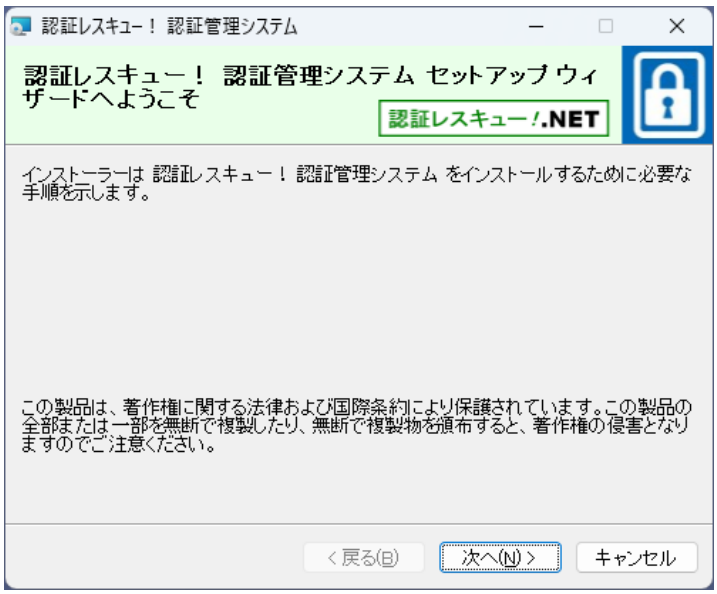




この画面では、サイト、仮想ディレクトリ、アプリケーションプールを指定しますが、通常はデフォルト（初期設定）のままで「次へ」ボタンを押します。以降は画面の指示に従ってください。

2.「認証業務用社内 PC」へのインストール

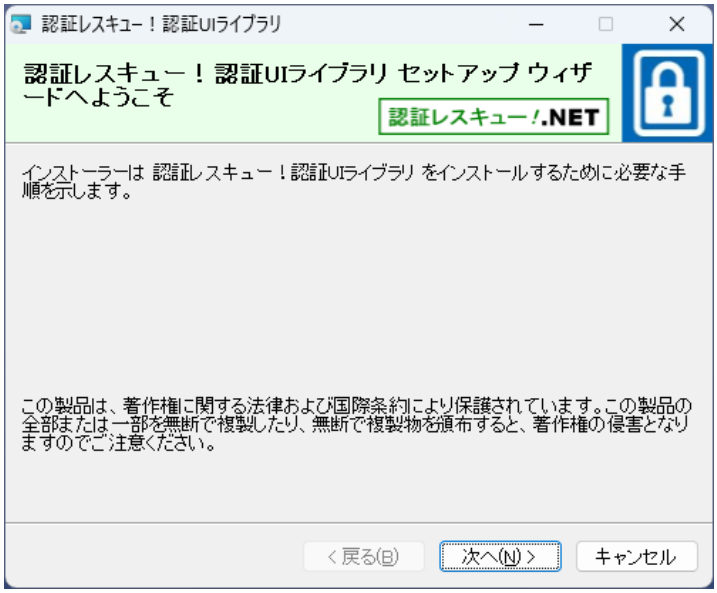
社内 PC で使用する「認証管理システム」をインストールします。認証管理システムでは出荷前製品の認証キーの作成やお客様（エンドユーザ）がインターネット経由で認証登録した記録などを閲覧できます。



インストールするには画面の指示に従ってください。

3.「アプリケーション開発用 PC」へのインストール

アプリケーションに認証登録用 UI の機能を付加するための認証 UI ライブラリ（DLL）とサンプルプログラムをインストールします。



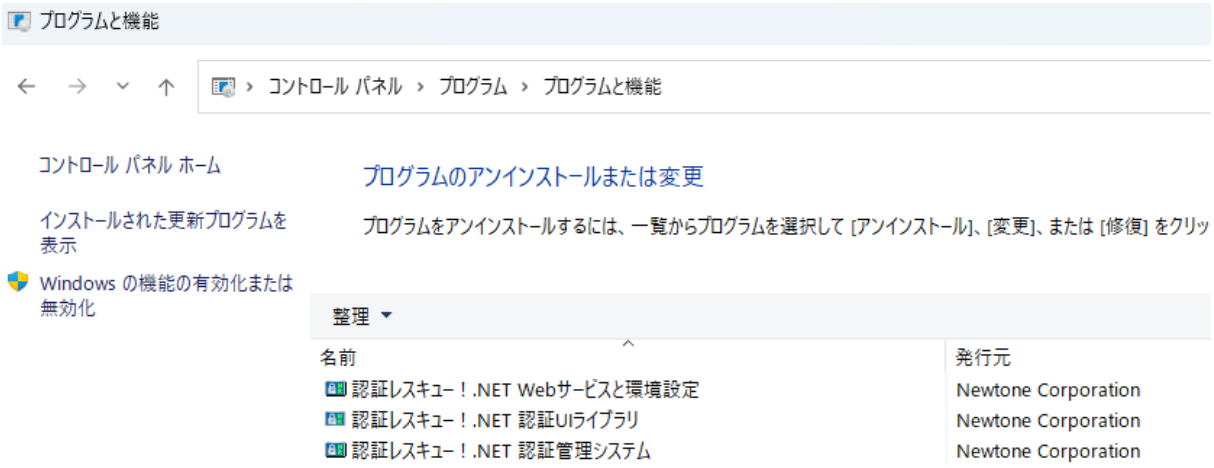
インストールするには画面の指示に従ってください。

4.インストールの終了後

すべてのセットアップが終了すると、デスクトップに「認証レスキュー！.NET Web 環境設定」へのショートカット、「認証レスキュー！.NET 認証管理システム」へのショートカット、「認証レスキュー！.NET 認証 UI ライブラリ」フォルダへのショートカット用アイコンがそれぞれ次のように作成されます。

また、プログラムメニュー（すべてのアプリ）には次のように「認証レスキュー！」が登録されます。

アンインストールは、「プログラムと機能」または「プログラムの追加と削除」から行います。



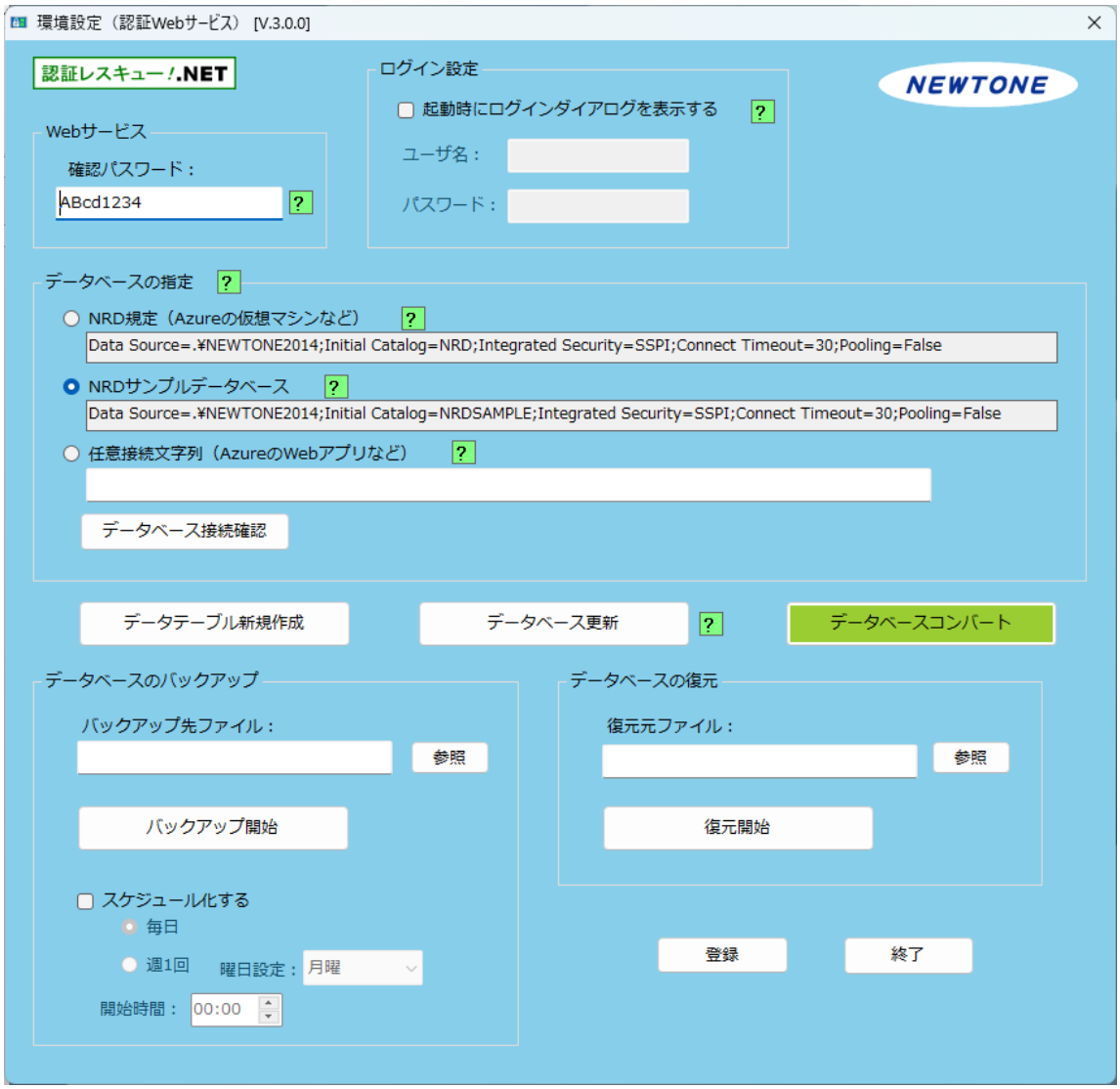
SQL Server Express 2014 をアンインストールする場合は、「プログラムと機能」または「プログラムの追加と削除」で次の項目をアンインストールします。PC の環境によっては、項目が異なる場合があります。

●Web サーバー用 PC と認証業務用社内 PC

インストールが終了したら、認証レスキュー！を利用する前の各種設定が必要です。

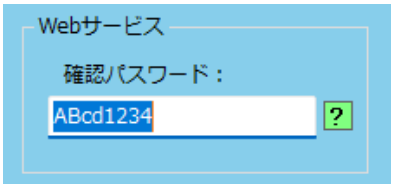
1.Web サーバー用 PC の「環境設定」処理

デスクトップ上の「認証レスキュー！.NET We 環境設定」へのショートカットを起動すると次の画面が表示されます。
この実行ファイルは、インストール先がデフォルトなら、
〈32bitOS の場合〉 C:¥Program Files¥Newtone¥NRD¥NRDWeb¥WebAdmin.exe、
〈64bitOS の場合〉 C:¥Program Files (x86)¥Newtone¥NRD¥NRDWeb¥WebAdmin.exe です。



◆必須設定処理項目
ここでは、設定が必須で省略できない設定について説明します。

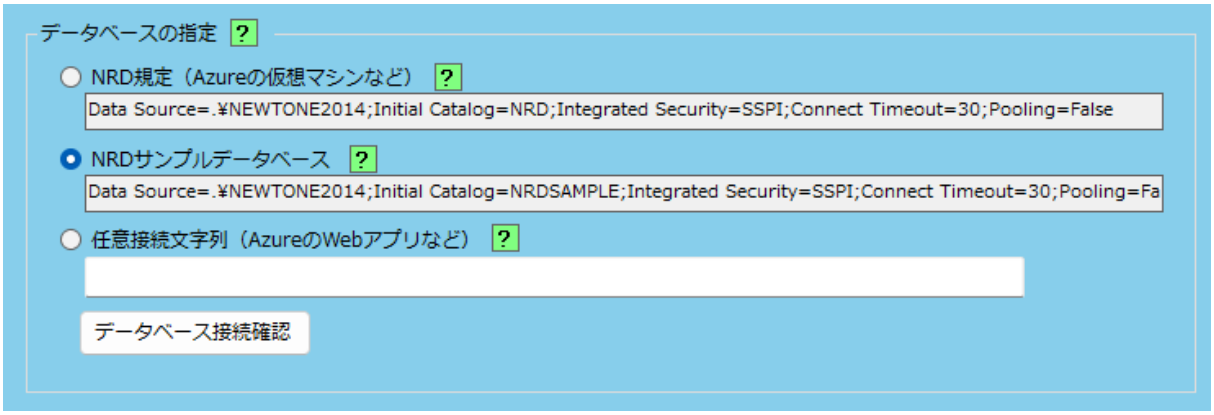
■Web サービス/確認パスワード



Web サービスを利用する場合の確認用のパスワードを設定します。
ここでは、Web サーバー側設定アプリケーション「認証 Web サービス」の環境設定処理の Web サービスの確認パスワードと同じものを設定します。
また、UI ライブラリ（DLL）利用時に次のプロパティにも同じ内容を設定する必要があります。
WebServiceCheckPassword（既定 UI 系利用時）
APIWebServiceCheckPassword（カスタム UI 系利用時）
APIxWebServiceCheckPassword（ASP.NET 系利用時）

確認パスワードは必須項目です、省略はできません。
8 文字以上で半角の次の文字が使用できます。
・大文字の英字（A～Z）
・小文字の英字（a～z）
・数字（0～9）
・記号（+-*/!#$%&()=¥@<>?）

■データベースの指定



認証レスキュー！で使用するデータベースを指定します。選択肢は 3 種類です。

NRD 規定

認証レスキュー！での規定のデータベースです。
実際の運用には通常、この「NRD 規定」を選択します。

マイクロソフト社のクラウドサービス Microsoft Azure の仮想マシン(Windows Server 2022 など)を利用する場合も、この「NRD 規定」を指定します。
Microsft Azure の仮想マシンを利用する方法は、後述の Microsoft Azure で「認証レスキュー！」を利用する方法を参照してください。

認証レスキュー！では、Web サーバー用 PC へのインストール時に SQL Server 2014 Express のインストールを選択できます。SQL Server 2014 Express をインストールして認証レスキュー！のインスタンス（Newtone）を作成し、NRD 規定のデータベース（NRD）を作成します。この段階では、データベース（NRD）にテーブル（データ）はまだありません。

後述の「テーブルデータ新規作成」を実行することで空のテーブルが作成されます。

NRD サンプルデータベース

認証レスキュー！のサンプル用のデータベースです。簡単なサンプルデータが各テーブルに既に格納されたデータベースです。取り急ぎ認証レスキュー！全体を把握したい場合などに選択します。

後述の「テーブルデータ新規作成」を実行することでテーブルが作成され、サンプルデータが格納されます。

任意接続文字列(Azure など)

Web サーバー用 PC とは異なる（データベース）サーバーなどに SQL Server が用意され、そこに認証レスキュー！用のデータベースを置く場合に選択します。

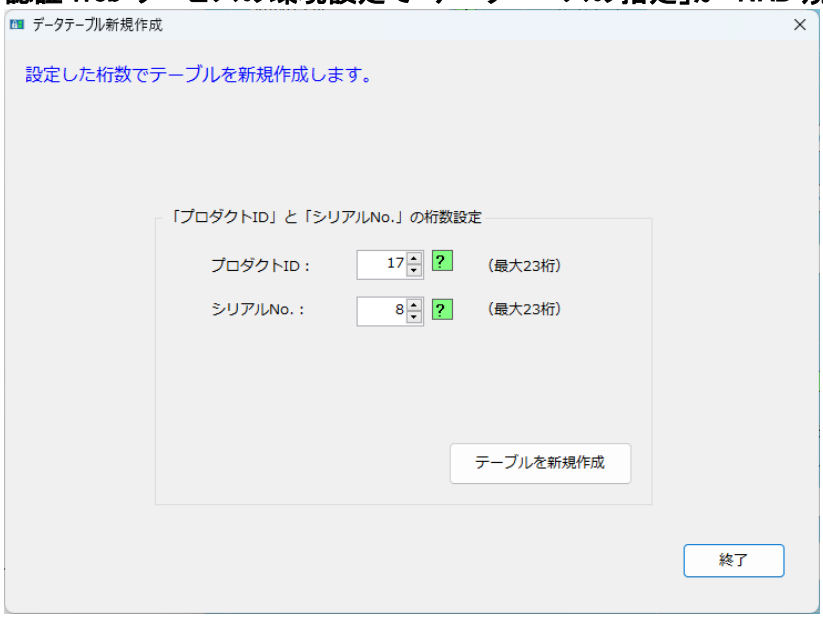
また、Web サーバー用 PC の SQL Server を利用するが、たとえばインスタンス名をデフォルトの「Newtone」から別のものに変更する場合などにもこの「任意接続文字列」を利用できます。
この任意接続文字列の使用時は、テキストボックスに正しい接続文字列を入力する必要があります。

上記の設定をしたら「データベース接続確認」ボタンを押して、データベースに接続できることを確認してください。

■データテーブル新規作成

次に、データベースのテーブルの新規作成をします。
環境設定で「データテーブル新規作成」ボタンを押します。

認証 Web サービスの環境設定で「データベースの指定」が「NRD 規定」の場合



「テーブルを新規作成」ボタンを押すとデータベース内に**データの無いテーブル**が作成されます。
実際のプロダクト ID やシリアル No.を含むキーデータは、「認証管理システム」のメニューの「認証キー作成」（自動ナンバリング）処理などで作成します。

以下にプロダクト ID やシリアル No.を簡単に説明しています。

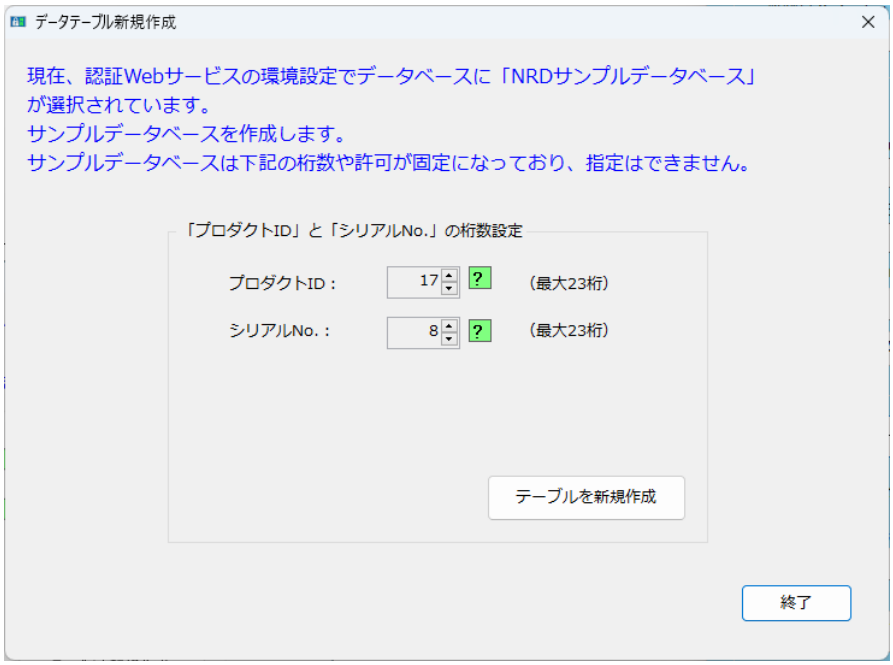
・プロダクト ID

プロダクト ID は、製品を表わす任意の文字列です。桁数は運用開始時に設定し、以降は変更できません。
桁数の初期値は 17 桁です。最大 23 桁です。
なお、データベースの選択で「NRD サンプルデータベース」をお試しいただく場合の桁数は 17 桁固定となり設定できません。
このプロダクト ID には、製品分類やバージョン、ライセンス数などを識別できる桁取りがあると管理しやすくなります。
たとえば、0000A-00002-00001 で製品 A のバージョン 2 の 1 ライセンスを、0000A-00002-00004 で製品 A のバージョン 2 の 4 ライセンスを、それぞれ示すように設定します。
プロダクト ID は、半角の次の文字が使用できます。
・大文字の英字（A～Z）
・小文字の英字（a～z）
・数字（0～9）
・記号（+-*/!#$%&()=¥@<>?）
このプロダクト ID と「シリアル No.」で出荷時の一意の製品を示すキーとなります。

・シリアル No.

シリアル No.は、同一製品の識別連番を表わす文字列です。桁数は運用開始時に設定し、以降は変更できません。桁数の初期値は 8 桁です。最大 23 桁です。
なお、データベースの選択で「NRD サンプルデータベース」をお試しいただく場合の桁数は 8 桁固定となり設定できません。
「認証キー作成」（自動ナンバリング）処理では、任意の上位桁数の文字列を指定することができます。
たとえば、8 桁中、上位 3 桁を"ABC"と指定した場合は次のような
シリアル番号の自動作成が可能です。
ABC00001
ABC00002
ABC00003
ABC00004
・・・・・
シリアル No. は、半角の次の文字が使用できます。
・大文字の英字（A～Z）
・小文字の英字（a～z）
・数字（0～9）
・記号（+-*/!#$%&()=¥@<>?）
「プロダクト ID」とこのシリアル No.で出荷時の一意の製品を示す
キーとなります。

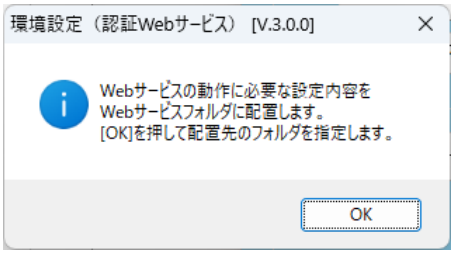
認証 Web サービスの環境設定で「データベースの指定」が「NRD サンプルデータベース」の場合



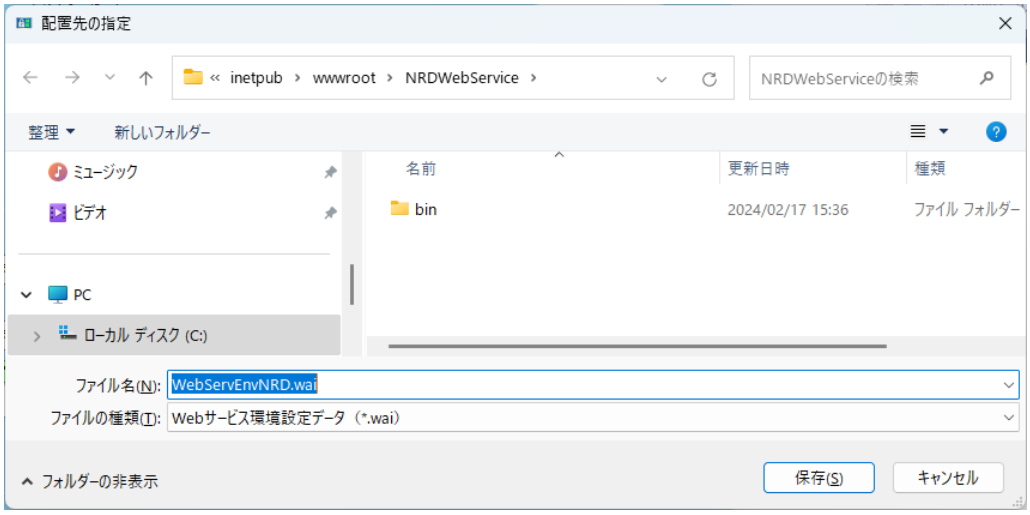
「テーブルを新規作成」ボタンを押すとデータベース内に**サンプルデータが入ったテーブル**が作成されます。

■「登録」ボタン

環境設定が終了したら「登録」ボタンを押します。次のようなメッセージが表示されます。



ここで、「OK」ボタンを押すと、次のような「配置先の指定」ダイアログが表示されます。

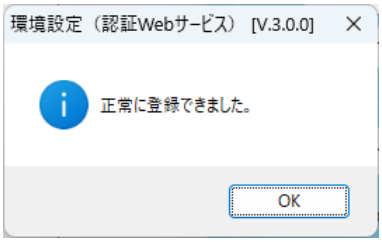


環境設定で入力した情報の内、Web サービスの動作に必要な情報を Web サービス環境設定データ（WebServEnv.wai）ファイルとして、この「配置先の指定」ダイアログで IIS の認証レスキュー！用の Web サービスが配置してあるフォルダにも出力します。通常は、自動的に（例：C:¥inetpub¥www

root¥NRDWebService などの）適切な出力先が表示されますので、確認してそのまま「保存」ボタンを押します。

なお、環境設定で入力した情報で Web サービスの動作と関係しない情報は、Web サービス環境設定データ（WebServEnv.wai）ファイルではなく、この「環境設定」を実行している PC のレジストリに記録されます。

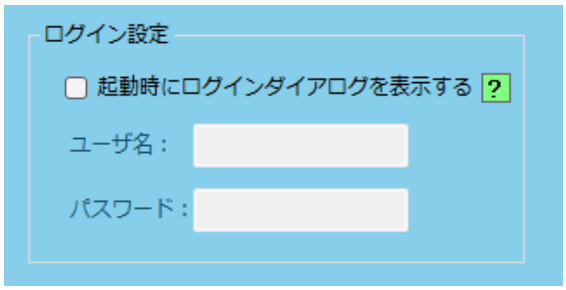
「保存」ボタン押下後、登録に成功すると次のメッセージが表示されます。



◆任意設定処理項目

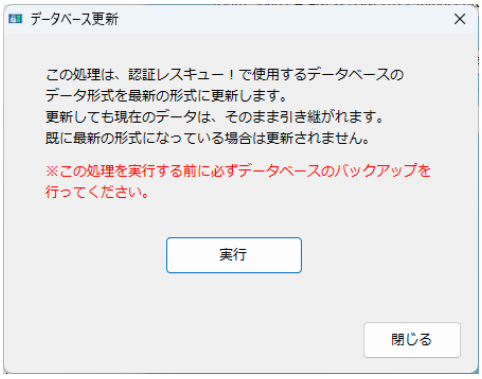
ここでは、必要に応じて行う処理について説明します。

■ログイン設定



ここでは、この環境設定（認証 Web サービス）にログインを使って起動する場合のユーザ名とパスワードを登録します。これらの情報は忘れないようにご注意ください。

■データベース更新



この処理は、認証レスキュー！で使用するデータベースのデータ形式を最新の形式に更新します。更新しても現在のデータは、そのまま引き継がれます。既に最新の形式になっている場合は更新されません。

■データベースコンバート

この処理は、前製品「認証レスキュー！2」のデータベースを本製品「認証レスキュー！.NET」のデータベースに変換します。前製品「認証レスキュー！2」のユーザ様で、現在のデータベースを本製品「認証レスキュー！.NET」でそのまま使用したいお客様がご利用ください。
コンバート元とコンバート先のデータベースを指定して、「コンバートを実行」ボタンを押します。
なお、データベースコンバート後も前製品「認証レスキュー！2」のデータベースはそのまま残ります。

■データベースのバックアップ

この処理は、認証レスキュー！で使用するデータベースをバックアップします。
「バックアップ先ファイル」を指定して「バックアップ開始」ボタンを押すとすぐにバックアップが実行されます。定期的なバックアップを設定するには、「スケジュール化する」をチェックしてバックアップする間隔を指定します。

■データベースの復元

この処理は、バックアップしてあるデータベースを復元します。
「復元元ファイル」を指定して「復元開始」ボタンを押すとすぐに復元がが実行されます。

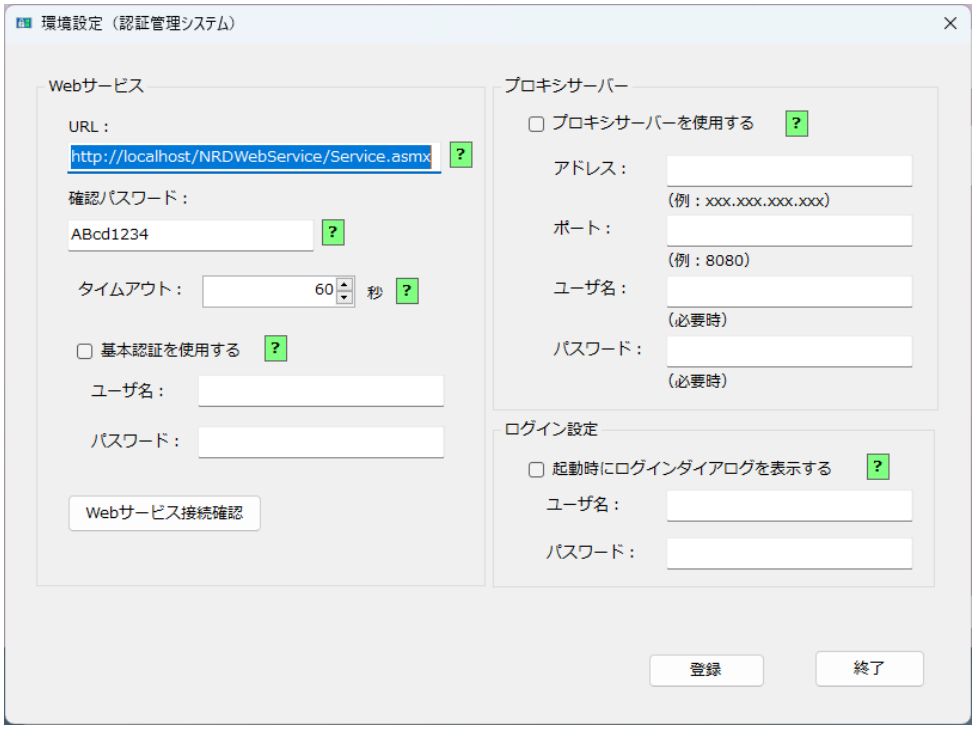
2.認証業務用社内 PC の「認証管理システム」

デスクトップ上の「認証レスキュー！.NET 認証管理システム」へのショートカットを起動すると次の画面が表示されます。
この実行ファイルは、インストール先がデフォルトなら、
〈32bitOS の場合〉 C:¥Program Files¥Newtone¥NRD¥NRDInsideSystem¥InsideSystem.exe、
〈64bitOS の場合〉 C:¥Program Files (x86)¥Newtone¥NRD¥NRDInsideSystem¥InsideSystem.exe
です。

「認証管理システム」のメニューが表示されますが、最初に行うのは「環境設定」処理です。初期状態では他の処理は選択できません。

◆必須設定処理項目
ここでは、設定が必須で省略できない設定について説明します。

■環境設定
「環境設定」ボタンを押すと次のような画面が表示されます。



各項目について説明します。

・Web サービス/URL
認証に関するシステムを Web サービスとして提供する Web サーバーの URL を指定します。
以下にいつくか例を示します。

★自 PC のローカルホストの Web サーバー（IIS）にアクセスする場合：
http://localhost/NRDWebService/Service.asmx

★自社 Web サーバー（IIS）にアクセスする場合：
例えば、URL は
http://www.newtone.co.jp/NRDWebService/Service.asmx
といったものになります。
この、「www.newtone.co.jp」部分が貴社の Web サイトになります。

★クラウドサービス Microsoft Azure の仮想マシン（IIS）を利用する場合：
例えば、URL は
http://172.207.72.171/NRDWebService/Service.asmxx
といったものになります。
この、「172.207.72.171」部分が、貴社の Azure ポータルの「仮想マシン」の「パブリック IP アドレス」に表示されているものとなります。
また、Azure ポータルの「仮想マシン」の「DNS 名」で指定することも可能です。
例えば、DNS 名が「nrvm3-2022-ip.japaneast.cloudapp.azure.com」の場合の URL は
http://nrvm3-2022-ip.japaneast.cloudapp.azure.com/NRDWebService/Service.asmxx
といったものになります。

・Web サービス/確認パスワード

Web サービスを利用する場合の確認用のパスワードを設定します。ここでは、Web サーバー側設定アプリケーション「認証 Web サービス」の環境設定処理の Web サービスの確認パスワードと同じものを設定します。
確認パスワードは必須項目です、省略はできません。
8 文字以上で半角の次の文字が使用できます。
・大文字の英字（A～Z）
・小文字の英字（a～z）
・数字（0～9）
・記号（+-*/!#$%&()=¥@<>?）

・Web サービス/タイムアウト

Web サービスに接続して応答を待つ最大時間を秒単位で指定します。
初期値は 60 秒です。

・Web サービス/基本認証を使用する

Web サーバーで基本認証を使用する場合はチェックを入れます。
初期値は、未チェック（基本認証を使用しない）です。
Web サーバー（IIS）側で特定のアカウント（ユーザ名とパスワード）でアクセスできるフォルダにこの Web サービスが配置してある場合に使用できるセキュリティ設定です。
Web サーバーで基本認証を使用する一般的な手順は次の通りです。
1.サーバーPC 上でユーザの作成を行う。
　この際のユーザ名とパスワードがそのまま基本認証に使われます。
2.基本認証を使用するフォルダのセキュリティ設定を行う。
　フォルダのプロパティを開き、セキュリティタブで上記 1 のユーザ名を追加し、「読み取り」権限を付与します。
3.IIS でのセキュリティ設定
　IIS で該当フォルダに対し「認証」の設定で「匿名認証」を無効にして「基本認証」を有効にします。
なお、Web サーバーでの基本認証の詳細につきましてはマイクロソフト社の関連ドキュメントをご覧ください。

・プロキシサーバーを使用する

インターネットへのアクセス環境としてプロキシサーバーを使用している場合は、チェックを入れて「アドレス」と「ポート」を、更に必要に応じて「ユーザ名」と「パスワード」をそれぞれ指定します。
Web サービスは HTTP 経由で利用されます。その際、環境によってはセキュリティ上の制限から、プロキシサーバーを経由してインターネットにアクセスしなければならない場合があります。
そういった場合、使用しているプロキシサーバーの情報を設定することで問題なくプロキシサーバー経由で Web サーバーの Web サービスにアクセスすることができます。

ここまでの設定をしたら、「Web サービス接続確認」ボタンを押して、Web サービスに接続できることを確認してください。

・ログイン設定

認証管理システムを起動する際に、ログイン画面を表示する場合はチェックを入れて、「ユーザ名」と「パスワード」を指定します。

「環境設定」の各項目の入力が終わったら、「登録」ボタンを押して設定を保存します。設定内容はこの「環境設定」を実行している PC のレジストリに保存されます。

◇認証キーの作成

次は、認証キーを作成します。
次の 5 種類の作成方法がありますが、どれを使用するかは任意ですが、少なくてもどれかの方法で認証キーを作成する必要があります。

・自動ナンバリング
・表形式
・個別
・ランダム生成
・インポート

以降で各処理を説明します。

■認証キー作成（自動ナンバリング）
ここでは自動ナンバリングで作成します。
「認証管理システム」のメニューで「認証キー作成（自動ナンバリング）」ボタンを押します。

次の画面が表示されます。

以下に各項目を説明します。

・プロダクト ID
プロダクト ID は、製品を表わす任意の文字列です。
桁数は運用開始時に設定し、以降は変更できません。
桁数の初期値は 17 桁です。
なお、データベースの選択で「NRD サンプルデータベース」をお試しいただく場合の桁数は 17 桁

固定となり設定できません。
このプロダクト ID には、製品分類やバージョン、ライセンス数などを識別できる桁取りがあると管理しやすくなります。
たとえば、0000A-00002-00001 で製品 A のバージョン 2 の 1 ライセンスを、0000A-00002-00004 で製品 A のバージョン 2 の 4 ライセンスを、それぞれ示すように設定します。
プロダクト ID は、半角の次の文字が使用できます。
・大文字の英字（A～Z）
・小文字の英字（a～z）
・数字（0～9）
・記号（+-*/!#$%&()=¥@<>?）
このプロダクト ID と「シリアル No.」で出荷時の一意の製品を示すキーとなります。

・シリアル No.
プシリアル No.は、同一製品の識別連番を表わす文字列です。
桁数は運用開始時に設定し、以降は変更できません。
桁数の初期値は 8 桁です。
なお、データベースの選択で「NRD サンプルデータベース」をお試しいただく場合の桁数は 8 桁固定となり設定できません。
シリアル No. は、半角の次の文字が使用できます。
・大文字の英字（A～Z）
・小文字の英字（a～z）
・数字（0～9）
・記号（+-*/!#$%&()=¥@<>?）
「プロダクト ID」とこのシリアル No.で出荷時の一意の製品を示すキーとなります。

・上位固定文字列
上位固定文字列は、シリアル No.において任意の上位桁数の文字列を指定する場合に指定します。
たとえば、8 桁中、上位 3 桁を"ABC"と指定した場合は次のようなシリアル番号の自動作成が可能です。
ABC00001
ABC00002
ABC00003
ABC00004
・・・・・

・開始番号
開始番号は、シリアル No.において連番を自動付番する場合の最初の数を指定します。「間隔（ステップ）数」や「ナンバリング数」とともにシリアル No.の自動付番時の必須項目です。
たとえば、8 桁中、上位 3 桁を"ABC"と指定し、開始番号を 1、間隔（ステップ）数を 2、ナンバリング数を 5 とそれぞれ指定した場合、次のようなシリアル番号が自動作成されます。
ABC00001
ABC00003
ABC00005
ABC00007
ABC00009

・間隔（ステップ）数
間隔（ステップ）数は、シリアル No.において連番を自動付番する場合の付番間隔を数で指定します。「開始番号」や「ナンバリング数」とともにシリアル No.の自動付番時の必須項目です。
たとえば、8 桁中、上位 3 桁を"ABC"と指定し、開始番号を 1、間隔（ステップ）数を 2、ナンバリン

グ数を 5 とそれぞれ指定した場合、次のようなシリアル番号が自動作成されます。
ABC00001
ABC00003
ABC00005
ABC00007
ABC00009

・ナンバリング数

ナンバリング数は、シリアル No.において自動付番で作成する（シリアル No.の）数を指定します。「開始番号」や「間隔（ステップ）数」とともにシリアル No.の自動付番時の必須項目です。
たとえば、8 桁中、上位 3 桁を"ABC"と指定し、開始番号を 1、間隔（ステップ）数を 2、ナンバリング数を 5 とそれぞれ指定した場合、次のようなシリアル番号が自動作成されます。
ABC00001
ABC00003
ABC00005
ABC00007
ABC00009

・ライセンス数

ライセンス数は、製品のライセンス数を指定します。
たとえば、1 ライセンスなら 1、5 ライセンスなら 5 と指定します。「認証キー作成」（自動ナンバリング）処理でライセンス数を指定する場合、1 度に作成する自動付番内に異なるライセンスを指定することはできません。
ライセンス数は、「プロダクト ID」や「シリアル No.」などとともに認証キーテーブルの項目のひとつです。

・有効期限設定

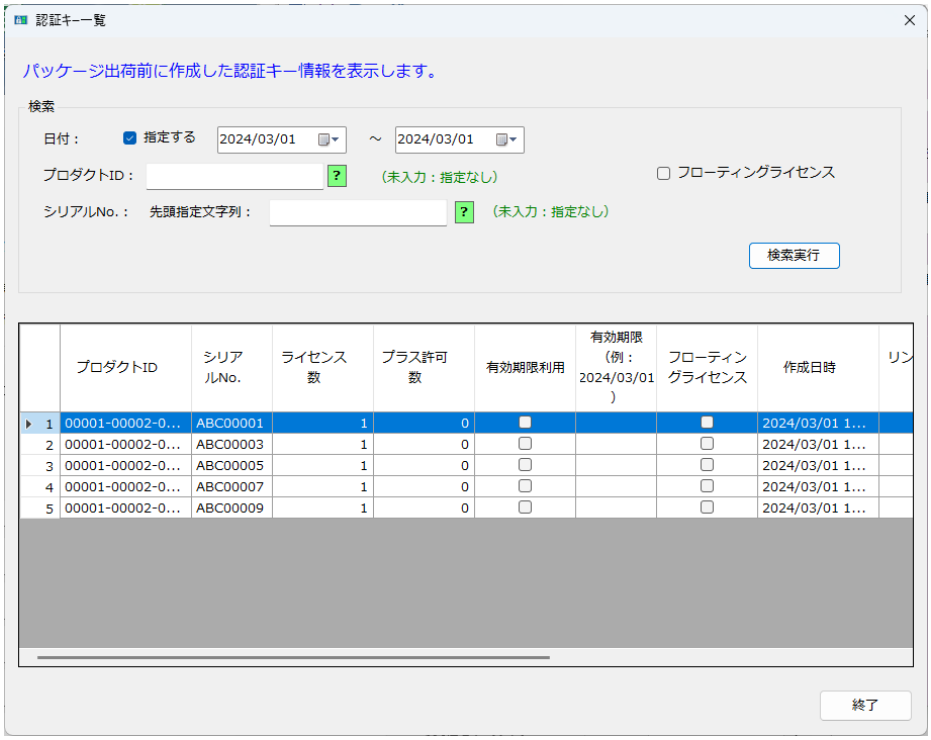
製品の認証登録後、無期限で使用できるライセンスではなく、特定の有効期限まで使用可能とするライセンス形態にする場合はこの設定を行います。
有効期限後も継続して使用可能とする場合の有効期限の再設定は、認証管理システムの「認証キー編集（表形式）」処理を利用します

・フローティングライセンス

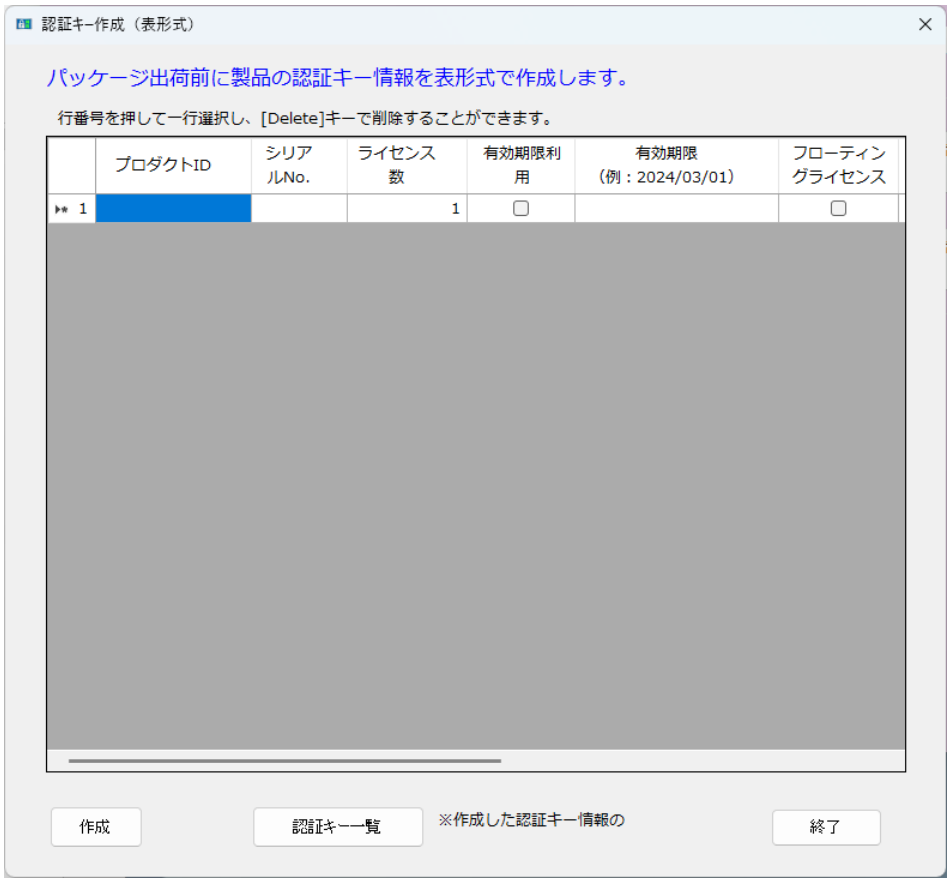
同時に使用できる PC の数を設定する場合に選択します。
例えば、「フローティングライセンス」にチェックを入れて、ライセンス数を 10 としたら、それは貴社のアプリケーションを同時に最大 10 台までの PC でエンドユーザが利用できるライセンスを指します。また、前項の有効期限設定との組み合わせも可能です。例えば、同時に最大 10 台までの PC で 1 年間、といったライセンスも作成できます。

上記の各項目を入力して、「作成」ボタンを押すと認証キーが作成されます。

「認証キー一覧」ボタンを押すと作成した認証キーの一覧が次のように表示されます。


認証キー一覧
パッケージ出荷前に作成した認証キー情報を表示します。
検索
日付：
指定する
2024/03/01
～
2024/03/01
プロダクトID：
（未入力：指定なし）
フローティングライセンス
シリアルNo.：
先頭指定文字列：
（未入力：指定なし）
検索実行
プロダクトID
シリアルNo.
ライセンス数
プラス許可数
有効期限利用
有効期限（例：2024/03/01）
フローティングライセンス
作成日時
リン
1 00001-00002-0… ABC00001 1 0 2024/03/01 1…
2 00001-00002-0… ABC00003 1 0 2024/03/01 1…
3 00001-00002-0… ABC00005 1 0 2024/03/01 1…
4 00001-00002-0… ABC00007 1 0 2024/03/01 1…
5 00001-00002-0… ABC00009 1 0 2024/03/01 1…
終了

■認証キー作成（表形式）



パッケージ出荷前に製品 1 本毎の認証キー情報を表形式で一度に複数作成できます。

「作成」ボタン:
表に入力された内容で、＜認証キーテーブル＞に追加します。

「認証キー一覧」ボタン:
登録されている＜認証キーテーブル＞の一覧を検索条件を指定して表示します。

■認証キー作成（個別）

パッケージ出荷前に製品 1 本毎の認証キー情報を作成します。

「作成」ボタン：
入力された内容で、＜認証キーテーブル＞に 1 レコード追加します。

「認証キー一覧」ボタン：
登録されている＜認証キーテーブル＞の一覧を検索条件を指定して表示します。

■認証キー作成（ランダム生成）

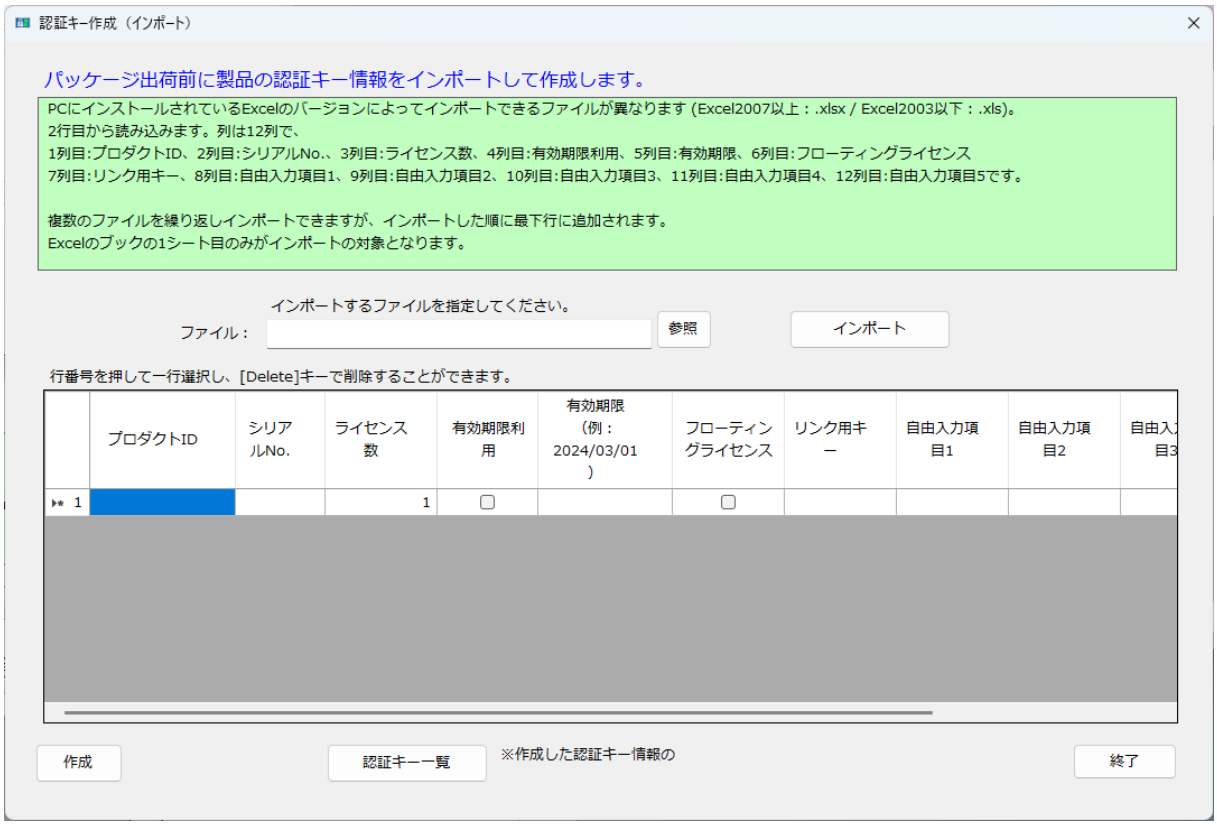
パッケージ出荷前に製品の認証キー情報を指定した数だけランダムに自動生成します。

「上位固定文字列」の指定や「ランダム指定」の選択ができます。

「作成」ボタン：
入力された内容で、＜認証キーテーブル＞に追加します。

「認証キー一覧」ボタン：
登録されている＜認証キーテーブル＞の一覧を検索条件を指定して表示します。

■認証キー作成（インポート）



パッケージ出荷前に、製品の認証キー情報をインポートして作成します。

PC にインストールされている Excel のバージョンによってインポートできるファイルが異なります（Excel2007 以上：.xlsx ／ Excel2003 以下：.xls）。

2 行目から読み込みます。列は 12 列で、
1 列目:プロダクト ID、2 列目:シリアル No.、3 列目:ライセンス数、4 列目:有効期限利用、5 列目:有効期限、6 列目:フローティングライセンス、7 列目:リンク用キー、8 列目:自由入力項目 1、9 列目:自由入力項目 2、10 列目:自由入力項目 3、11 列目:自由入力項目 4、12 列目:自由入力項目 5 です。

複数のファイルを繰り返しインポートできますが、インポートした順に最下行に追加されます。
Excel のブックの 1 シート目のみがインポートの対象となります。

「参照」ボタン：
インポートするファイルを選択します。

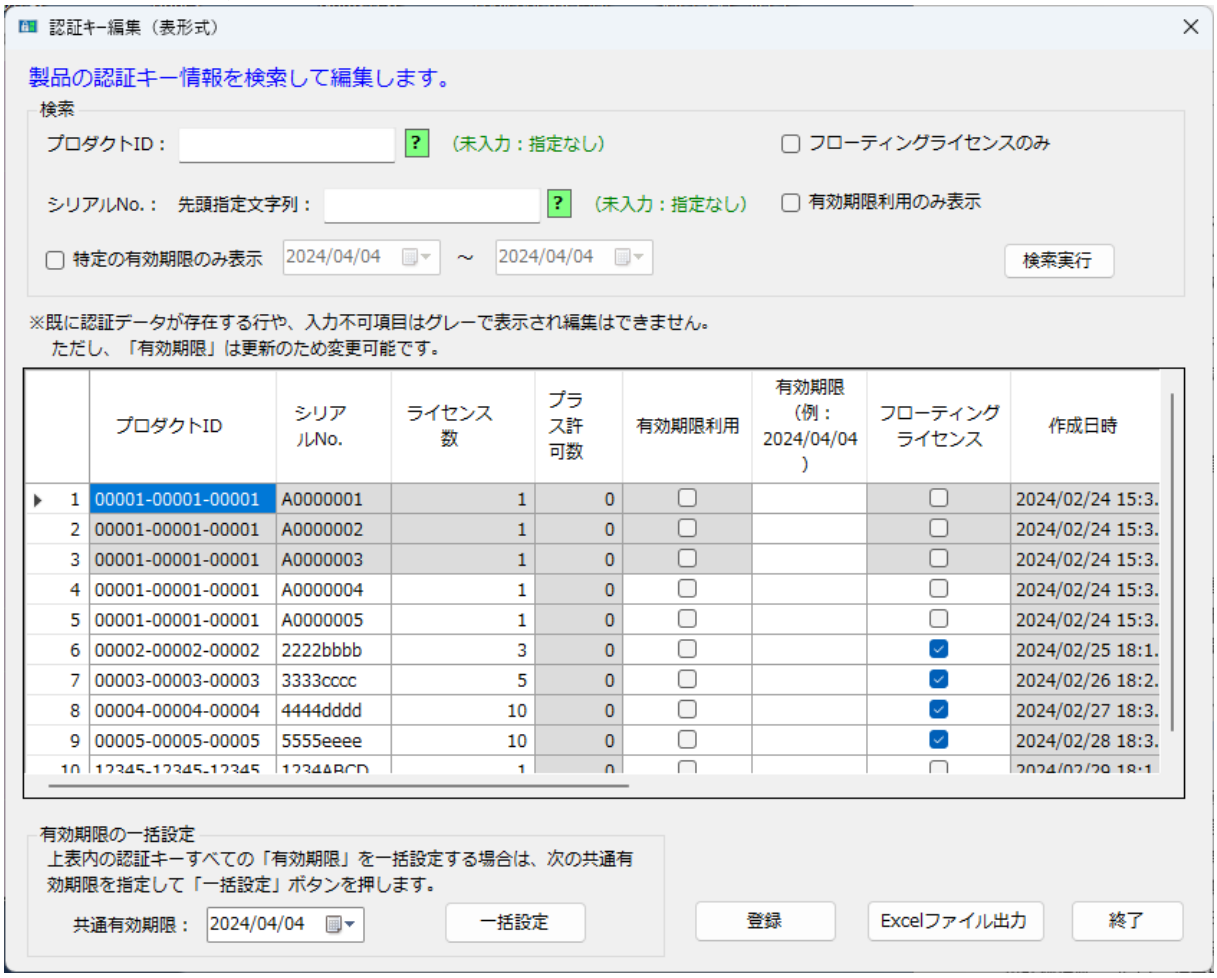
「インポート」ボタン：
インポートを実行します。

「作成」ボタン：
インポートされた内容で、＜認証キーテーブル＞に追加します。

「認証キー一覧」ボタン：
登録されている＜認証キーテーブル＞の一覧を検索条件を指定して表示します。

◆任意設定処理項目
ここでは、必要に応じて行う処理について説明します。

■認証キー編集（表形式）



既に存在する認証キーを表形式で編集します。

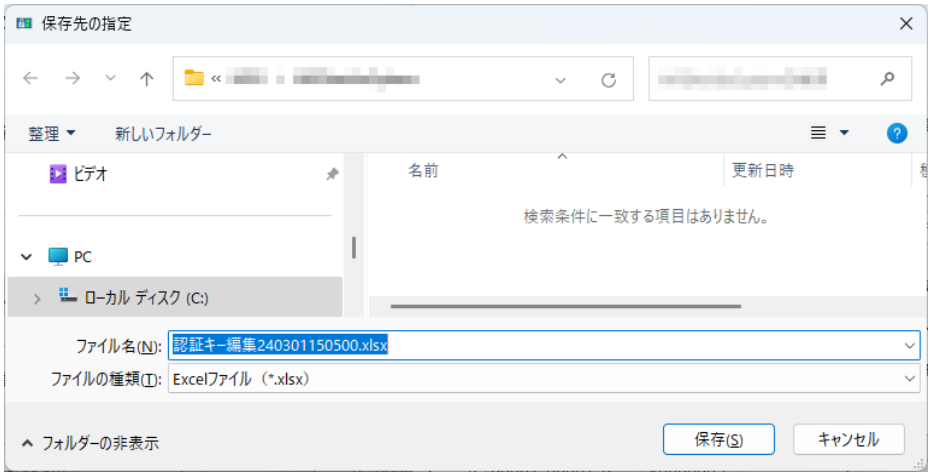
※ライセンス形態が有効期限形式の場合で、特定のプロダクト ID とシリアル No に新しい有効期限を設定する場合は、当処理を利用します。

「検索実行」ボタン：
入力された検索条件で、＜認証キーテーブル＞の該当するレコードを表示します。
検索結果に対して表上で編集できます。
その際、既に（子データである）認証データが存在する認証キーの行や、入力不可項目はグレーで表示され編集はできません。

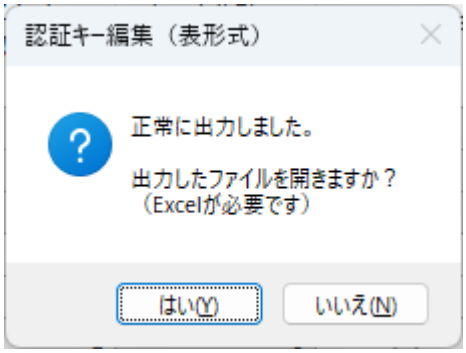
「登録」ボタン：
表に入力された内容で、＜認証キーテーブル＞を更新します。

「有効期限の一括設定」
現在表示されている表内の認証キーすべての「有効期限」を一括で同日に設定できます。

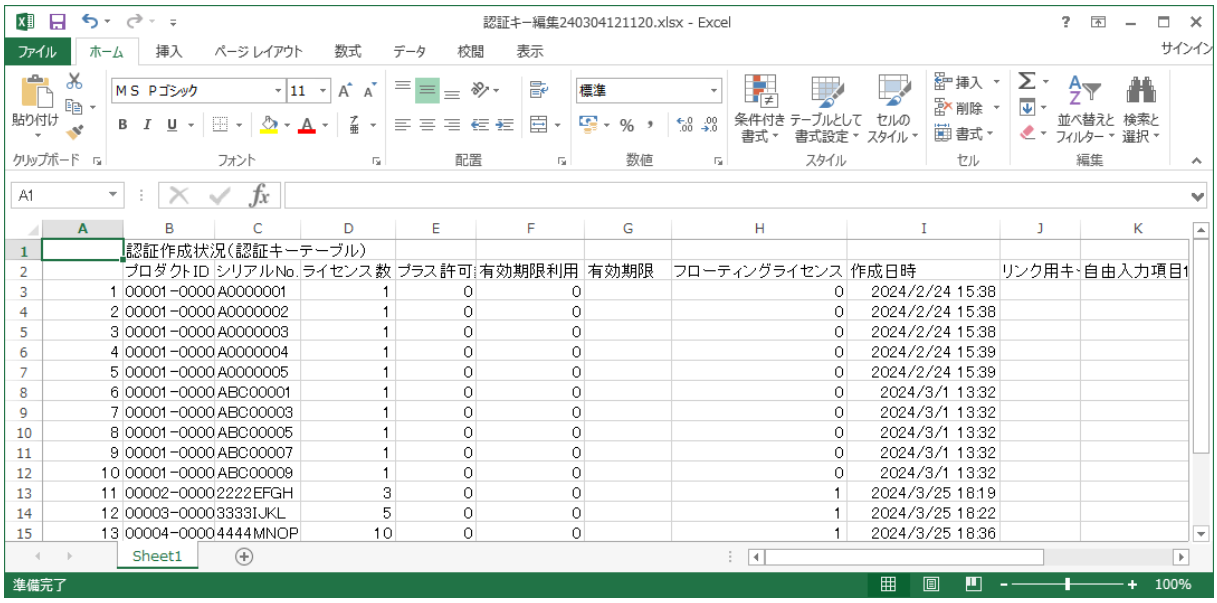
「Excel ファイル出力」ボタン：
現在の表示内容を Excel ファイル（.xlsx）として出力します。ボタンを押すと出力先を指定するダイアログが表示されます。「保存」ボタンを押します。



出力した Excel ファイルを開くか選択します。

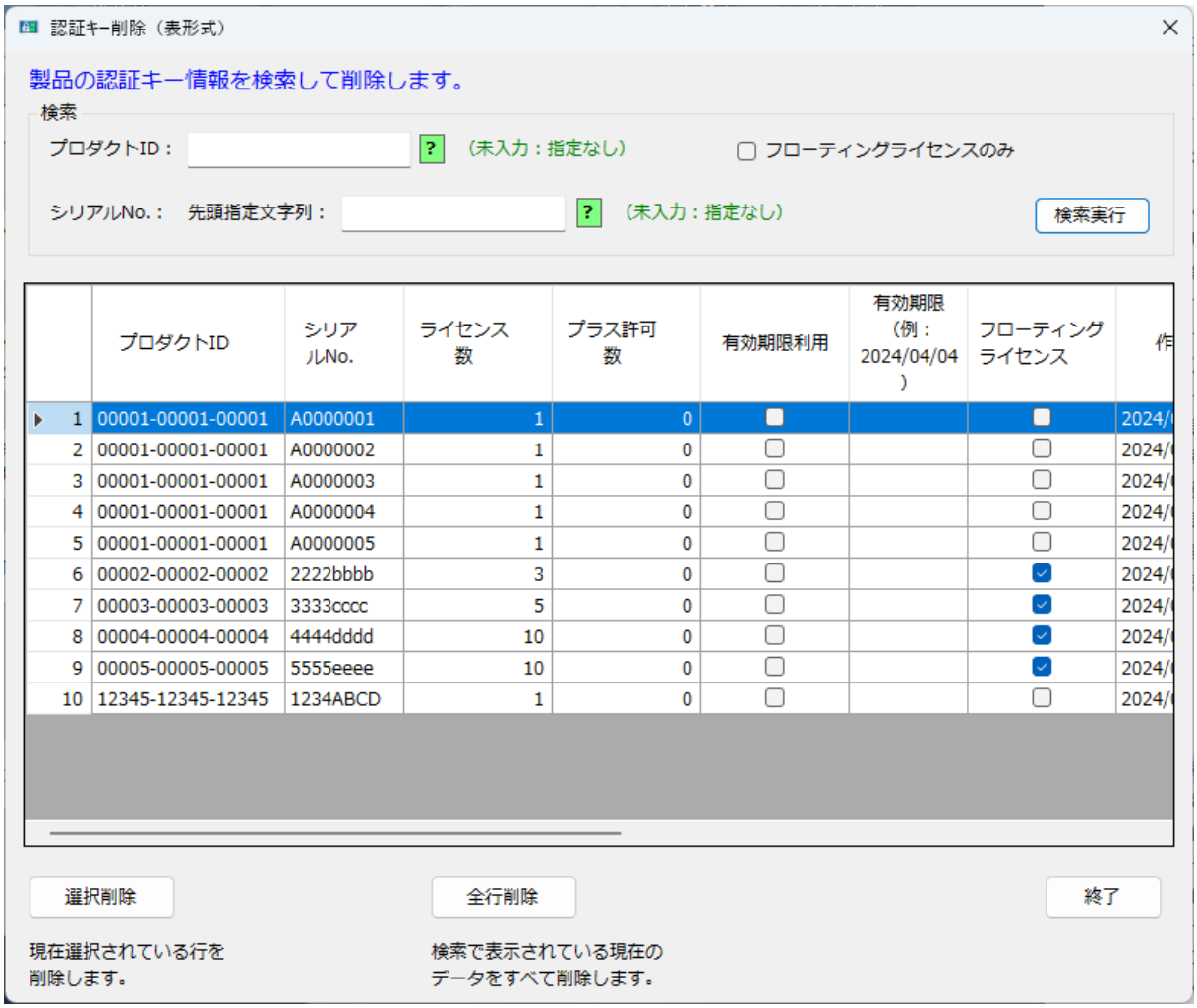


出力した Excel ファイルの例：



	A	B	C	D	E	F	G	H	I	J	K
1		認証作成状況(認証キーテーブル)									
2		プロダクトID	シリアルNo.	ライセンス数	プラス許可	有効期限利用	有効期限	フローティングライセンス	作成日時	リンク用キー	自由入力項目1
3	1	00001-0000	A0000001	1	0	0		0	2024/2/24 15:38		
4	2	00001-0000	A0000002	1	0	0		0	2024/2/24 15:38		
5	3	00001-0000	A0000003	1	0	0		0	2024/2/24 15:38		
6	4	00001-0000	A0000004	1	0	0		0	2024/2/24 15:39		
7	5	00001-0000	A0000005	1	0	0		0	2024/2/24 15:39		
8	6	00001-0000	ABC00001	1	0	0		0	2024/3/1 13:32		
9	7	00001-0000	ABC00003	1	0	0		0	2024/3/1 13:32		
10	8	00001-0000	ABC00005	1	0	0		0	2024/3/1 13:32		
11	9	00001-0000	ABC00007	1	0	0		0	2024/3/1 13:32		
12	10	00001-0000	ABC00009	1	0	0		0	2024/3/1 13:32		
13	11	00002-0000	2222EFGH	3	0	0		1	2024/3/25 18:19		
14	12	00003-0000	3333IJKL	5	0	0		1	2024/3/25 18:22		
15	13	00004-0000	4444MNOP	10	0	0		1	2024/3/25 18:36		

■認証キー削除（表形式）

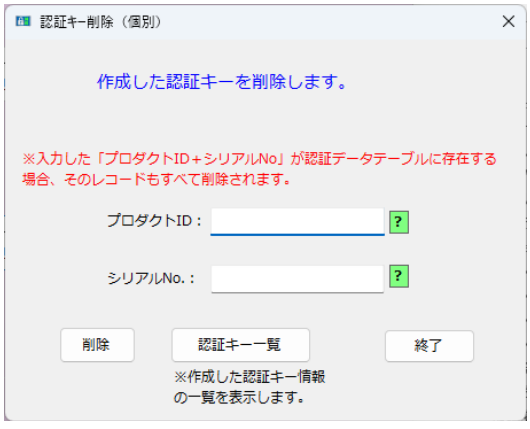


「検索実行」ボタン：
入力された検索条件で、＜認証キーテーブル＞の該当するレコードを表示します。
検索結果に対して表上で削除する行を指定します。

「選択削除」ボタン：
現在選択されている行を削除します。複数行の選択も可能です。

「全行削除」ボタン：
検索結果として表示されている現在のデータをすべて削除します。

■認証キー削除（個別）

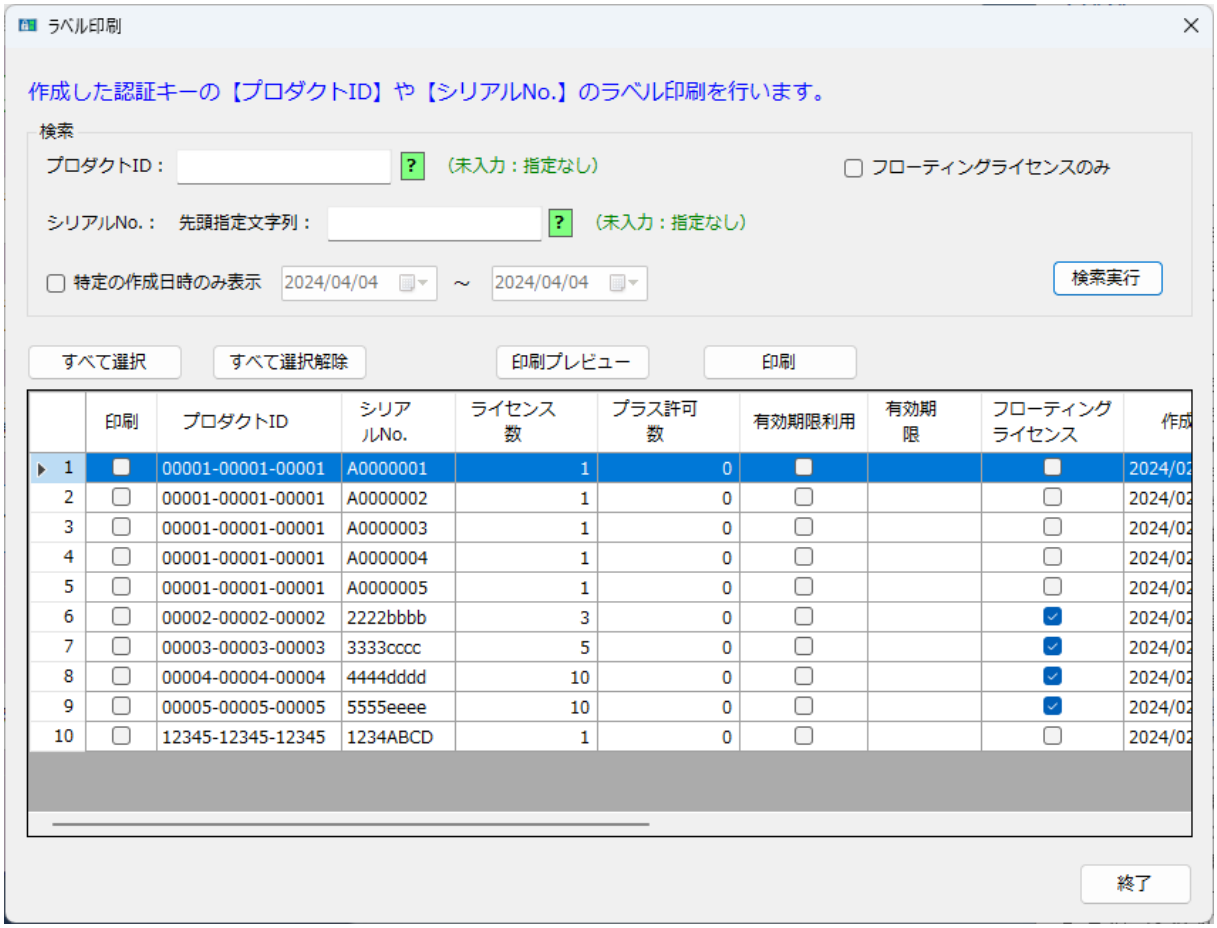


既存の認証キーを＜認証キーテーブル＞から削除します。
この際、入力した「プロダクト ID+シリアル No.」が（子データの）＜認証データテーブル＞に存在する場合は、そのレコードも削除されます。

「削除」ボタン：
入力された内容の認証キーを削除します。

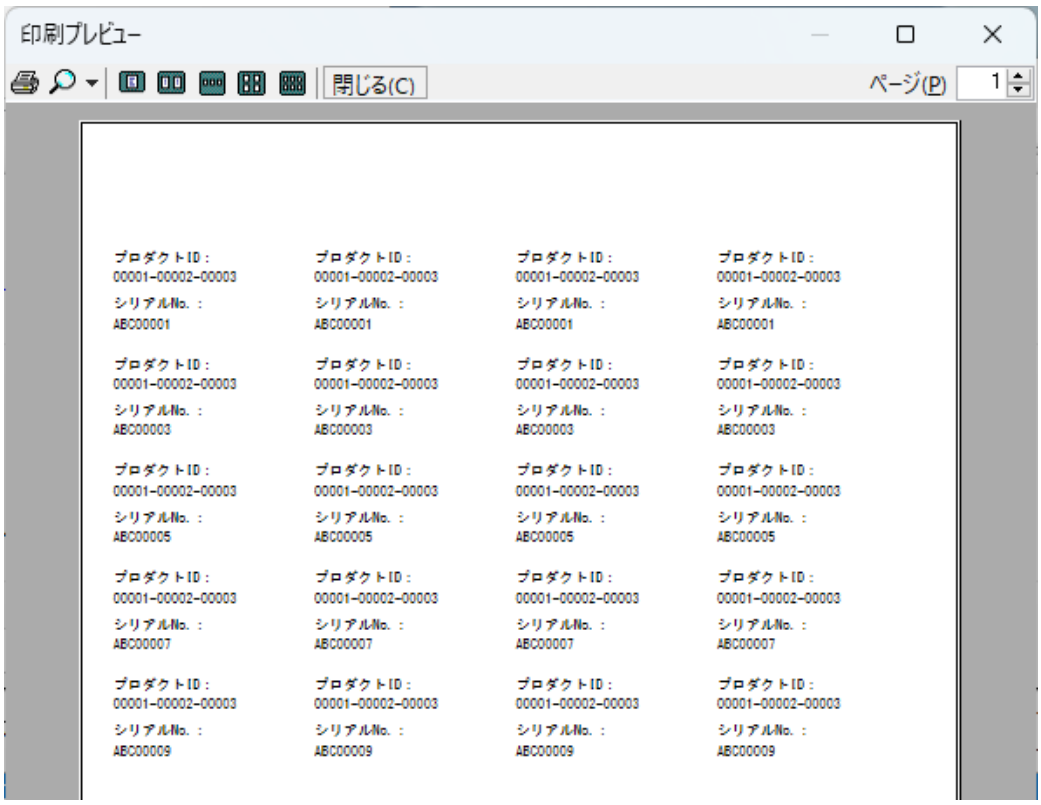
「認証キー一覧」ボタン：
登録されている＜認証キーテーブル＞の一覧を検索条件を指定して表示します。

■ラベル印刷
次に、お客様（エンドユーザ）にアプリケーションを配布するために、前工程で作成した認証キーからプロダクト ID とシリアル No.のラベルシールを印刷します。
「認証管理システム」のメニューで「ラベル印刷」ボタンを押します。



	印刷	プロダクトID	シリアルNo.	ライセンス数	プラス許可数	有効期限利用	有効期限	フローティングライセンス	作成
1	☐	00001-00001-00001	A0000001	1	0	☐		☐	2024/02
2	☐	00001-00001-00001	A0000002	1	0	☐		☐	2024/02
3	☐	00001-00001-00001	A0000003	1	0	☐		☐	2024/02
4	☐	00001-00001-00001	A0000004	1	0	☐		☐	2024/02
5	☐	00001-00001-00001	A0000005	1	0	☐		☐	2024/02
6	☐	00002-00002-00002	2222bbbb	3	0	☐		☑	2024/02
7	☐	00003-00003-00003	3333cccc	5	0	☐		☑	2024/02
8	☐	00004-00004-00004	4444dddd	10	0	☐		☑	2024/02
9	☐	00005-00005-00005	5555eeee	10	0	☐		☑	2024/02
10	☐	12345-12345-12345	1234ABCD	1	0	☐		☐	2024/02

ここでは、必要に応じて検索条件を入力して「検索実行」ボタンを押します。
表示された認証キーの一覧に対して印刷する対象にチェックをつけます。
「印刷プレビュー」ボタンを押すと次のような印刷プレビューが表示されます。



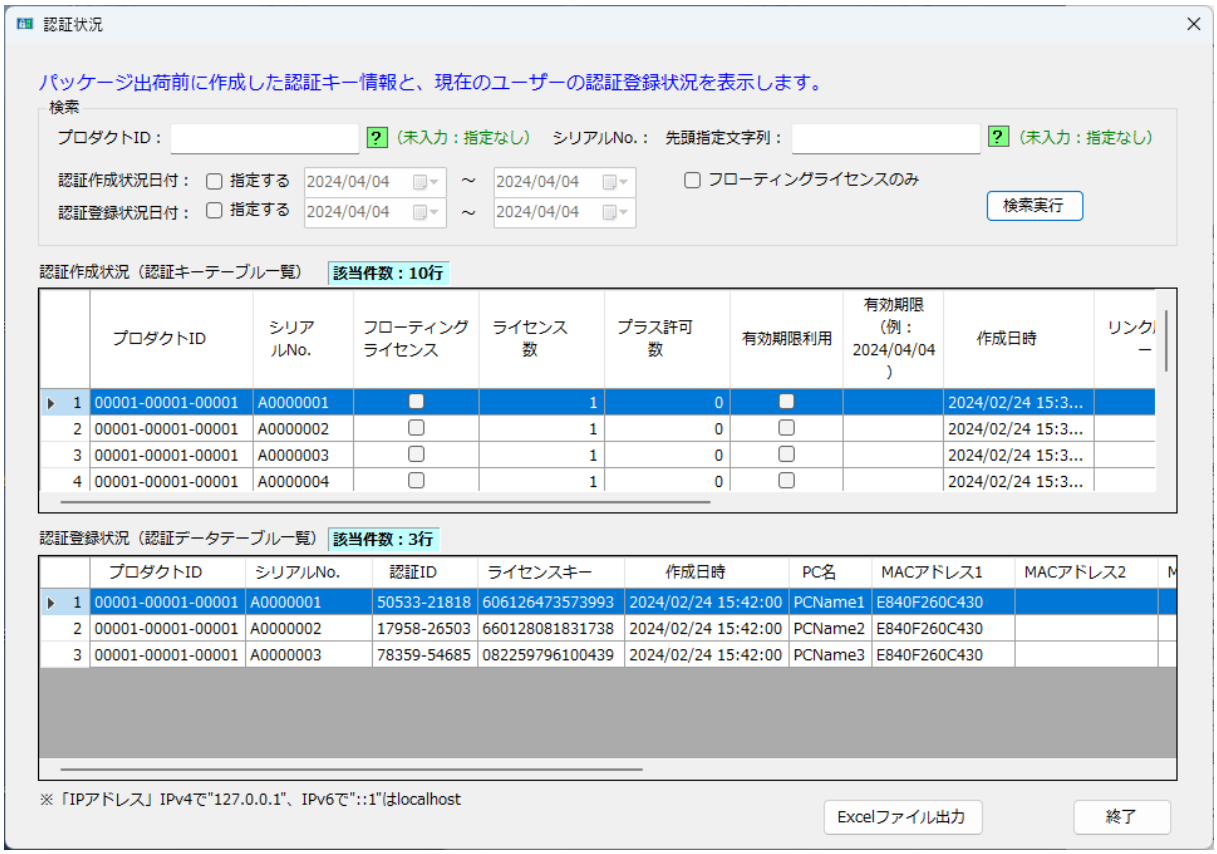
ラベルシールは、このようにプロダクト ID とシリアル No. が横にそれぞれ 4 枚ずつ印刷されます。
その 1 行分にはすべて同じ内容が印刷されます。
奇数行目のプロダクト ID と次の行の偶数行目のシリアル No.で対となります。

同じラベルシールが 4 枚あるのは、それぞれの用途を想定しています。たとえば、次のような場合です。
1 枚目：パッケージ内の DVD ジャケット（ケース）添付（貼付）用
2 枚目：パッケージ内のユーザ登録用紙添付用
3 枚目：出荷指示書添付用
4 枚目：注文書控え添付用

・用紙の仕様

このラベルシールの印刷用紙は、一般的な市販のラベルシール用紙を想定しています。
具体的には、次の仕様です。
用紙サイズ：A4
ラベル数：横 4×縦 9＝36 面
ラベルサイズ：45.7×25.4mm
余白：上 31.8mm/左 10.2mm/右 9.5mm/下 36.6mm
メーカ例：ヒサゴ「A4 タックシール」GB871 など

■認証状況

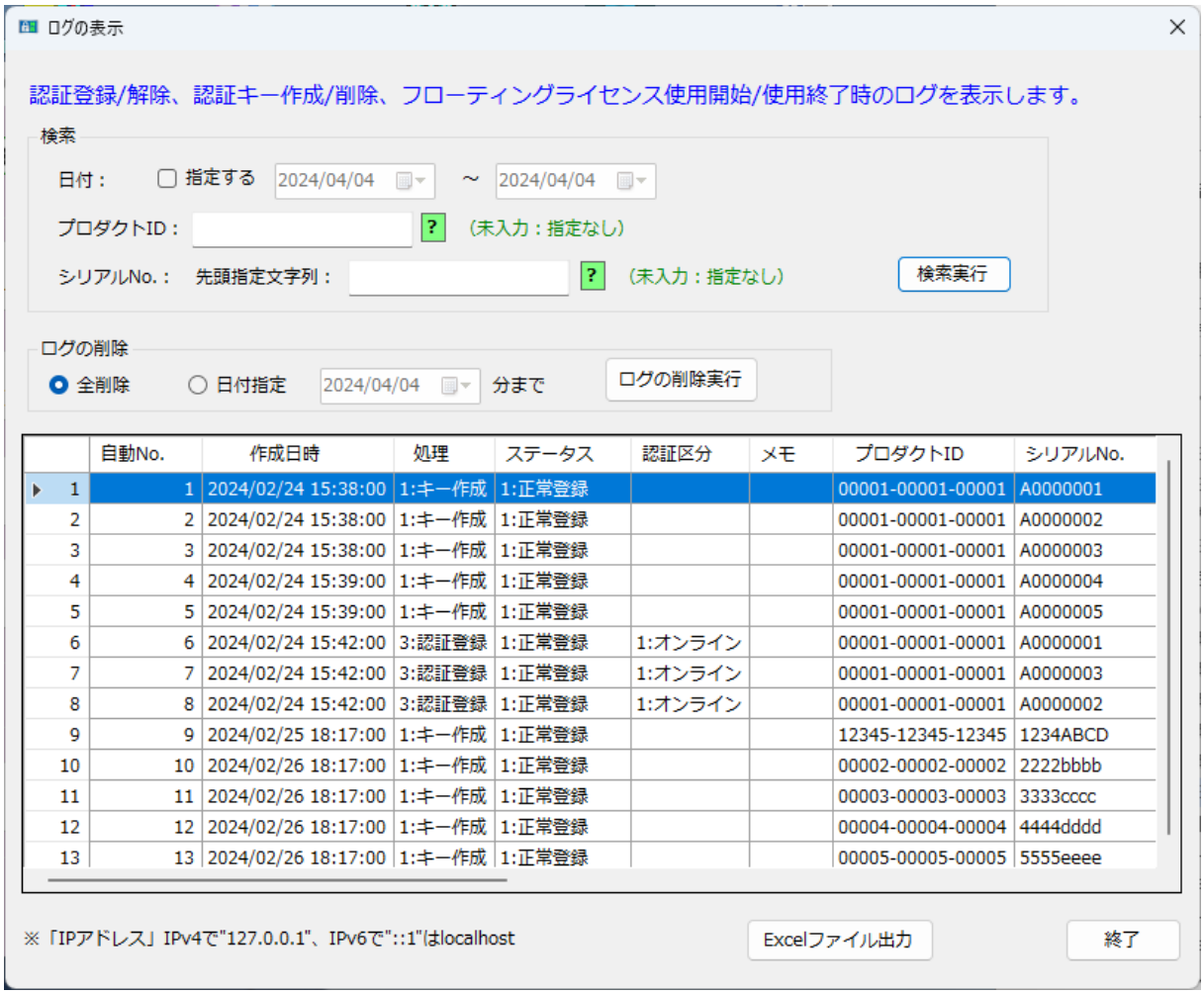


パッケージ出荷前に作成した認証キー情報（上表）と現在のエンドユーザによるライセンスの認証登録状況（下表）を表示します。

「検索実行」ボタン：
入力された検索条件で、該当するデータを表示します。

「Excel ファイル出力」ボタン：
現在の表示内容を Excel ファイル（.xlsx）として出力します。

■ログの表示



認証登録、認証解除、認証キー作成、認証キー削除時のログを表示します。また、ログの削除もできます。

「検索実行」ボタン：
入力された検索条件で、該当するデータを表示します。

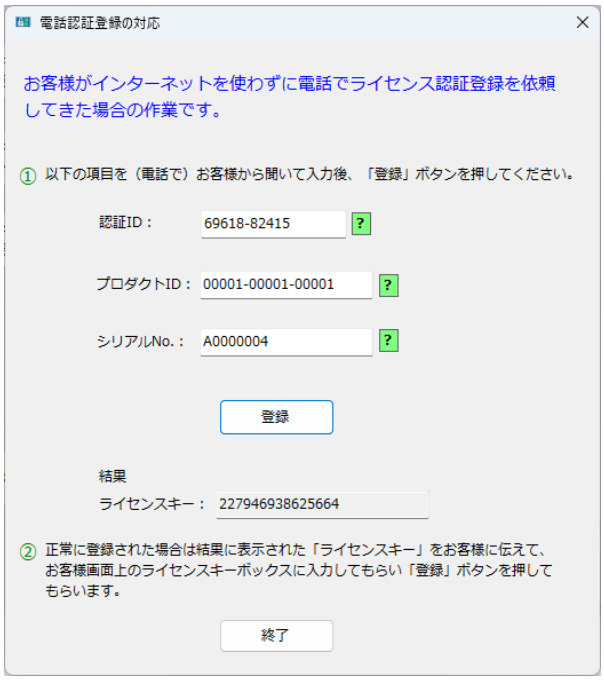
「ログの削除実行」ボタン：
削除する条件を、「全削除」か「日付指定」から選択し、「日付指定」の場合はいつまでのログを削除するかを年月で指定後、この「ログの削除実行」ボタンを押すと指定した条件のログデータが削除されます。

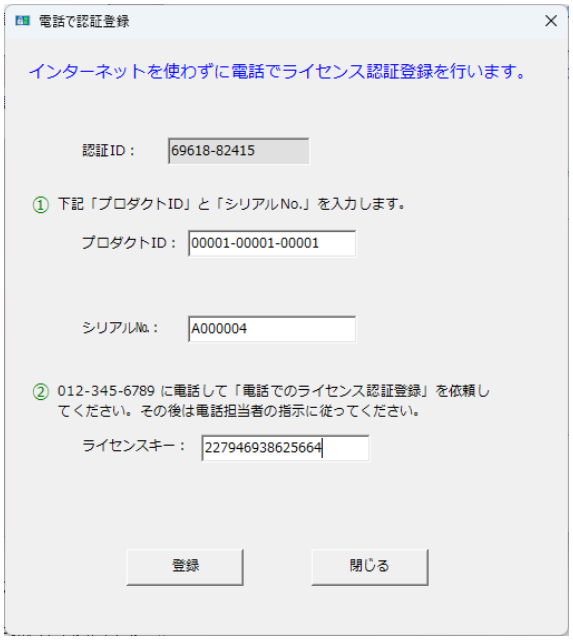
「Excel ファイル出力」ボタン：
現在の表示内容を Excel ファイル（.xlsx）として出力します。

出力した Excel ファイルを開くか選択します。

出力した Excel ファイルの例：

■電話認証登録の対応





本処理（貴社の認証業務用 PC）の画面　　エンドユーザ PC 上の画面

エンドユーザがインターネットを使わずに電話でライセンス認証登録を依頼してきた場合の作業です。

上記の項目を電話でユーザから聞いて入力し、「登録」ボタンを押します。
その後、表示されたライセンスキーをエンドユーザに伝えて、エンドユーザ画面上の「ライセンスキーボックス」に入力してもらい「登録」ボタンを押してもらいます。

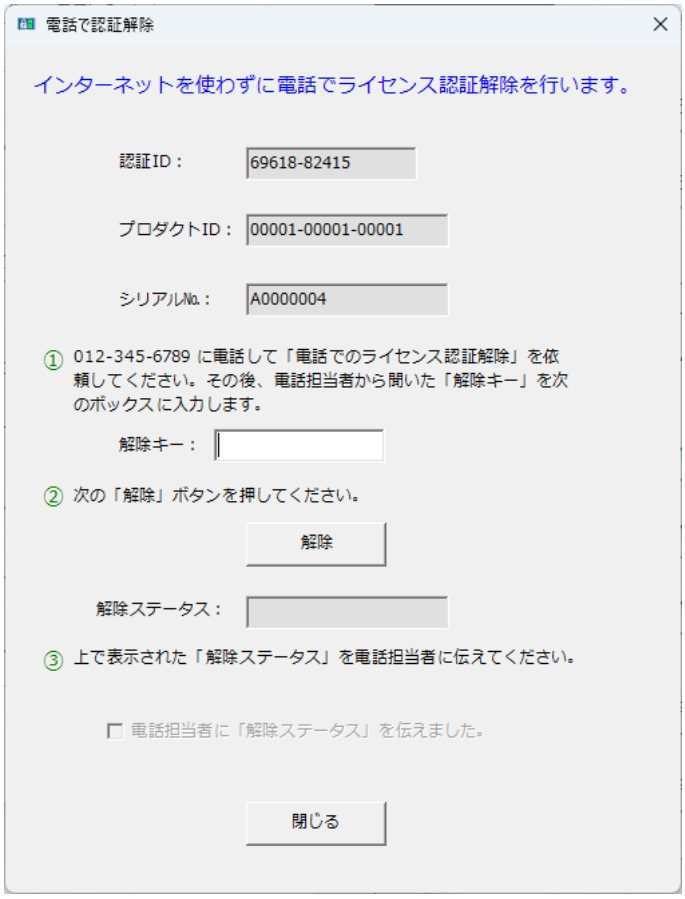
※上図のように貴社の認証業務用の PC とエンドユーザ用の PC の画面を並べて表示すると、ユーザとの電話対応がイメージできます。

■電話認証解除の対応

◇オフライン（エンドユーザの PC が動作している場合）

本処理（貴社の認証業務用 PC）の画面

エンドユーザ PC 上の画面



エンドユーザがインターネットを使わずに電話でライセンス認証解除を依頼してきた場合の作業です。

上記の項目を電話でエンドユーザから聞いて入力し、「解除キーの表示」ボタンを押します。
その後、表示された解除キーをユーザに伝えて、エンドユーザ画面上の「解除キーボックス」に入力してもらい「解除」ボタンを押してもらいます。エンドユーザが正常に解除できたか確認したら社内用 PC 上の「解除」ボタンを押します。

「解除キー」、「解除ステータス」について

この解除キーと解除ステータスは、この処理でしか使用しません。また、データベースやエンドユーザ PC のレジストリにも保存しません。

解除キーはインターネット以外で認証解除をする場合に、必ず貴社に電話をかけさせるための手段として用意されています。

「電話認証解除の対応」の画面の「オフライン（ユーザの PC が動作している場合）」を見ると分かりますが、この解除キーの入力が無いとすると、「解除」ボタンを押すだけでエンドユーザが勝手に PC 上で認証解除ができてしまいます。この場合、エンドユーザ PC 上は認証解除状態になっても、貴社のデータベース上は認証解除にはなりません。情報のやり取りがないのですから当たり前です。そのままでは、再度エンドユーザが認証登録をしなおした場合に貴社のデータベースは解除されていないので登録は拒否されます。

そこでこのようなトラブルを避けるため、オフラインでの認証解除にはこの解除キーを必要とし、エンドユーザは貴社へ電話をして解除キーを聞かなければいけない仕組みになっているのです。

また、解除ステータスはエンドユーザがこの「電話で認証解除」を利用した不正ライセンスの流用防止のために使われます。

◇クラッシュ（エンドユーザの PC が動作不能になった場合）

エンドユーザの PC がクラッシュしてしまい、OS などを入れなおした PC や別の PC に貴社のパッケージを再インストールする場合で、そのために以前の認証解除をしなければいけない状況への対応処理です。

この場合、エンドユーザはパッケージに添付されているプロダクト ID とシリアル No.しか分かりません。

この処理では、上記の項目を電話でエンドユーザから聞いて入力後、「解除」ボタンを押します。

前述の「＜認証キーテーブル＞の「プラス許可数」項目の必要性」に書きましたが、この処理によりデータベースには新たに認証登録できる猶予が作成されます。

●有効期限機能の利用方法

認証レスキュー！では、貴社がお客様（エンドユーザ）に対して通常の認証登録による無期限のライセンスとは別に、有効期限によるライセンスを設定することができます。
以下にその基本的な利用手順を示します。

1.貴社側作業①　-　認証キー作成時に有効期限を設定する

「認証管理システム」内の認証キー作成処理には、自動ナンバリング、表形式、個別、ランダム生成、インポートの 5 種類があります。
これらの認証キー作成処理を使用して出荷前に最初の有効期限を設定します。
各処理の画面には指定したプロダクト ID とシリアル No.に対する「有効期限利用（する）」と「有効期限」の項目があります。
有効期限を設定するには、「有効期限利用（する）」をチェックして「有効期限」項目に有効期限としたい日付を設定します。

＜認証キー作成（個別）処理の画面例＞

上記の画面例では 5 ライセンス（マルチライセンス）に対して共通な有効期限を設定しています。

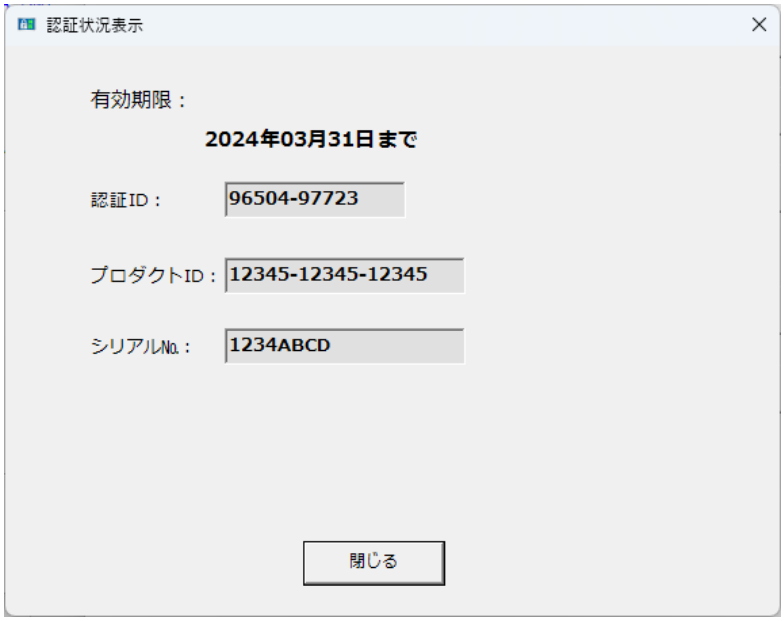
2.エンドユーザ側作業①　-　認証登録を行う

エンドユーザに配布された貴社のアプリケーションから、通常の有効期限のない認証登録と同様に、認証 UI ライブラリ（DLL）を呼び出すことにより認証登録を実行してもらいます。

認証登録・解除の方法としてインターネットでの認証登録・解除、電話での認証登録・解除、代理認証登録・解除の 3 種類がありますが、**有効期限によるライセンスの場合は「電話での認証登録・解除」は利用できませんので、ご注意ください。**

＜「インターネットで認証登録」の画面例＞

＜貴社のアプリケーションから「認証状況表示」を呼び出した場合の画面例＞
下図のように、貴社が出荷前に「認証管理システム」の「認証キー作成」処理で設定した有効期限がエンドユーザの PC にも設定されます。



3.貴社側作業②　-　有効期限更新に備え、新しい有効期限を設定する

有効期限によるライセンスで使用していたエンドユーザの有効期限が迫ってきていて、貴社とエンドユーザの間で継続して使用するライセンスを更新する契約が合意されたとします。
その場合、貴社はエンドユーザが使用しているプロダクトIDとシリアルNo.に対し新しい有効期限を設定する必要があります。
「認証管理システム」の「認証キー編集（表形式）」処理で新しい有効期限を設定します。

＜認証キー編集（表形式）の画面例＞

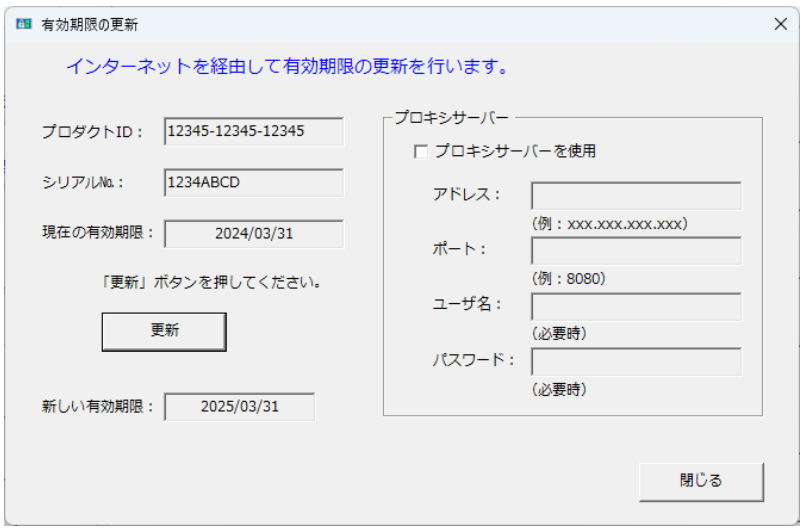
上図の例では、プロダクト ID「12345-12345-12345」、シリアル No.「1234ABCD」のライセンスに対して、2024/03/31 までだった有効期限を、新しく 2025/03/31 までと、一年間更新しています。
この処理後は、貴社はエンドユーザに対し有効期限を延長したことを何らかの形で通知します。

4.エンドユーザ側作業②　-　有効期限の更新を行う

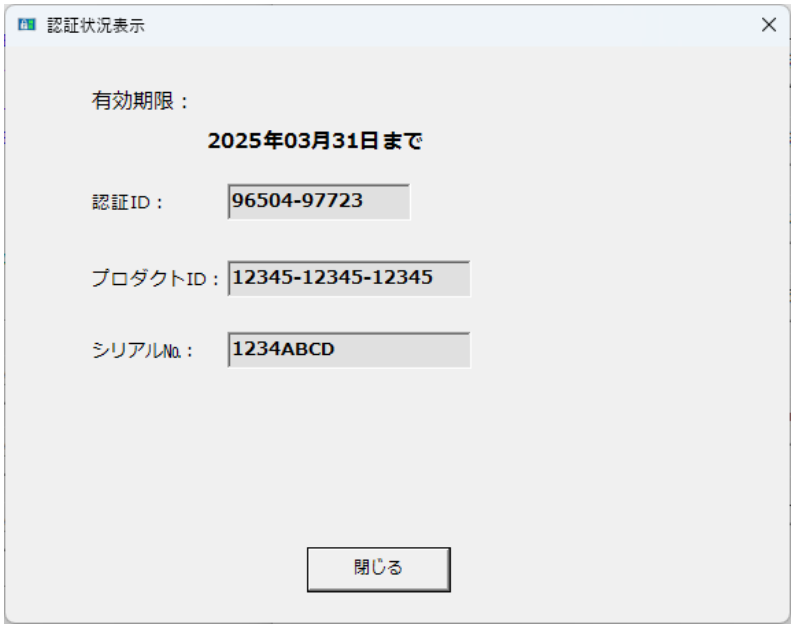
エンドユーザは、貴社からの有効期限延長の通知を受け取ります。
次にエンドユーザに、貴社のアプリケーションから認証 UI ライブラリ（DLL）を呼び出すことにより「有効期限の更新」を実行してもらいます。

この処理は、エンドユーザ PC が有効期限による認証済みでない場合は、実行することはできません。

＜「有効期限の更新」の画面例＞
下図のように、プロダクト ID、シリアル No.、現在の有効期限が自動的に表示されます。
エンドユーザが「更新」ボタンを押すと、貴社が「認証管理システム」の「認証キー編集（表形式）」処理で設定した新しい有効期限が「新しい有効期限」として表示されます。



＜貴社のアプリケーションから「認証状況表示」を呼び出した場合の画面例＞
下図のように、新しい有効期限がエンドユーザの PC にも設定されます。



◆Web アプリケーション（ASP.NET）から有効期限の更新を行う

ASP.NET（Web アプリケーション）用の DLL には Web ページから「有効期限の更新」を実行できる、APIxEditOfExpirationDate メソッドが用意されています。それを利用すれば、貴社が作成した Web ページの UI からエンドユーザに「有効期限の更新」を行わせることができます。

具体的な利用方法は、次の ASP.NET（Web アプリケーション）用のサンプルプロジェクトをご覧ください。
・SampleProject_Web フォルダ（ASP.NET 系サンプルプロジェクト）

●フローティングライセンスの利用方法

フローティングライセンスは貴社のアプリケーションを同時に使用できる最大の PC 台数を設定するライセンスです。

エンドユーザは、最大 PC 台数以内であればどの PC でも貴社のアプリケーションを使用できます。このライセンス形態は、エンドユーザの PC が常時インターネットに接続できる環境が必要ですが、世界中のどこにある PC でも、同じフローティングライセンス内での利用が可能です。

以下にその基本的な利用手順を示します。

■貴社側作業　-　認証キー作成処理時にフローティングライセンスを設定する

「認証管理システム」内の認証キー作成処理には、自動ナンバリング、表形式、個別、ランダム生成、インポートの 5 種類があります。
これらの認証キー作成処理を使用して出荷前にフローティングライセンスを設定します。
各処理の画面には「フローティングライセン」の項目があります。
フローティングライセンスを設定するには、「ライセンス数」を指定して「フローティングライセンス」をチェックします。

＜認証キー作成（個別）処理の画面例＞

上記の画面例では 10 ライセンスのフローティングライセンスを設定しています。

■エンドユーザ側作業

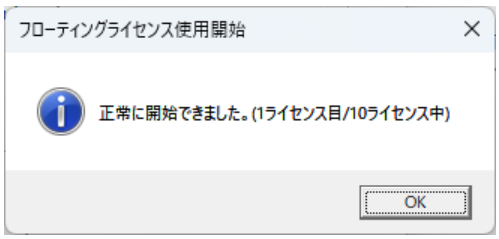
エンドユーザに配布された貴社のアプリケーションから、認証 UI ライブラリ（DLL）を呼び出すことによりエンドユーザに以下の処理を実行してもらいます。

エンドユーザ（お客様）側で行う、フローティングライセンスについては、次の 2 つの処理があります。

1.フローティングライセンス使用開始

貴社のアプリケーションの使用を開始する際に実行します。この処理を実行すると、フローティングライセンスを 1 ライセンス使用することになります。
最初の使用開始だけ、プロダクト ID とシリアル No.を入力します。
次回の使用開始からは必要な項目は自動的に表示されるので、「使用開始」ボタンを押すだけです。

貴社の設定ライセンス数以内で、正常にフローティングライセンスの使用が開始されると次のようなメッセージが表示されます。



この状態で、「認証状況オンライン表示」を実行すると次のように表示されます。

上の画面例では 10 ライセンス中の（自分の PC を含む）2 ライセンスが使用されていることを表しています。

2.フローティングライセンス使用終了

貴社のアプリケーションの使用を終了します。「使用終了」ボタンを押します。
その PC で使用を終了しないとフローティングライセンスを 1 ライセンス使ったままになりますので、注意が必要です。
次回、貴社のアプリケーションを使用するには再度、先の「使用開始」を実行します。

また、この PC のフローティングライセンスからの登録を解除する場合は「解除」ボタンを押します。この解除はこの PC でそのフローティングライセンスを使用する可能性がなくなった場合に実行してください。

●Microsoft Azure で認証レスキュー！を利用する方法

ここでは、「認証レスキュー！」をマイクロソフト社のクラウドサービス Microsoft Azure で利用する方法について説明します。

「認証レスキュー！」を Microsoft Azure で利用するには、Azure の仮想マシンを利用します。

この仮想マシンを利用すると、オンプレミスの自社サーバーへのインストールとほぼ同様の手順で「認証レスキュー！」の利用開始ができます。

《参考》Azure の仮想マシンのコストの例 （2024/02/29 時点）

・構成例

場所：Japan East
インスタンス：
標準-B2s (2vcpu、4GB RAM) x 730 時間 (従量課金制)、Windows (ライセンス込み)、OS のみ
マネージド ディスク：E10（128GB、Standard SSD）

・1 ケ月あたりの平均コスト（小数点以下切り上げ）

仮想マシン：	6,727 円
マネージドディスク：	1,418 円
合計：	8,145 円

マイクロソフト社の Azure を使用するための推定時間単位または月単位のコストを計算する「料金計算ツール」の URL は次の通りです。
https://azure.microsoft.com/ja-jp/pricing/calculator/

■Microsoft Azure の仮想マシンを利用する

まず、貴社がマイクロソフト社に申込み、Microsoft Azure の「仮想マシン」機能が使用できる状態が必要です。Microsoft Azure の申込みなどにつきましては、マイクロソフト社の情報をご覧ください。ここでは、貴社がすでに Microsoft Azure の「仮想マシン」を作成できる状態を前提としています。

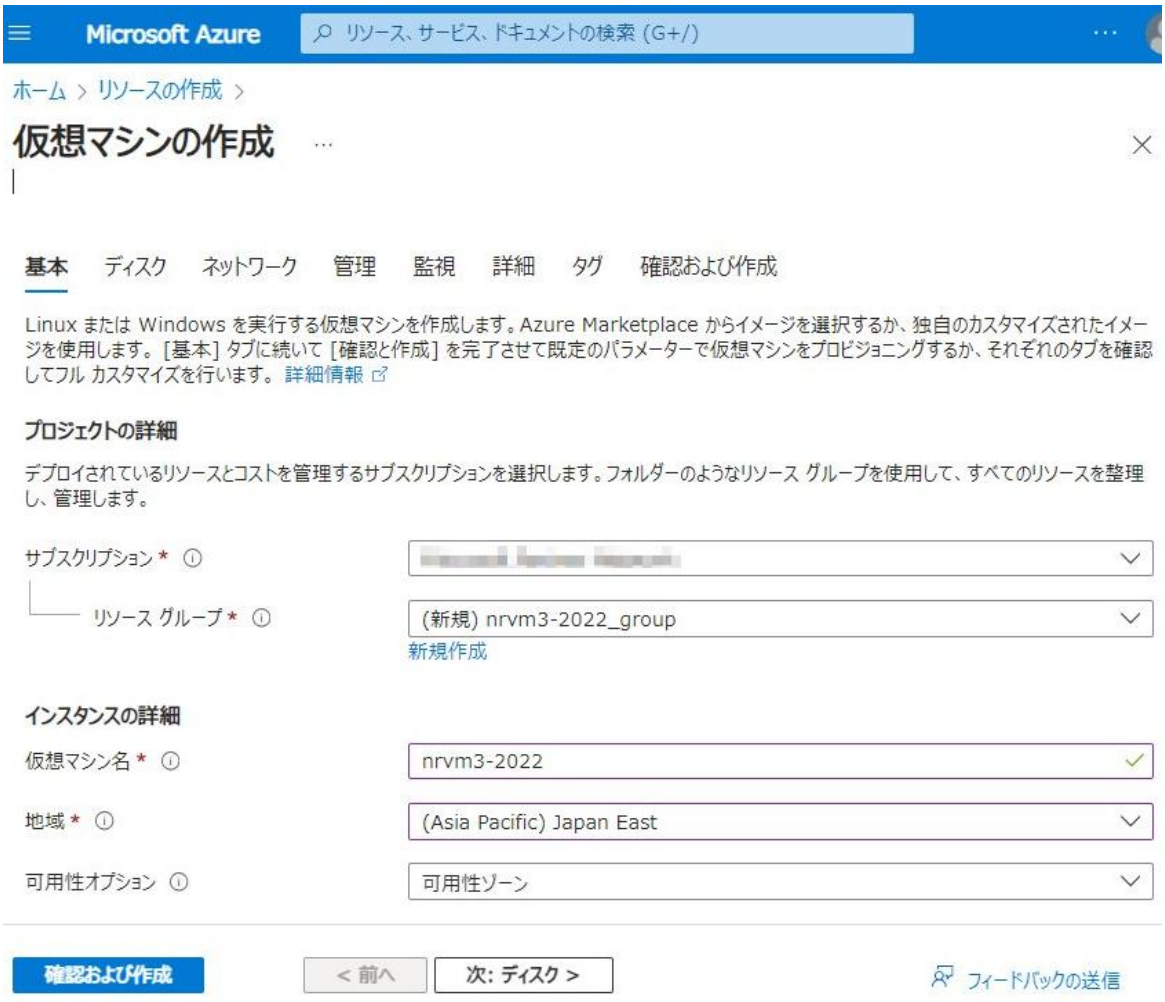
1.仮想マシンの作成

次のマイクロソフト社のサイト「クイック スタート:Azure Portal で Windows 仮想マシンを作成する」も参考にされてください。
https://learn.microsoft.com/ja-jp/azure/virtual-machines/windows/quick-create-portal

以降は、Microsoft Azure のポータルサイトでの操作の例です。

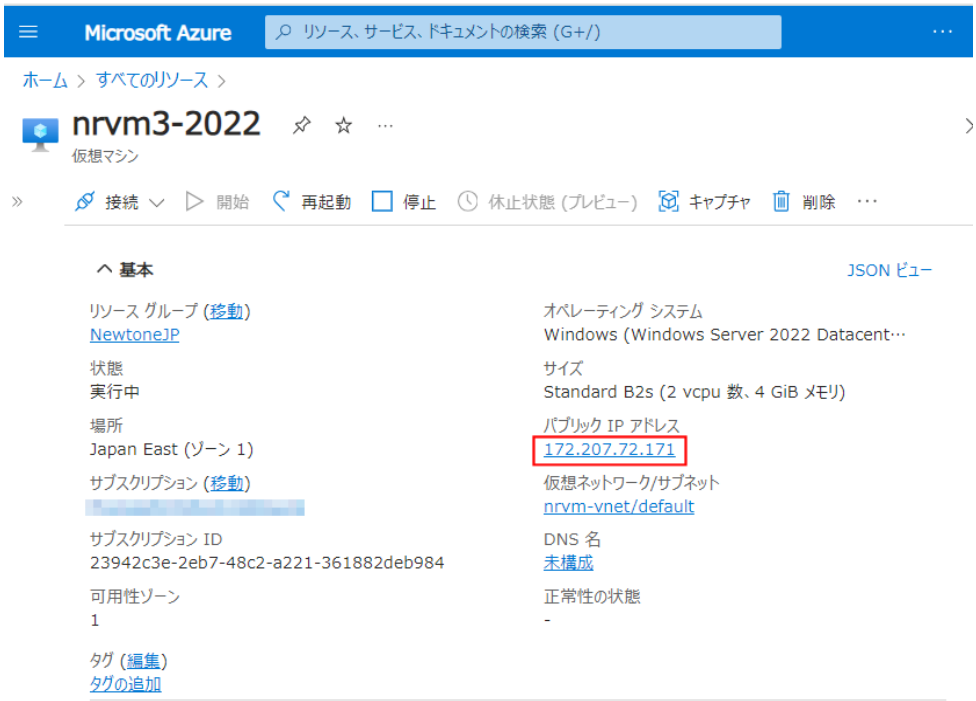
「リソースの作成」で仮想マシンの作成を選択します。

「仮想マシンの作成」で、必要事項を入力・選択して、「作成」ボタンを押します。
その際に、特段の理由がなければ、「地域」（仮想マシンの場所）項目は日本（Japan East/West）を選択することをお勧めします。例えば、デフォルトはアメリカ（米国東部など）となっていますが、日本からのアクセスだとやはり時間がかかります。



2.作成した仮想マシン

作成した仮想マシンの「概要」です。



この中で、認証レスキュー！にとって重要な項目があります。
上図の赤枠で囲んだ「172.207.72.171」はパブリック IP アドレスです。これは、仮想マシンを作成すると自動的に割り当てられるパブリック（グローバル）IP アドレスです。これは、認証レスキュー！の認証管理システムの「環境設定」や DLL のプロパティに設定してこの仮想マシンにアクセスするためのインターネット上の住所となります。

3.仮想マシンのパブリック IP アドレスと DNS 名

前述のパブリック IP アドレス「172.207.72.171」は、下図のように Web サービスの URL の一部となります。
下図は認証管理システムの「環境設定」の例です。

なお、この URL に設定するのが、パブリック IP アドレスではなく、次のように仮想マシンの「DNS 名」で設定することもできます。

例えば次図のように、DNS 名が、「nrvm3-2022-ip.japaneast.cloudapp.azure.com」なら、URL は
http://nrvm3-2022-ip.japaneast.cloudapp.azure.com/NRDWebService/Service.asmx
となります。

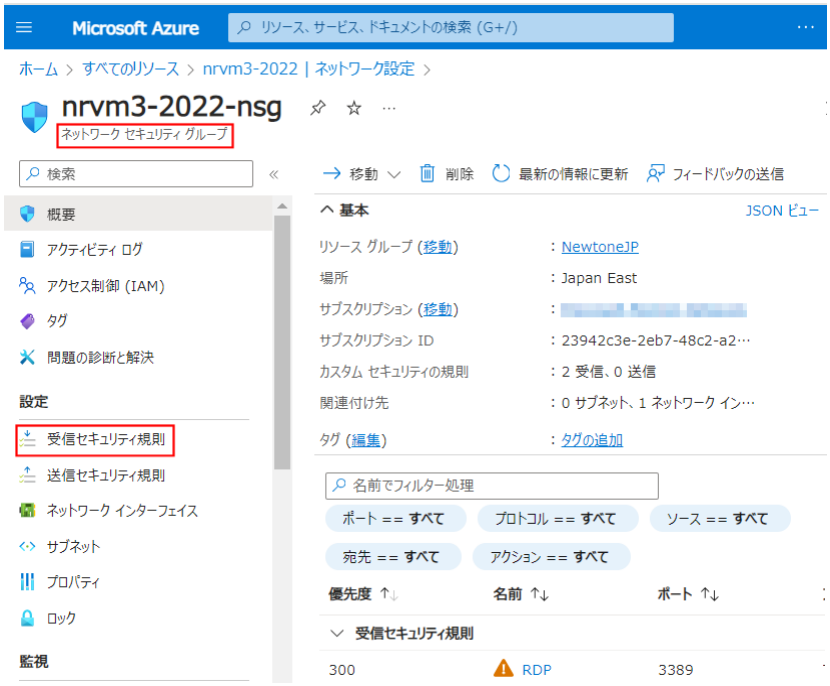
また。貴社で独自のドメインを割り当てることもできます。その場合は、貴社のドメインの DNS サーバーにこの仮想マシンのパブリック IP アドレスを設定する必要があります。

また、http ではなく https の SSL サイトとしてアクセスさせる場合は、仮想マシンのIISに SSL サーバー証明書をインストールする必要があります。

4.仮想マシンの受信セキュリティの規則の追加

次に、仮想マシンに外部から認証レスキュー！の Web サービスに接続させるための設定を行います。

仮想マシンの「ネットワークセキュリティグループ」で左メニューの「受信セキュリティの規則」を選択します。



受信セキュリティの規則の画面で「+追加」を押します。

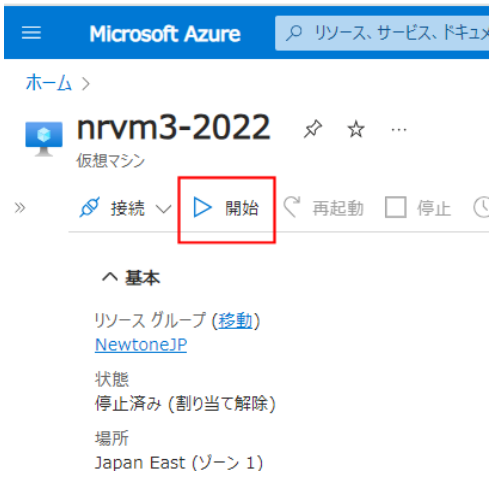
次図のように項目を設定して「保存」ボタンを押します。
HTTPS

受信セキュリティの規則が追加されました。

これで、認証レスキュー！の Web サービスで使うためのポート 80（HTTP）が設定されました。
また、Web サービスの通信をポート 443（HTTPS）で行うこともできます。その場合、ここでの受信セキュリティの規則の追加のほかに、一般的には、仮想マシンの IIS で、あらかじめ取得した SSL サーバー証明書をバインディングする必要があります。

5.仮想マシンの開始

仮想マシンを開始します。「開始」を選択して、しばらく待ちます。
なお、通常は仮想マシンを、開始により起動している間はマイクロソフト社により課金されます。
初期のセットアップや動作確認などで使用する場合は適時、「停止」により仮想マシンを「停止済み(割り当て解除)」としておくのがよいでしょう。



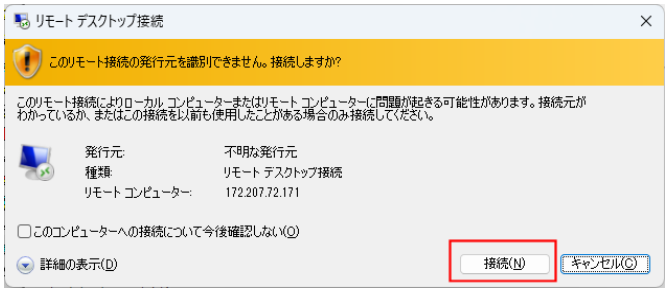
6.仮想マシンの接続

仮想マシンに接続します。接続タブメニューの「接続」を選択します。

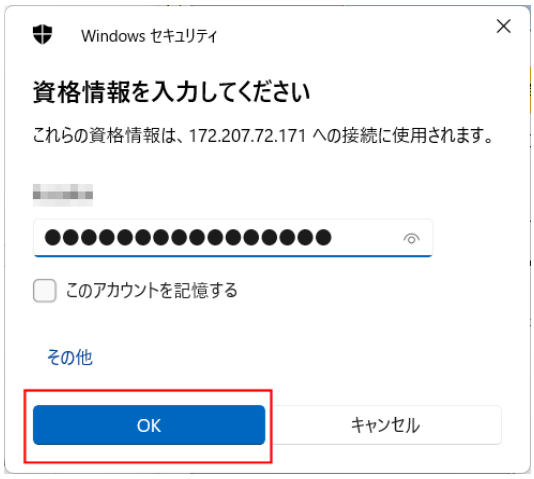
次に、ここでは、接続の方法として RDP（リモート デスクトップ）を選択します。

「RDP ファイルのダウンロード」をクリックして、次ののダイアログで開くアイコンを選択します。

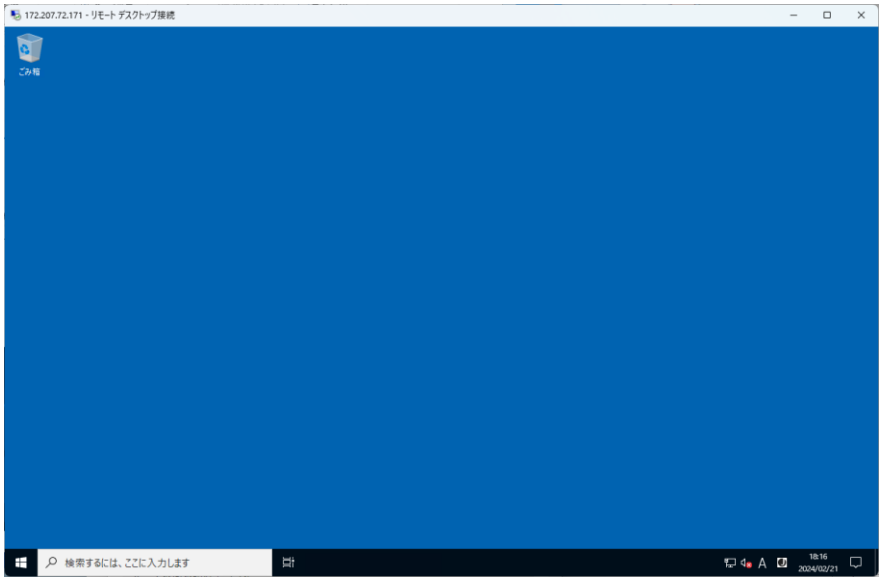
次のダイアログで「接続」を選択します。



次のダイアログで、仮想マシンの管理者ユーザー名のパスワードを入力して、「OK」を押します。。



仮想マシン（この場合、Windows Server 2022）のデスクトップが表示されます。



Azure の仮想マシンはデフォルトで英語（米国）ですが、上の画面は言語を日本語、タイムゾーンや時間など日本に設定した後のものです。
次の手順を参考にして、それらを日本語化にしてから、後述の「仮想マシンへの認証レスキュー！の Web サーバー用 PC へのインストール」を行うことをお勧めします。

7.仮想マシンの日本語化の手順

Azure 仮想マシンの日本語化などの一般的な手順の例（Windows Server 2022）は、次の通りです。

(1) ログイン: Azure ポータルにログインします。

(2) 仮想マシンの選択: 左側のナビゲーションメニューから「仮想マシン」を選択し、設定を変更したい仮想マシンを選択します。

(3) RDP 接続: 仮想マシンに RDP 経由で接続します。

(4) 言語と地域の設定:
- スタートメニューを開き、「Settings（設定）」を選択します。

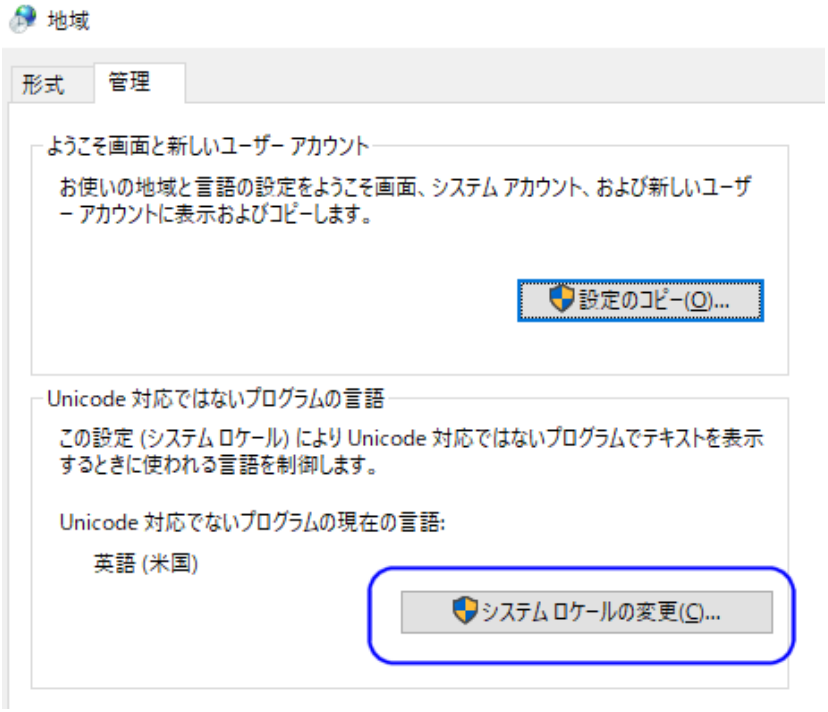
- 「Time & Language（日時と言語）」をクリックします。
- 「Region」タブを選択し、表示されるリストから「Japan」を選択します。

(5) 日付と時刻の設定:
- 「Date & Time（日付と時刻）」タブを選択します。
- 「Time zone（タイムゾーン）」で「(UTC+09:00) Osaka, Sapporo, Tokyo」を選択します。

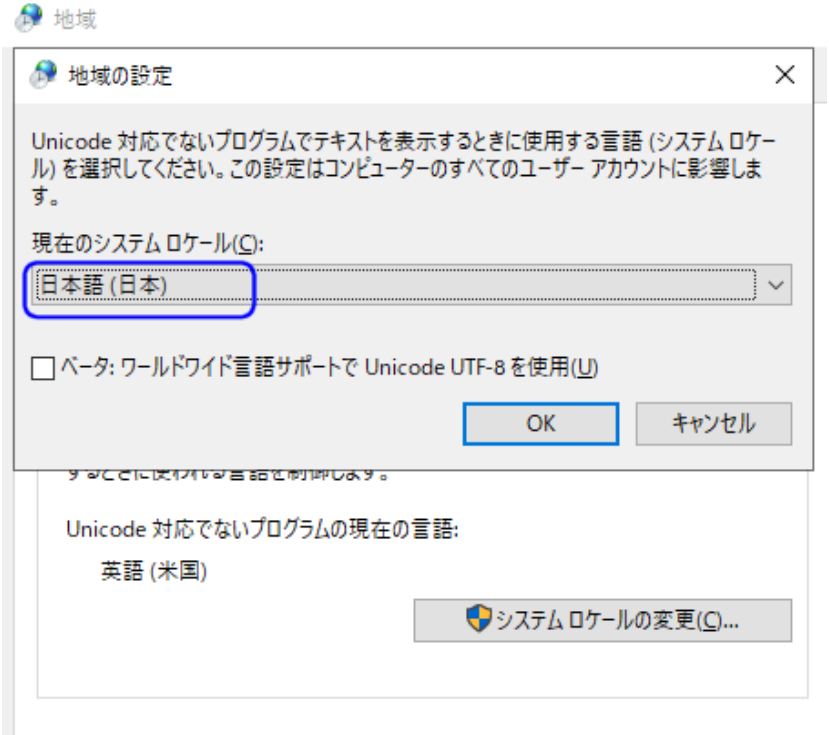
(6) 言語の設定:
- 「Language（言語）」タブを選択します。
- 「Preferred languages（優先言語）」をクリックし、「Add a language（言語を追加）」を選択します。
- 表示されるリストから「Japanese（日本語）」を選択し、「Next（次へ）」をクリックします。
- 「日本語 （日本） - [ja-JP]」を選択し、「Next（次へ）」をクリックします。
- オプションでデフォルトの表示言語を変更するかどうかを選択し、「Install（インストール）」をクリックします。
- その後、再起動が必要な場合は再起動します。

(7) 管理用言語の設定:
- 「言語」タブで「管理用の言語の設定」を選択します。

- 「管理」タブで「システムロケールの変更」を選択します。



- 「地域の設定」で「現在のシステムロケール」を「日本語（日本）」に設定して「OK」をクリックします。



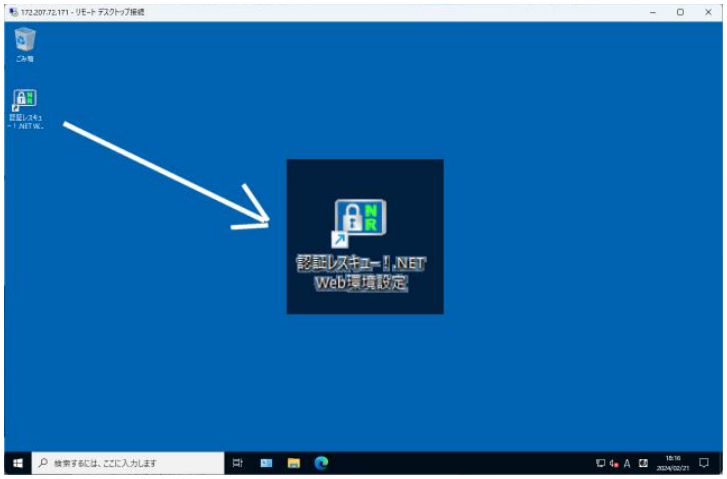
これで、仮想マシンの Windows Server 2022 の言語、地域、日付、および時間が日本や日本語に設定されます。

8.仮想マシンへの認証レスキュー！の Web サーバー用 PC のインストール

入手した認証レスキュー！がパッケージの場合は、ディスク内の全フォルダの全ファイル、ダウンロードなどの場合は解凍したフォルダにある全フォルダの全ファイルを、仮想マシンの任意のフォルダにコピーしてください。全体で 1.3GB ほどありますので通信環境によっては時間がかかる場合があります。

次に、コピーしたルートフォルダにある「NRInstallMenu.exe」を実行してください
通常通り、認証レスキュー！のインストーラで「Web サーバー用 PC」へのインストールを行います。
以降の手順は、本操作ガイドの「インストール」の「Web サーバー用 PC へのインストール」と同様ですので、リンク先をご覧ください。

「Web サーバー用 PC へのインストール」が完了するとデスクトップに「認証レスキュー！ .NET Web 環境設定」のショートカットが表示されます。



これで、Microsoft Azure の仮想マシンへのインストールは終了です。

9.仮想マシンの Web サーバー用 PC の「環境設定」処理

あとは、仮想マシン上の Web サーバー用 PC の「環境設定」を設定します。
詳しくは前述の Web サーバー用 PC の「環境設定」処理を参照してください。

●Amazon AWS で認証レスキュー！を利用する方法

ここでは、「認証レスキュー！」をアマゾン社のクラウドサービス Amazon AWS で利用する方法について説明します。

「認証レスキュー！」を Amazon AWS で利用するには、AWS の EC2（仮想サーバー）を利用します。

この仮想サーバーを利用すると、オンプレミスの自社サーバーへのインストールとほぼ同様の手順で「認証レスキュー！」の利用開始ができます。

《参考》AWS の EC2（仮想サーバー）のコストの例 （2024/09/06 時点）

・構成例

リージョン（場所）：アジアパシフィック（東京）
テナンシー （共有インスタンス）インスタンス数：1
オペレーティングシステム：Windows Server
EC2 インスタンス：t3a.medium、Family: t3a、2vCPU、4 GiB メモリ
EBS ストレージ量：（128 GB、汎用 SSD）

・1 ケ月あたりのコス合計： 48.21 USD

仮想マシン：	35.92 USD	
マネージドディスク：	12.29 USD	
合計：	48.21 USD	（約 6,990 円）※145 円/USD

アマゾン社の AWS を使用するための月単位のコストを計算する「AWS 料金見積りツール」の URL は次の通りです。
https://calculator.aws/#/addService?nc2=pr

■Amazon AWS の仮想サーバーを利用する

まず、貴社がアマゾン社に申込み、Amazon AWS の EC2（仮想マシン）が使用できる状態が必要です。Amazon AWS の申込みなどにつきましては、アマゾン社の情報をご覧ください。
ここでは、貴社がすでに Amazon AWS の EC2（仮想マシン）を作成できる状態を前提としています。

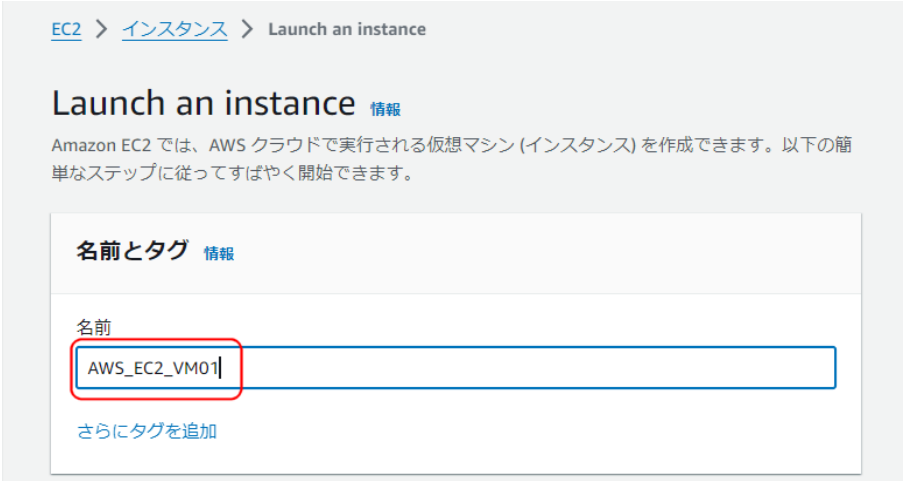
1.仮想サーバーの作成

次のアマゾン社のサイト「Amazon EC2 の使用を開始する」も参考にされてください。
https://docs.aws.amazon.com/ja_jp/AWSEC2/latest/UserGuide/EC2_GetStarted.html

以降は、Amazon EC2 コンソールでの操作の例です。

「EC2 ダッシュボード」で「インスタンスを起動」をクリックします。
その際に、特段の理由がなければ、「リージョン」（仮想サーバーの場所）項目は、アジアパシフィック(東京または大阪)を選択することをお勧めします。仮想サーバーを海外で指定すると、日本からのアクセスだとやはり時間がかかります。右上のリストでリージョンを指定できます。

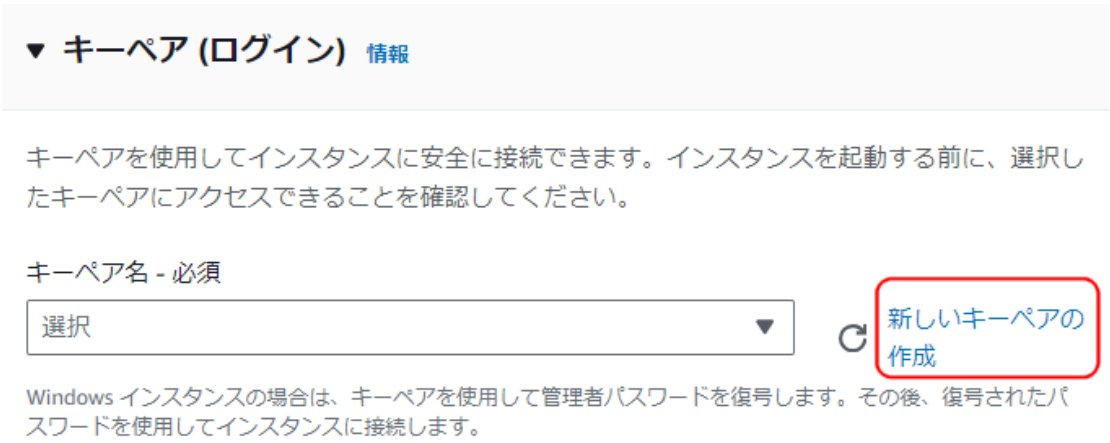
「Launch an instance」の「名前とタグ」で、任意の名前を入力します。



次に、「アプリケーションおよび OS イメージ（Amazon マシンイメージ）」で「Windows」を選択します。

次に、「インスタンスタイプ」を選択します。
この例では、2vCPU、4GB メモリの「t2.medium」を指定しています。認証レスキュー！での使用では少なくともこのレベルをお勧めします。

次に、キーペア（ログイン)を作成します。「新しいキーペアの作成」をクリックします。



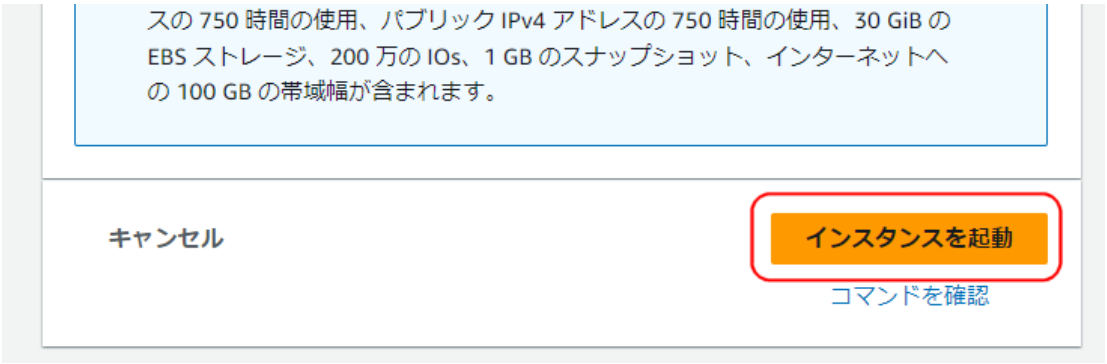
次に、「キーペアを作成」画面が表示されますので、任意のキーペア名を入力して、「キーペアを作成」をクリックします。

そうすると、ブラウザの「ダウンロード」に次のように履歴が表示されます。
このプライベートキーファイル（.pem）は後でインスタンスの接続時に使用しますので保存してください。

次に、ネットワーク設定を行います。
「RDP トラフィックを許可」は、貴社の固定グローバル IP アドレスがあれば、自動的に表示されますので「自分の IP」に設定すると他の IP アドレスでは RDP（リモートデスクトップ）接続は不可となり、安全です。
さらに、「インターネットからの HTTPS トラフィックを許可」と「インターネットからの HTTP トラフィックを許可」にチェックを入れます。

次に、ストレージを設定します。
例では、容量は 128GB を設定しています。

最後に、一番下までスクロールして、「インスタンスを起動」をクリックします。



Amazon EC2 コンソールの「インスタンス」の「インスタンス」を選択すると作成した仮想サーバーのインスタンスが表示されます。
そのインスタンス ID をクリックします。

作成した仮想サーバーのインスタンスの概要を確認できます。

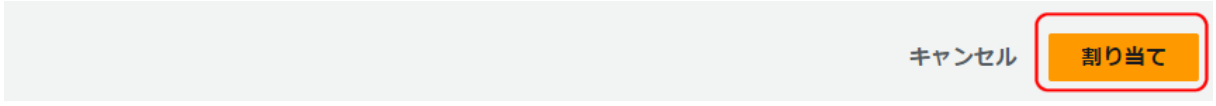
2.固定グローバル IP アドレスと DNS 名の設定と割り当て

ここでは、認証レスキュー！の認証管理システムの「環境設定」や DLL のプロパティに設定して利用することになる、仮想サーバーの固定グローバル IP アドレスや DNS の設定をします。

Amazon EC2 コンソールの「Elastic IP」を選択して、「Elastic IP アドレスを割り当てる」をクリックします。

Elastic IP アドレスを割り当てる画面の「Elastic IP アドレスの設定」の「パブリック IPv4 アドレスプール」が「Amazon の IPv4 アドレスプール」が選択されているのを確認します。

下までスクロールして、「割り当て」をクリックします。



すると、次の画面が表示されますので引き続き、右上の「この Elastic IP アドレスを関連付ける」をクリックします。

「Elastic IP アドレスの関連付け」の画面が表示されますので、「リソースタイプ」が「インスタンス」に選択されているのを確認します。さらに「インスタンス」が初期状態は空白になりますので、テキストボックスをクリックして表示されるリストから、先に作成したインスタンスを選択します。
最後に、下の「関連付ける」をクリックします。

そうすると、Elastic IP アドレスの関連付けが終了して次のように表示されます。

この場合は、仮想サーバーの固定グローバル IP アドレスは、「54.150.131.249」で、DNS は「ec2-54-150-131-249.ap-northeast-1.compute.amazonaws.com」ということになります。
つまりこれが、認証レスキュー！の認証管理システムの「環境設定」や DLL のプロパティに設定して利用する仮想サーバーの固定グローバル IP アドレスや DNS となります。

具体的には、この場合の認証レスキュー！の Web サービスの URL は、
http:// 54.150.131.249/NRDWebService/Service.asmx
または、
http:// ec2-54-150-131-249.ap-northeast-1.compute.amazonaws.com/NRDWebService/Service.asmx
と設定することになります。

次が認証レスキュー！の認証管理システムの環境設定画面での例です。

また、この場合の認証レスキュー！の DLL の Web サービスの URL（ WebServiceURL プロパティ）への設定は、次の通りとなります。（VB.NET）

Class1.myActivate.WebServiceURL =
"http://54.150.131.249/NRDWebService/Service.asmx"

または

Class1.myActivate.WebServiceURL =
"http://ec2-54-150-131-249.ap-northeast-1.compute.amazonaws.com/NRDWebService/Service.asmx"

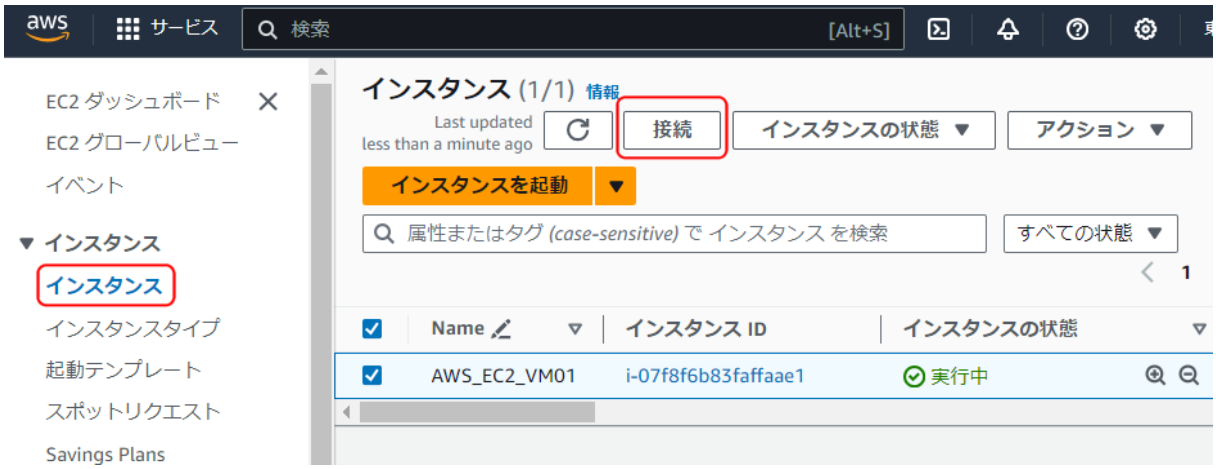
なお、貴社で独自のドメインを割り当てることもできます。その場合は、貴社のドメインの DNS サーバーにこの仮想マシンのパブリック IP アドレスを設定する必要があります。

また、http ではなく https の SSL サイトとしてアクセスさせる場合は、仮想サーバーのIISに SSL サーバー証明書をインストールする必要があります。

3.仮想サーバーの接続

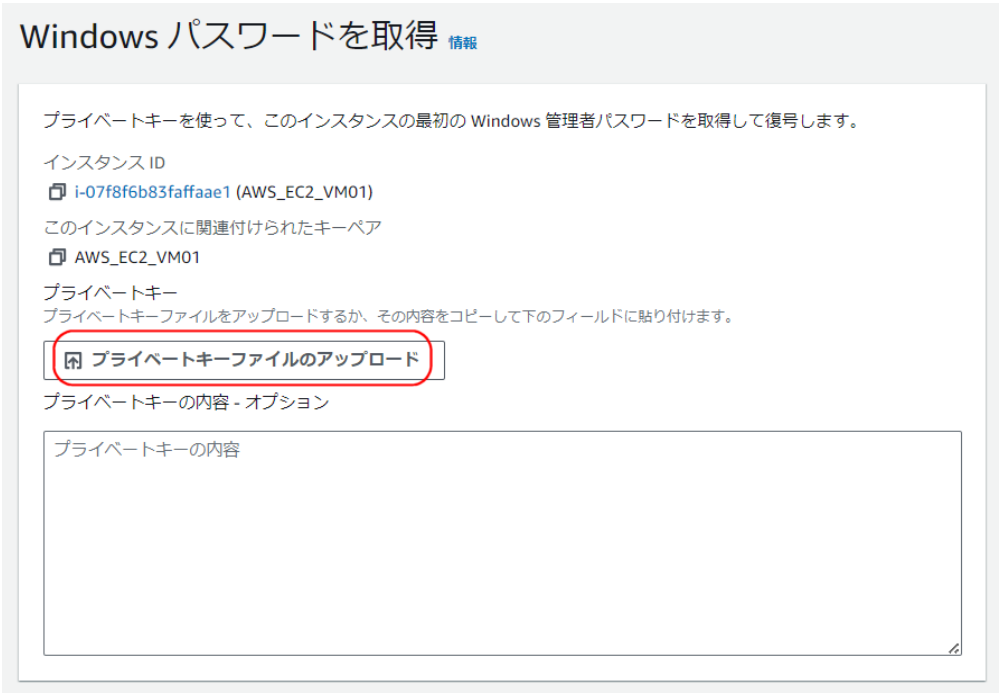
仮想サーバーに接続します。

Amazon EC2 コンソールの「インスタンス」の「インスタンス」で、該当するインスタンスチェックが入っている状態で、右上中央の「接続」をクリックします。

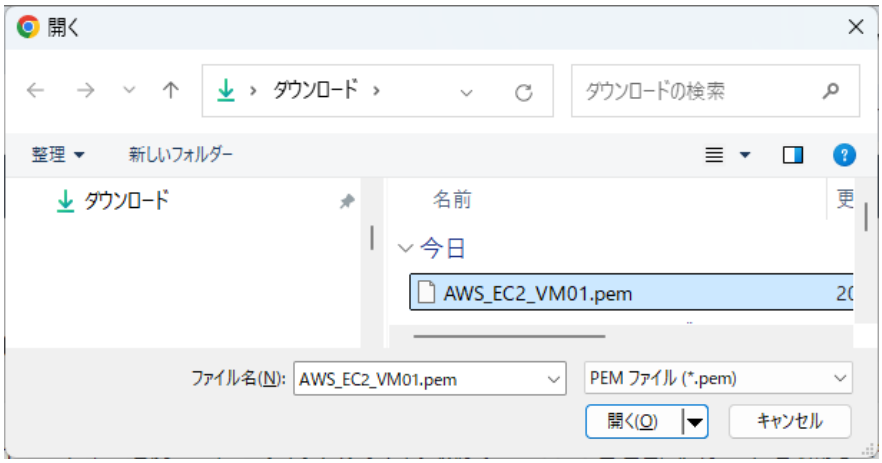


「インスタンスに接続」画面の、「RDP クライアント」タブを選択して、「パスワードを取得」をクリックします。

Windows パスワードを取得画面で、「プライベートキーファイルのアップロード」をクリックします。



ファイルダイアログが開くので、先の「キーペアを作成」時に保存したプライベートキーファイル（.pem）を選択します。

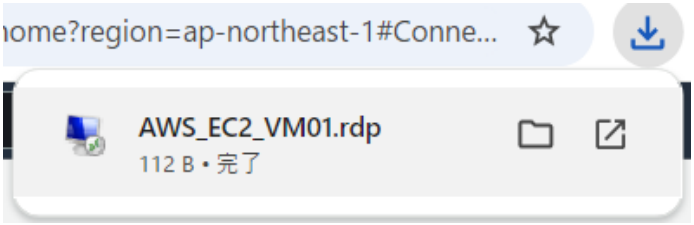


そうすると、Windows パスワードを取得画面の「プライベートキーの内容」のテキストボックスにプライベートキーの内容が表示されます。
そこで、「パスワードを復号化」をクリックします。

インスタンスに接続画面にパスワードが表示されます。
Public DNS、パスワード、ユーザ名はそれぞれコピーボタンを使って、適切なファイルに保存しておいてください。RDP（リモートデスクトップ）のログイン時に使用します。

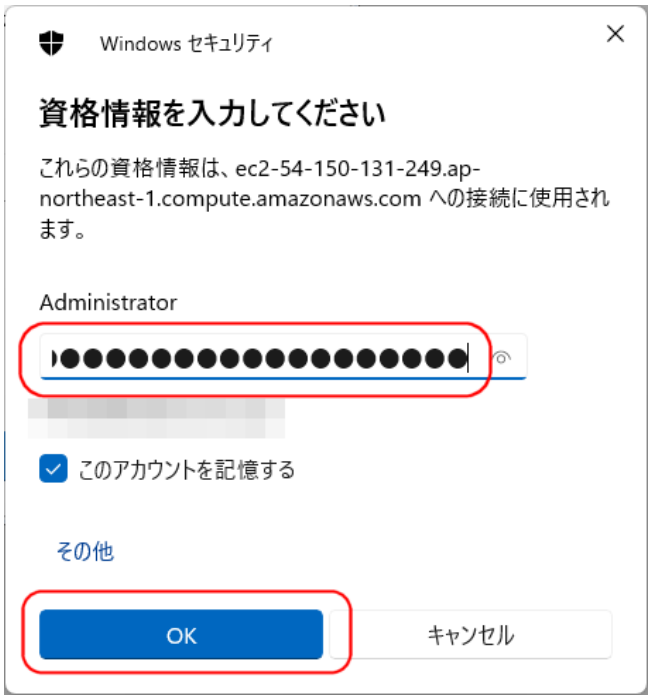
「リモートデスクトップファイルのダウンロード」をクリックします。

ダウンロードしたファイル（.rdp）を実行します。

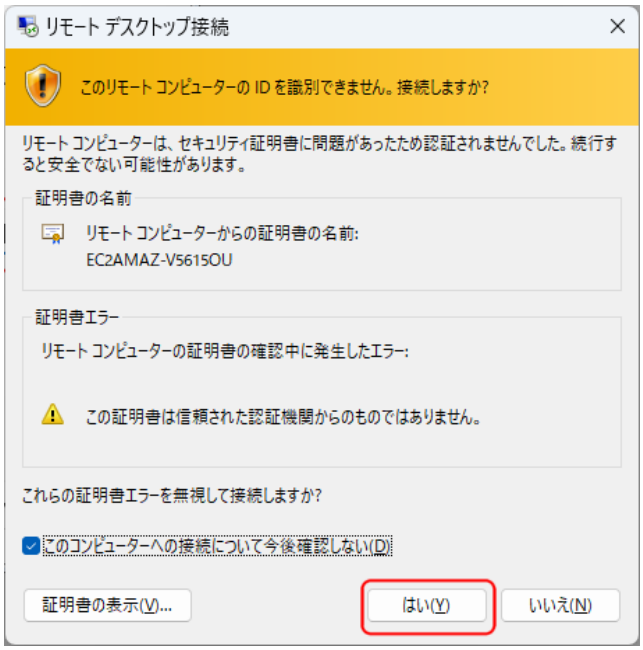


リモートデスクトップ接続のダイアログで「接続」をクリックします。

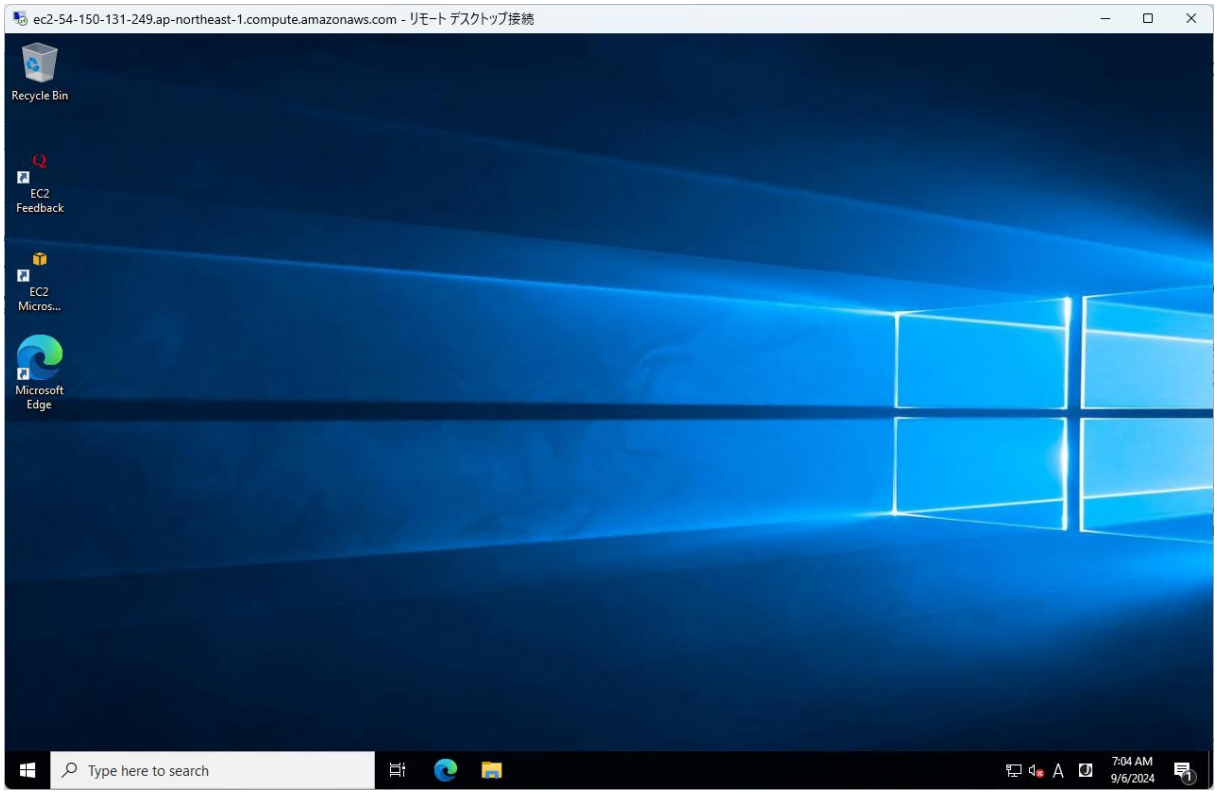
資格情報入力で、パスワードを入力して「OK」を押します。パスワードは、先のインスタンス接続画面で表示されたものです。



リモートデスクトップ接続画面で、「はい」をクリックします。



仮想サーバー（この場合、Windows Server 2022）のデスクトップが表示されます。



AWS EC2 仮想サーバーはデフォルトで英語（米国）ですが、次の手順を参考にして、それらを日本語化にしてから、後述の「仮想サーバーへの認証レスキュー！の Web サーバー用 PC のインストール」を行うことをお勧めします。

4.仮想サーバーの日本語化の手順

AWS EC2 仮想サーバーの日本語化などの一般的な手順の例（Windows Server 2022）は、次の通りです。

(1) RDP（リモートデスクトップ）接続: 仮想サーバーに RDP 経由で接続します。

(2) 言語と地域の設定:
- スタートメニューを開き、「Settings（設定）」を選択します。
- 「Time & Language（日時と言語）」をクリックします。
- 「Region」タブを選択し、表示されるリストから「Japan」を選択します。

(3) 日付と時刻の設定:
- 「Date & Time（日付と時刻）」タブを選択します。
- 「Time zone（タイムゾーン）」で「(UTC+09:00) Osaka, Sapporo, Tokyo」を選択します。

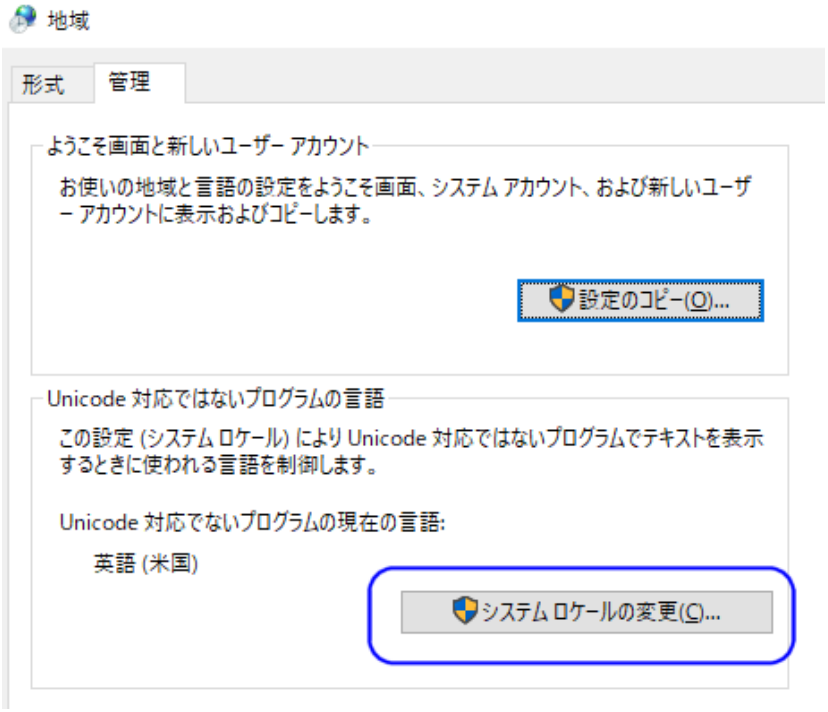
(4) 言語の設定:
- 「Language（言語）」タブを選択します。
- 「Preferred languages（優先言語）」をクリックし、「Add a language（言語を追加）」を選択します。
- 表示されるリストから「Japanese（日本語）」を選択し、「Next（次へ）」をクリックします。
- 「日本語 （日本） - [ja-JP]」を選択し、「Next（次へ）」をクリックします。
- オプションでデフォルトの表示言語を変更するかどうかを選択し、「Install（インストール）」をクリックします。

- その後、再起動が必要な場合は再起動します。

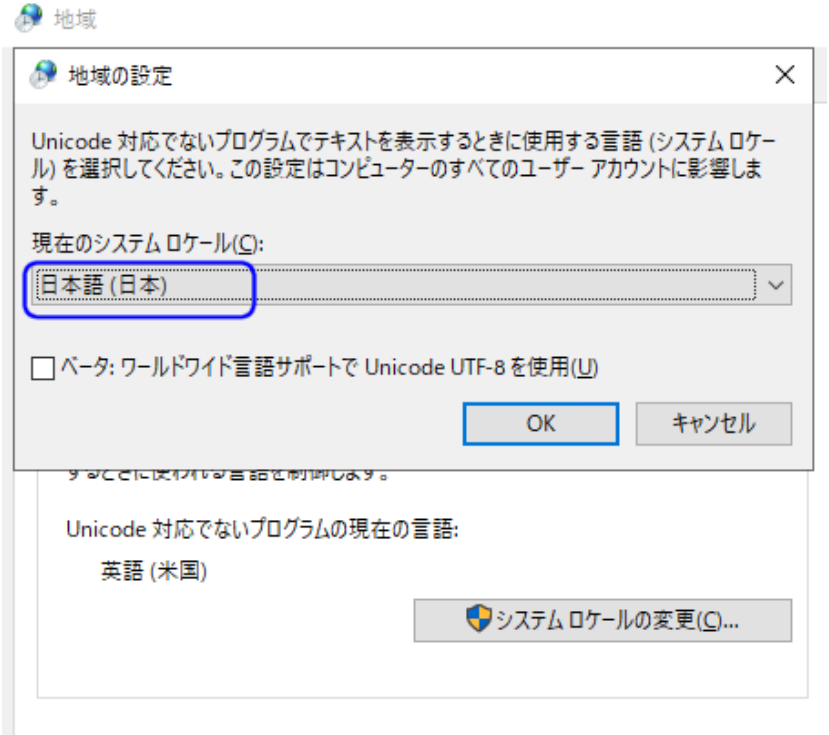
(5) 管理用言語の設定:

- 「言語」タブで「管理用の言語の設定」を選択します。

- 「管理」タブで「システムロケールの変更」を選択します。



- 「地域の設定」で「現在のシステムロケール」を「日本語（日本）」に設定して「OK」をクリックします。



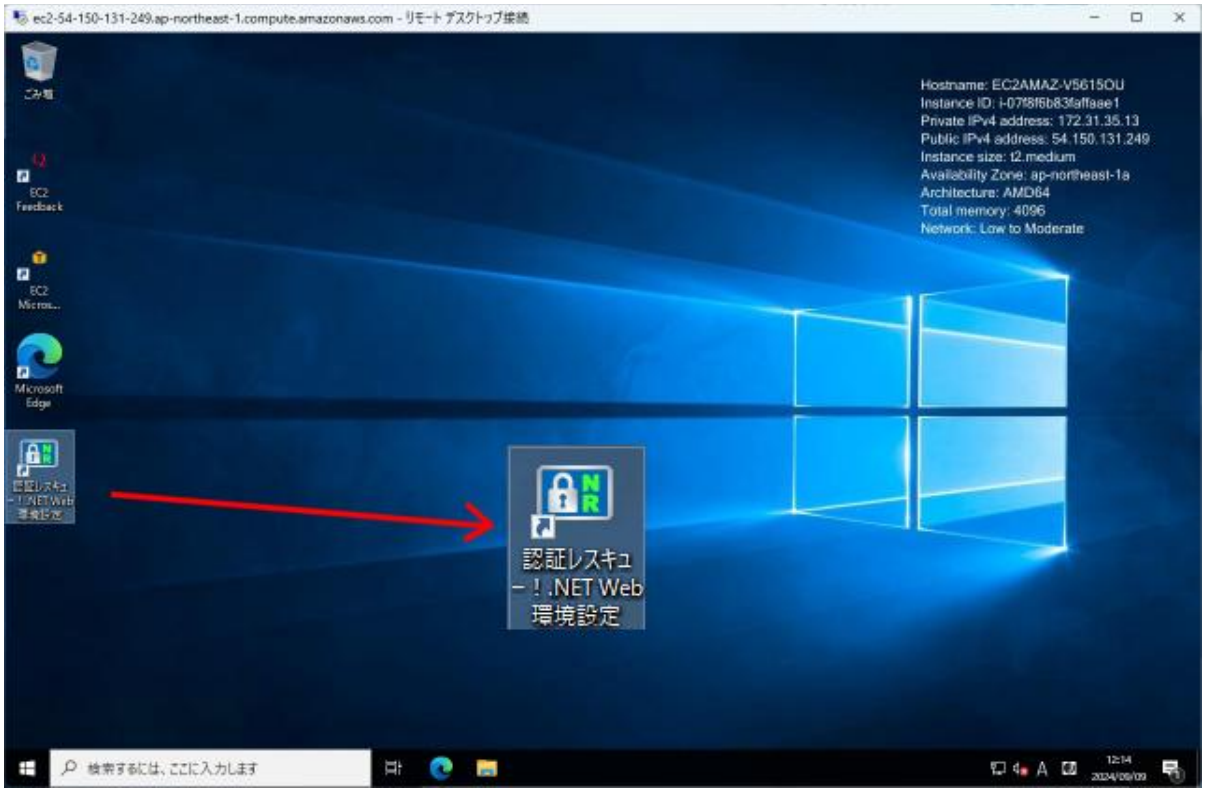
これで、仮想サーバーの Windows Server 2022 の言語、地域、日付、および時間が日本や日本語に設定されます。

5.仮想サーバーへの認証レスキュー！の Web サーバー用 PC のインストール

入手した認証レスキュー！がパッケージの場合は、ディスク内の全フォルダの全ファイル、ダウンロードなどの場合は解凍したフォルダにある全フォルダの全ファイルを、仮想マシンの任意のフォルダにコピーしてください。全体で 1.3GB ほどありますので通信環境によっては時間がかかる場合があります。

次に、コピーしたルートフォルダにある「NRInstallMenu.exe」を実行してください
通常通り、認証レスキュー！のインストーラで「Web サーバー用 PC」へのインストールを行います。
以降の手順は、本操作ガイドの「インストール」の「Web サーバー用 PC へのインストール」と同様ですので、リンク先をご覧ください。

「Web サーバー用 PC へのインストール」が完了するとデスクトップに「認証レスキュー！ .NET Web 環境設定」のショートカットが表示されます。



これで、AWS　EC2 の仮想サーバーへのインストールは終了です。

6.仮想サーバーの Web サーバー用 PC の「環境設定」処理

あとは、仮想サーバー上の Web サーバー用 PC の「環境設定」を設定します。
詳しくは前述の Web サーバー用 PC の「環境設定」処理を参照してください。

●アプリケーション開発用 PC での「認証 UI ライブラリ」の利用

■認証 UI ライブラリ関連ファイル

デスクトップ上の「認証レスキュー！ 認証 UI ライブラリ」へのショートカットを実行すると認証 UI ライブラリが格納されている（インストール先がデフォルトの場合）次のフォルダが開きます。

〈32bitOS の場合〉 C:¥Program Files¥Newtone¥NRD¥NRDDLL¥NRDLL
〈64bitOS の場合〉 C:¥Program Files (x86)¥Newtone¥NRD¥NRDDLL¥NRDLL

このフォルダに格納されているファイルは次の通りです。

フォルダ名	ファイル名/フォルダ名	内容
NRDLL	Newtone.NR.ASPNET.dll	ASP.NET 系認証 UI ライブラリ（DLL）
	Newtone.NR.FW45.dll	Framework4.5 用認証 UI ライブラリ（DLL）
	Newtone.NR.NET6.dll	.NET6 用認証 UI ライブラリ（DLL）
	NewtoneNRDvcpp.dll	VC++用ラッパーDLL（VC++使用時必要
	NewtoneNRDvcpp.tlb	タイプライブラリ（VC++使用時必要）
SampleProject（既定 UI 系サンプルプロジェクト）または SampleProject_API（カスタム UI 系サンプルプロジェクト）	VisualBasic フォルダ	Visual Basic 用のサンプルプロジェクト
	CSharp フォルダ	C# 用のサンプルプロジェクト
	VC++フォルダ	VC++ 用のサンプルプロジェクト
SampleProject_Web（ASP.NET 系サンプルプロジェクト）	VisualBasic フォルダ	Visual Basic 用のサンプルプロジェクト
	CSharp フォルダ	C# 用のサンプルプロジェクト

このライブラリを利用して、お客様（エンドユーザ）へ配布する貴社のアプリケーションにアクティベーション機能を組み込みます。
まずは、サンプルプロジェクトをお試しになり、ソースファイルをご確認ください。
なお、既定 UI 系機能とカスタム UI 系機能の利用形態の選択については次ページをご覧ください。

■認証 UI ライブラリの利用形態による選択

既定 UI 系機能とカスタム UI 系機能の大きな違いは、既定 UI 系が認証レスキュー！の既定の UI(ユーザインターフェイス)を利用する前提であることに対し、カスタム UI 系は貴社オリジナルの UI を作成し利用できる点です。

たとえば、「認証登録/インターネット」処理を考えた場合、既定 UI 系では ActivateRegisterInternet メソッドを呼び出すだけで、「認証登録/インターネット」処理の UI が表示され UI 上の項目の細かいプログラム的な制御などは全く必要ありません。
それに対し、カスタム UI 系では最終的には APIActivateRegisterInternet メソッドを呼び出しますが、UI にあたる Form のデザインや Form 上の項目に関する細かい制御や APIActivateRegisterInternet メソッドを呼び出すための必須プロパティの設定などのコードを貴社で作成する必要があります。カスタム UI 系では各メソッド実行結果の細かいステータスは返しますが、メッセージの表示を含め一切の画面表示は行いません。

認証 UI ライブラリ(DLL)の利用形態によるメリットとデメリットは次表の通りです。

利用形態	メリット	デメリット
既定 UI 系機能	●認証レスキュー！既定の UI を利用するため、貴社アプリケーションにアクティベーション機能を短期間で実装することが可能	●貴社オリジナルの UI を構築できないため、貴社アプリケーションとのシームレスな UI デザインや UI 上の(認証レスキュー！既定 UI には用意されていない)新規項目などは利用できない ●認証レスキュー！既定の UI を利用するため日本語のみの UI となる
カスタム UI 系機能	●貴社オリジナルの UI を構築できるため、貴社アプリケーションとのシームレスな UI デザインや UI 上の(認証レスキュー！既定 UI には用意されていない)新規項目などを作成できる ・貴社オリジナルの UI を構築できるため日本語以外の多言語の UI が作成可能 ●プロダクト ID やシリアル No.を(レジストリより)取得できるメソッドがあるため、その文字列を定義することで、異なる製品や同一製品の追加オプションのライセンスを同一 PC 内で識別できる。 ※次ページの(2)に例を掲載、既定 UI 系にはそれらを取得する機能はない	●貴社オリジナルの UI を構築するため既定 UI 系に比べ、貴社アプリケーションのアクティベーション機能実装に時間がかかる

■同一 PC 内で異なるアプリケーションやオプションのライセンスを識別する方法

2 つの方法があります。

(1) 同一 PC 内で貴社の異なるアプリケーションのライセンスを識別する場合

この方法は、同一 PC 内の異なるアプリケーションのライセンスの識別ができますが、同一アプリケーション内の異なるオプションなどについてはライセンスの識別はできません。

ベンダアプリケーション開始レジストリキーパス設定用のプロパティにアプリケーションごとに異なるレジストリパスを設定します。

既定 UI 系:
VendorsProductStartRegistryKeyPath プロパティ

アプリケーションA内での設定コード例:
VendorsProductStartRegistryKeyPath = "Software¥Company¥A"
アプリケーションB内での設定コード例:
VendorsProductStartRegistryKeyPath = "Software¥Company¥B"

カスタム UI 系:
APIVendorsProductStartRegistryKeyPath プロパティ

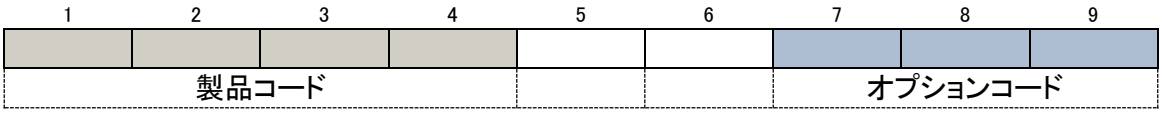
アプリケーションA内での設定コード例:
APIVendorsProductStartRegistryKeyPath = "Software¥Company¥A"
アプリケーションB内での設定コード例:
APIVendorsProductStartRegistryKeyPath = "Software¥Company¥B"

(2) 同一 PC 内で貴社の異なるアプリケーションやオプションのライセンスを識別する場合

この方法は、同一 PC 内の同一アプリケーション内の異なるオプションについてもライセンスの識別ができます。この方法は、カスタム UI 系限定の機能を使用します。既定 UI 系では利用できません。

例えば、「プロダクト ID」9 桁とし先頭 4 桁をアプリケーションの製品コード、最後の 3 桁をオプションコードとします。

[プロダクト ID の桁の定義]



[製品コードの定義]
0001:アプリケーション A
0002:アプリケーション B
0003:アプリケーション C
・・・・

[オプションコードの定義]
オプションコード1桁目(プロダクト ID 7 桁目):追加機能 1 がある場合 1、ない場合 0

オプションコード 2 桁目（プロダクト ID　8 桁目）：追加機能 2 がある場合 1、ない場合 0
オプションコード 3 桁目（プロダクト ID　9 桁目）：追加機能 3 がある場合 1、ない場合 0

[貴社でのプログラム]
ライセンスを確認したいタイミングで APIGetRegisteredInfoFromRegistry メソッドを実行して、認証後のレジストリより「プロダクト ID」を取得します。その値の先頭の 4 桁の製品コードでアプリケーションを識別し、最後の 3 桁でオプションを識別できます。

例えば、取得したプロダクト ID が次の通りであれば

1	2	3	4	5	6	7	8	9
0	0	0	2	1	2	1	0	1

エンドユーザは、アプリケーション B の追加機能 1 と追加機能 3 のライセンスを保持していることになります。

■既定 UI 系サンプルプロジェクトとカスタム UI 系サンプルプロジェクトについて

認証 UI ライブラリ（DLL）用のサンプルプロジェクトは次の 2 種類です。

・SampleProject フォルダ（既定 UI 系サンプルプロジェクト）
・SampleProject_API フォルダ（カスタム UI 系サンプルプロジェクト）

これらのサンプルプロジェクトは、実際に実行してみると一見同じように見えますが、ソースコードを確認するとその相違点がわかります。

既定 UI 系サンプルプロジェクトが認証レスキュー！既定の UI を呼び出すサンプルプロジェクトになっているのに対し、カスタム UI 系サンプルプロジェクトでは UI 用の Form は全て貴社でカスタマイズ可能です。

■ASP.NET 系サンプルプロジェクトについて

利用できる ASP.NET（Web アプリケーション）用のサンプルプロジェクトは次の通りです。

・SampleProject_Web フォルダ（ASP.NET 系サンプルプロジェクト）

このサンプルプロジェクトは、ASP.NET で作成された貴社の Web ページから利用することができる、ASP.NET 系の DLL の利用方法が含まれています。

■DLL 及びサンプルフォルダのツリー図

フォルダ名の凡例は次の通りです。

分類	凡例	説明
開発言語	CSharp	C＃用
	VisualBasic	VB.NET 用
	VC++	VC++用
サンプルプロジェクトの種類	PackageApp	下記「ProxyApp」を除いた認証機能のサンプルプロジェクト
	ProxyApp	代理認証のオフライン PC 側での操作のサンプルプロジェクト
フレームワーク	*.FW45	Framewaork 4.5 用 （Vusual Studio 2015 用）
	*.NET6	.NET6 用 （Visual Studio 2022 用）

```
├──NRDLL                → 認証 UI ライブラリ（DLL）の格納フォルダ
├──SampleProject        → 既定 UI 系のサンプルフォルダ
│    ├──CSharp
│    │    ├──PackageApp
│    │    │    ├──SamplePackageApp.FW45
│    │    │    └──SamplePackageApp.NET6
│    │    └──ProxyApp
│    │         ├──SampleProxyApp.FW45
│    │         └──SampleProxyApp.NET6
│    ├──VisualBasic          ****VB.NET 用
│    │    ├──PackageApp
│    │    │    ├──SamplePackageApp.FW45
│    │    │    └──SamplePackageAPP.NET6
│    │    └──ProxyApp
│    │         ├──SampleProxyApp.FW45
│    │         └──SampleProxyApp.NET6
│    └──ViC++
│         ├──PackageApp
│         │    ├──SamplePackageApp.FW45
│         └──ProxyApp
│              └──SampleProxyApp.FW45
├──SampleProject_API       → カスタム UI 系のサンプルフォルダ
│    ├──CSharp
│    │    ├──PackageApp
│    │    │    ├──SamplePackageAppAPI.FW45
│    │    │    └──SamplePackageAppAPI.NET6
│    │    └──ProxyApp
│    │         ├──SampleProxyAppAPI.FW45
│    │         └──SampleProxyAppAPI.NET6
│    ├──VisualBasic
│    │    ├──PackageApp
│    │    │    ├──SamplePackageAppAPI.FW45
│    │    │    └──SamplePackageAppAPI.NET6
│    │    └──ProxyApp
│    │         ├──SampleProxyAppAPI.FW45
│    │         └──SampleProxyAppAPI.NET6
│    └──VC++
```

```
│           ├──PackageApp
│           │      └──SamplePackageAppAPI.FW45
│           └──ProxyApp
│                  └──SampleProxyAppAPI.FW45
└──SampleProject_Web      →  ASP.NET 系サンプルフォルダ （Vusual Studio 2015 用）
    ├──CSharp
    │      └──WebApplication2
    └──VisualBasic
           └──WebApplication1
```

■認証 UI ライブラリ機能一覧

【既定 UI 系、カスタム UI 系 DLL】
●.NET Freamework4.5 用 DLL
アセンブリ：Newtone.NR.FW45 （Newtone.NR.FW45.dll 内）
名前空間：Newtone.NR.FW45
●.NET6 用 DLL
アセンブリ：Newtone.NR.NET6 （Newtone.NR.NET6.dll 内）
名前空間：Newtone.NR.NET6

Visual C++での利用時は、次の VC++用ラッパーDLL を呼び出すようになります。
アセンブリ：NewtoneNRDvcpp （NewtoneNRDvcpp.dll 内）
名前空間：NewtoneNRDvcpp

【ASP.NET 系 DLL】
アセンブリ：Newtone.NR.ASPNET （Newtone.NR.ASPNET.dll 内）
名前空間：Newtone.NR.ASPNET

<クラス>

内容	クラス名
認証に関する各種機能の提供（既定 UI 系）	NRD_Activation
認証に関する各種機能の提供（カスタム UI 系）	NRD_APIActivation
認証に関する各種機能の提供（ASP.NET 系）	NRD_APIActivation2

<既定 UI 系プロパティ>

プロパティー覧

プロパティ	機能
DisabledNICIgnore	処理を高速化するため、無効になっている NIC (Network Interface Card)を無視します。
EncryptionPassword	暗号化時のパスワードを設定
EncryptionSaltString	暗号化時の Salt 文字列を設定
ProductIdNumberOfDigits	プロダクト ID の桁数を設定
SerialNoNumberOfDigits	シリアル No.の桁数を設定
SetProductID	認証登録時の設定プロダクト ID を設定
SetSerialNo	認証登録時の設定シリアル No.を設定
TelephoneNumber	電話で認証時の電話番号を設定
TrialPeriod	猶予(試用)日数を設定
TrialPeriodName	猶予(試用)期間の名称を設定
UseCpuInfo	CPU 情報の使用を設定
UseMacAddress	MAC アドレスの使用を設定
VendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパスを設定
WebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワードを設定
WebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名を設定
WebServiceCheckPassword	Web サービス確認パスワードを設定
WebServiceTimeout	Web サービスのタイムアウトを設定
WebServiceURL	Web サービスの URL を設定
WebServiceUseBasicAuthentication	Web サービス時の基本認証の使用を設定

DisabledNICIgnore プロパティ

【機能】
処理を高速化するため、無効になっている NIC（Network Interface Card）を無視します。
デフォルト：True。

【構文】
<VB.NET>
Public Property **DisabledNICIgnore** As **Boolean**

<C#>
public **bool** **DisabledNICIgnore** { set; get; }

<VC++>
public : **VARIANT_BOOL NewtoneNRDvcpp::INRD_Activation::DisabledNICIgnore**

【解説】
認証レスキュー！の認証 UI ライブラリでは、認証登録など各処理時にエンドユーザ PC の MAC アドレスを最大で 5 個記録します。MAC アドレスは LAN ポートなどのネットワーク機器に固有な物理アドレスです。
このプロパティが False の場合、無効になっている NIC を一度有効にし、情報を取得後に再度 NIC を無効にします。この際に少し時間が掛かります。
このプロパティを True に設定すると、この無効の NIC は無視し、有効の NIC のみ情報を取得して記録します。これにより、情報の取得が高速化します。
認証レスキュー！の各処理では、最初に認証状況を確認しており、認証済みの場合、記録されている情報と、使用されている PC 情報が合致しているか確認しています。
このプロパティを True に設定することで、各処理が高速化します。

＜ご注意＞
たとえば、エンドユーザ PC の NIC が 2 つ有効で 3 つが無効の状態とします。

MACアドレスの例：
①A11111111111 → NIC が有効
②B22222222222 → NIC が有効
③C33333333333 → NIC が無効
④D44444444444 → NIC が無効
⑤E55555555555 → NIC が無効

このプロパティが False の場合、5 個の MAC アドレスを取得するために、2 つ有効（①②）になっている MAC アドレスと、3 つの無効（③④⑤）の NIC を一度有効にし、MAC アドレスを取得後に再度無効に設定します。
この無効の NIC を有効にし、再度無効にすることで、少々時間が掛かります。

このプロパティが True の場合、この 3 つの無効の NIC は無視し、2 つの有効な MAC アドレスのみ取得します。これにより、MACアドレスの取得時間が短縮されます。
しかし、次の点にご注意ください。
たとえば、このプロパティが True で、2 つの有効な（①②）になっている MAC アドレスで認証登録をします。
次に、2 つの有効な（①②）NIC を無効にし、3 つの無効な（③④⑤）を有効にして認証解除は、MAC

アドレスが一致しないため「-999：認証済みハードウエア情報不一致」となり実行できません。

EncryptionPassword プロパティ

【機能】
暗号化時のパスワードを設定します。

【構文】
<VB.NET>
Public Property **EncryptionPassword** As **String**

<C#>
public **string** **EncryptionPassword** { set; get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_Activation::EncryptionPassword**

【解説】
認証 UI ライブラリ(DLL)はエンドユーザ PC のレジストリやファイルにデータを出力する際に必要に応じてデータを暗号化しています。その際の暗号化の方法は貴社では指定できませんが、暗号化する時の 2 つのパラメータ、パスワードと Salt 文字列は貴社で指定できます。

EncryptionPassword プロパティには暗号化時のパスワードを指定します。全角でも指定できます。
(例)"認証レスキュー！"
この EncryptionPassword プロパティの文字数は、空文字列は不可で 1～65535 文字ですが、8 文字から 15 文字程度が妥当と思われます。

EncryptionSaltString プロパティ

【機能】
暗号化時の Salt 文字列(8 文字以上)を設定します。

【構文】
<VB.NET>
Public Property **EncryptionSaltString** As **String**

<C#>
public **string** **EncryptionSaltString** { set; get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_Activation::EncryptionSaltString**

【解説】
認証 UI ライブラリ(DLL)はエンドユーザ PC のレジストリやファイルにデータを出力する際に必要に応じてデータを暗号化しています。その際の暗号化の方法は貴社では指定できませんが、暗号化する時の 2 つのパラメータ、パスワードと Salt 文字列は貴社で指定できます。

EncryptionSaltString プロパティには暗号化時の Salt 文字列を指定します。
必ず 8 文字以上で指定します。
(例)"12345678ABCDEFGH"

ProductIdNumberOfDigits プロパティ

【機能】
プロダクト ID の桁数(デフォルト:17)を設定します。

【構文】
<VB.NET>
Public Property **ProductIdNumberOfDigits** As **Integer**

<C#>
public **int** **ProductIdNumberOfDigits** { set; get; }

<VC++>
public : **long NewtoneNRDvcpp::INRD_Activation::ProductIdNumberOfDigits**

【解説】
認証業務用社内 PC の「認証管理システム」の「データテーブル新規作成」処理時に指定したプロダクト ID の桁数を設定します。

SerialNoNumberOfDigits プロパティ

【機能】
シリアル No.の桁数（デフォルト：8）を設定します。

【構文】
<VB.NET>
Public Property **SerialNoNumberOfDigits** As **Integer**

<C#>
public **int** **SerialNoNumberOfDigits** { set; get; }

<VC++>
public : **long NewtoneNRDvcpp::INRD_Activation::SerialNoNumberOfDigits**

【解説】
認証業務用社内 PC の「認証管理システム」の「データテーブル新規作成」処理時に指定したシリアル No.の桁数を設定します。

SetProductID プロパティ

【機能】
認証登録時の設定プロダクト ID（デフォルト：空文字列）を設定します。

【構文】
<VB.NET>
Public Property **SetProductID** As **String**

<C#>
public **string** **SetProductID** { set; get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_Activation::SetProductID**

【解説】
このプロパティが空文字列の場合は、認証登録系のメソッドで表示されるダイアログのプロダクトID用のテキストボックスは、エンドユーザによる入力が可能となります。

このプロパティが空文字列ではない場合は、認証登録系のメソッドで表示されるダイアログ中のプロダクトID用のテキストボックスにこのプロパティ値がグレーアウト表示され文字列コピーはできるが入力はできません。

このプロパティは認証登録系のメソッドに対し、プロダクト ID を渡すためだけに利用され、認証登録処理後に実際に登録されたプロダクト ID で自動的に書き換えられるものではありません。

対象となる認証登録系メソッドは、次の通りです。
「認証登録/インターネット」処理の呼び出し
（ActivateRegisterInternetメソッド）
「認証登録/電話」処理の呼び出し
（ActivateRegisterTelephoneメソッド）
「認証登録状態回復」処理の呼び出し
（RestoreRegisterStatusメソッド）
「代理認証登録/実行」処理の呼び出し
（ProxyActivateRegisterExecute メソッド）

SetProductID、SetSerialNoプロパティの目的
プロダクトIDやシリアルNo.に貴社が取り決めたある識別を設けた場合に、意図したように認証登録を行うため。

SetSerialNo プロパティ

【機能】
認証登録時の設定シリアル No.(デフォルト:空文字列)を設定します。

【構文】
<VB.NET>
Public Property **SetSerialNo** As **String**

<C#>
public **string** **SetSerialNo** { set; get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_Activation::SetSerialNo**

【解説】
このプロパティが空文字列の場合は、認証登録系のメソッドで表示されるダイアログのシリアルNo.用のテキストボックスは、エンドユーザによる入力が可能となります。

このプロパティが空文字列ではない場合は、認証登録系のメソッドで表示されるダイアログ中のシリアルNo.用のテキストボックスにこのプロパティ値がグレーアウト表示され文字列コピーはできるが入力はできません。

このプロパティは認証登録系のメソッドに対し、シリアルNo.を渡すためだけに利用され、認証登録処理後に実際に登録されたシリアルNo.で自動的に書き換えられるものではありません。

対象となる認証登録系メソッドは、次の通りです。
「認証登録/インターネット」処理の呼び出し
(ActivateRegisterInternetメソッド)
「認証登録/電話」処理の呼び出し
(ActivateRegisterTelephoneメソッド)
「認証登録状態回復」処理の呼び出し
(RestoreRegisterStatusメソッド)
「代理認証登録/実行」処理の呼び出し
(ProxyActivateRegisterExecute メソッド)

このプロパティの目的と利用例は、SetProductID プロパティの説明をご覧ください。

TelephoneNumber プロパティ

【機能】
電話で認証時の電話番号を設定します。

【構文】
<VB.NET>

```
Public Property TelephoneNumber As String
```

<C#>

```
public string TelephoneNumber { set; get; }
```

<VC++>

```
public : _bstr_t NewtoneNRDvcpp::INRD_Activation::TelephoneNumber
```

【解説】
エンドユーザがインターネットを使わずに電話での認証を行う「認証登録/電話」処理で、エンドユーザがかける電話の番号を指定します。

TrialPeriod プロパティ

【機能】
猶予(試用)日数(デフォルト:0 日、設定可能範囲:1～365)を設定します。

【構文】
<VB.NET>
Public Property **TrialPeriod** As **Integer**

<C#>
public **int** **TrialPeriod** { set; get; }

<VC++>
public : **long NewtoneNRDvcpp::INRD_Activation::TrialPeriod**

【解説】
エンドユーザがライセンス認証登録をしなくてもアプリケーションが動作する期間を指定します。
1(日)～365(日)の間で設定できます。
この TrialPeriod プロパティを 0 に設定すると、猶予期間は無いことになり、ライセンス認証登録をしない限りアプリケーション(またはアプリケーションの主機能など)が動作しないようにできます。

TrialPeriodName プロパティ

【機能】
猶予(試用)期間の名称(デフォルト:"猶予(試用)")を設定します。

【構文】
<VB.NET>
Public Property **TrialPeriodName** As **String**

<C#>
public **string** **TrialPeriodName** { set; get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_Activation::TrialPeriodName**

【解説】
たとえば、ActivateStatusDisp メソッドでは認証状態が表示されます。現在、猶予(試用)中ならば、「猶予(試用)期間残日数:6 日」といったように表示されます。その中の「猶予(試用)」という文字列を別の文字列で指定することができます。それがこの TrialPeriodName プロパティです。
(例)"体験版"

UseCpuInfo プロパティ

【機能】
CPU 情報の使用(デフォルト:True)を設定します。
次の 3 つのメソッドの処理において、認証情報とエンドユーザ PC 内の CPU 情報とを照合するかどうかを設定します。
・ActivateStatusCheck メソッド(エンドユーザ PC 内での認証状態の確認)
・ActivateStatusCheckOnline メソッド(オンラインで認証状態の確認)
・RestoreRegisterStatus メソッド(認証登録状態回復」処理の呼び出し)

【構文】
<VB.NET>
```
Public Property UseCpuInfo As Boolean
```

<C#>
```
public bool UseCpuInfo { set; get; }
```

<VC++>
```
public : VARIANT_BOOL NewtoneNRDvcpp::INRD_Activation::UseCpuInfo
```

【解説】
認証レスキュー！の認証 UI ライブラリでは、認証登録など各処理時にエンドユーザ PC の CPU 情報を記録します。等プロパティを True にするとエンドユーザ PC の CPU 情報を認証識別情報として利用します。これを利用することでエンドユーザ PC の不正利用時の認証識別情報の精度を上げます。
※RestoreRegisterStatus メソッド(認証登録状態回復」処理の呼び出し)を使用する際は、当プロパティを必ず True に設定してください。

UseMacAddress プロパティ

【機能】
MAC アドレスの使用（デフォルト：True）を設定します。
次の 3 つのメソッドの処理において、認証情報とエンドユーザ PC 内の MAC アドレスとを照合するかどうかを設定します。
・ActivateStatusCheck メソッド（エンドユーザ PC 内での認証状態の確認）
・ActivateStatusCheckOnline メソッド（オンラインで認証状態の確認）
・RestoreRegisterStatus メソッド（認証登録状態回復」処理の呼び出し）

【構文】
<VB.NET>
```
Public Property UseMacAddress As Boolean
```

<C#>
```
public bool UseMacAddress { set; get; }
```

<VC++>
```
public : VARIANT_BOOL NewtoneNRDvcpp::INRD_Activation::UseMacAddress
```

【解説】
認証レスキュー！の認証 UI ライブラリでは、認証登録など各処理時にエンドユーザ PC の MAC アドレスを最大で 5 個記録します。MAC アドレスは LAN ポートなどのネットワーク機器に固有な物理アドレスです。
当プロパティを True にするとエンドユーザ PC の MAC アドレスを認証識別情報として利用します。これを利用することでエンドユーザ PC の不正利用時の認証識別情報の精度を上げます。
※RestoreRegisterStatus メソッド（認証登録状態回復」処理の呼び出し）を使用する際は、当プロパティを必ず True に設定してください。

VendorsProductStartRegistryKeyPath プロパティ

【機能】
ベンダアプリケーション開始レジストリキーパスを設定します。

【構文】
<VB.NET>
Public Property **VendorsProductStartRegistryKeyPath** As **String**

<C#>
public **string** **VendorsProductStartRegistryKeyPath** { set; get; }

<VC++>
public : **_bstr_t**

NewtoneNRDvcpp::INRD_Activation::VendorsProductStartRegistryKeyPath

【解説】
エンドユーザに配布したアプリケーションが認証 UI ライブラリ（DLL）の機能を使う場合、その各種情報の記録先として使うエンドユーザ PC 上のレジストリの開始キーのパスを指定します。レジストリ内の「HKEY_LOCAL_MACHINE」以降を指定します。
（例）"Software¥Newtone¥NinshoRescue¥NRD¥SampleProject"

なお、物理的なレジストリの位置は動作する貴社のアプリケーションが32bitなのか64bitなのかと、動作するPCのOSが32bitなのか64bitなのかによって異なります。
たとえば、HKEY_LOCAL_MACHINE¥SOFTWARE¥ABCというレジストリパスは次のようになります。
32bitOS上で32bitアプリケーションが動作する場合、または64bitOS上で64bitアプリケーションが動作する場合は、
HKEY_LOCAL_MACHINE¥SOFTWARE¥ABC
そのままです。
しかし、64bitOS上で32bitアプリケーションが動作する場合は、
HKEY_LOCAL_MACHINE¥SOFTWARE**¥Wow6432Node**¥ABC
となります。

また、貴社の異なる複数のアプリケーションをエンドユーザの同一PC上で使用させるには、このVendorsProductStartRegistryKeyPathプロパティにアプリケーションごとのそれぞれ異なるレジストリパスを設定してください。たとえば、次のように設定します。
アプリケーションA内での設定コード例：
VendorsProductStartRegistryKeyPath = "Software¥Company¥A"
アプリケーションB内での設定コード例：
VendorsProductStartRegistryKeyPath = "Software¥Company¥B"

WebServiceBasicAuthenticationPassword プロパティ

【機能】
Web サービス時の基本認証パスワードを設定します。

【構文】
<VB.NET>
Public Property **WebServiceBasicAuthenticationPassword** As **String**

<C#>
public **string** **WebServiceBasicAuthenticationPassword** { set; get; }

<VC++>
public : **_bstr_t**

NewtoneNRDvcpp::INRD_Activation::WebServiceBasicAuthenticationPassword

【解説】
Web サーバーで基本認証を使用する場合に、そのパスワードを指定します。
基本認証に関しては、WebServiceUseBasicAuthentication プロパティを参照してください。

WebServiceBasicAuthenticationUserName プロパティ

【機能】
Web サービス時の基本認証ユーザ名を設定します。

【構文】
<VB.NET>
Public Property **WebServiceBasicAuthenticationUserName** As **String**

<C#>
public **string** **WebServiceBasicAuthenticationUserName** { set; get; }

<VC++>
public : **_bstr_t**

NewtoneNRDvcpp::INRD_Activation::WebServiceBasicAuthenticationUserName

【解説】
Web サーバーで基本認証を使用する場合に、そのユーザ名を指定します。
基本認証に関しては、WebServiceUseBasicAuthentication プロパティを参照してください。

WebServiceCheckPassword プロパティ

【機能】
Web サービス確認パスワード(8 文字以上)を設定します。

【構文】
<VB.NET>
Public Property **WebServiceCheckPassword** As **String**

<C#>
public **string** **WebServiceCheckPassword** { set; get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_Activation::WebServiceCheckPassword**

【解説】
Web サービスを利用する場合の確認用のパスワードを設定します。ここでは、Web サーバー側設定アプリケーション「認証 Web サービス」の環境設定処理の Web サービスの確認パスワードと同じものを設定します。
確認パスワードは必須項目です、省略はできません。8 文字以上で半角の次の文字が使用できます。
・大文字の英字（A～Z）
・小文字の英字（a～z）
・数字（0～9）
・記号（+-*/!#$%&()=¥@<>?）

WebServiceTimeout プロパティ

【機能】
Web サービスのタイムアウト(デフォルト:60 秒)を設定します。

【構文】
<VB.NET>

```
Public Property WebServiceTimeout As Integer
```

<C#>

```
public int WebServiceTimeout { set; get; }
```

<VC++>

```
public : long NewtoneNRDvcpp::INRD_Activation::WebServiceTimeout
```

【解説】
Web サービスに接続して応答を待つ最大時間を秒単位で指定します。初期値は 60 秒です。

WebServiceURL プロパティ

【機能】
Web サービスの URL を設定します。

【構文】
<VB.NET>
Public Property **WebServiceURL** As **String**

<C#>
public **string** **WebServiceURL** { set; get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_Activation::WebServiceURL**

【解説】
認証に関するシステムを Web サービスとして提供する Web サーバーの URL を指定します。
以下に例を示します。

自 PC のローカルホストの Web サーバー(IIS)にアクセスする例:
"http://localhost/NRDWebService/Service.asmx"
(注)この例は実際の運用ではありえません。貴社のアプリケーションで認証 UI ライブラリ(DLL)を使用した開発時に、テスト用に使用される URL です。

自社 Web サーバー(IIS)にアクセスさせる例:
"http://www.newtone.co.jp/NRDWebService/Service.asmx"

クラウドサービス Microsoft Azure の Web アプリ(App Service)に配置した Web サービスを利用する例:
"http://newtonecojp.azurewebsites.net/Service.asmx"

WebServiceUseBasicAuthentication プロパティ

【機能】
Web サービス時の基本認証の使用（デフォルト：False）を設定します。

【構文】
<VB.NET>
Public Property **WebServiceUseBasicAuthentication** As **Boolean**

<C#>
public **bool** **WebServiceUseBasicAuthentication** { set; get; }

<VC++>
public : **VARIANT_BOOL**

NewtoneNRDvcpp::INRD_Activation::WebServiceUseBasicAuthentication

【解説】
Web サーバーで基本認証を使用する場合は、この WebServiceUseBasicAuthentication プロパティを True にします。
初期値は、False（基本認証を使用しない）です。

この WebServiceUseBasicAuthentication プロパティは、Web サーバー（IIS）側で特定のアカウント（ユーザ名とパスワード）でアクセスできるフォルダにこの Web サービスが配置してある場合に使用できるセキュリティ設定です。

Web サーバーで基本認証を使用する一般的な手順は次の通りです。
1.サーバーPC 上でユーザを作成。
　この際のユーザ名とパスワードがそのまま基本認証に使われます。
2.基本認証フォルダのセキュリティ設定
　フォルダのプロパティを開き、セキュリティタブで上記 1 のユーザ名を 追加し、「読み取り」権限を付与します。
3.IIS でのセキュリティ設定
　IIS で該当フォルダに対し「認証」の設定で「匿名認証」を無効にして 「基本認証」を有効にします。

なお、Web サーバーでの基本認証の詳細につきましてはマイクロソフト社の関連ドキュメントをご覧ください。

<既定 UI 系メソッド>

メソッド一覧

メソッド	機能
ActivateRegisterInternet	「認証登録/インターネット」処理の呼び出し
ActivateRegisterTelephone	「認証登録/電話」処理の呼び出し
ActivateRemoveInternet	「認証解除/インターネット」処理の呼び出し
ActivateRemoveTelephone	「認証解除/電話」処理の呼び出し
ActivateStatusCheck	エンドユーザ PC 内での認証状態の確認
ActivateStatusCheckOnline	オンラインで認証状態の確認
ActivateStatusDisp	「認証状態表示」処理の呼び出し
ActivateStatusOnlineVerify	「認証状況オンライン表示」処理の呼び出し
DeterminationOfProxyUpdateOfExpirationDate	「代理有効期限更新/確定」処理の呼び出し
FloatingLicenseFinish	「フローティングライセンス使用終了」処理の呼び出し
FloatingLicenseStart	「フローティングライセンス使用開始」処理の呼び出し
GetProxyUpdateOfExpirationDate	「代理有効期限/取得」処理の呼び出し(代理 PC で利用)
PreparationOfProxyUpdateOfExpirationDate	「代理有効期限更新/準備」処理の呼び出し
ProxyActivateRegisterExecute	「代理認証登録/実行」処理の呼び出し(代理 PC で利用)
ProxyActivateRegisterFix	「代理認証登録/確定」処理の呼び出し
ProxyActivateRegisterPrepare	「代理認証登録/準備」処理の呼び出し
ProxyActivateRemoveExecute	「代理認証解除/実行」処理の呼び出し(代理 PC で利用)
ProxyActivateRemovePrepare	「代理認証解除/準備」処理の呼び出し
RestoreCancelStatus	「認証解除状態回復」処理の呼び出し
RestoreRegisterStatus	「認証登録状態回復」処理の呼び出し
RunNR2AppDateRemove	「アプリ起動日」を削除
TrialStartDateRemove	「猶予日数」の「開始日」を削除
UpdateOfExpirationDate	「有効期限の更新」処理の呼び出し

ActivateRegisterInternet メソッド

【機能】
「認証登録/インターネット」処理の呼び出しを行います。

【構文】
<VB.NET>
Public Function **ActivateRegisterInternet**() As **Boolean**

<C#>
public **bool** **ActivateRegisterInternet**()

<VC++>
public：**VARIANT_BOOL**

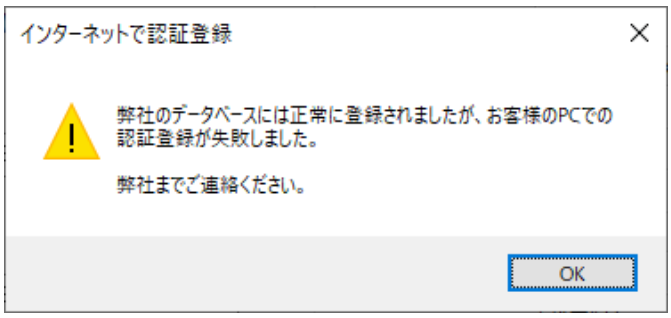
NewtoneNRDvcpp::INRD_Activation::ActivateRegisterInternet()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

【解説】
「認証登録/インターネット」処理の呼び出しを行います。

なお、何らかの原因で、データベースは登録済、エンドユーザ PC のレジストリが解除済の状態になった場合は、次のようなダイアログが表示されます。



この場合、エンドユーザ様がお使いの PC レジストリの状態をご確認の上、「認証登録状態回復メソッド」処理をしていただくか、認証管理システムの「電話認証解除の対応」の「クラッシュ（お客様の PC が動作不能になった場合）」で、プラス許可数を 1 増やしてご対応ください。

ActivateRegisterTelephone メソッド

【機能】
「認証登録/電話」処理の呼び出しを行います。

【構文】
<VB.NET>
Public Function **ActivateRegisterTelephone**() As **Boolean**

<C#>
public **bool** **ActivateRegisterTelephone**()

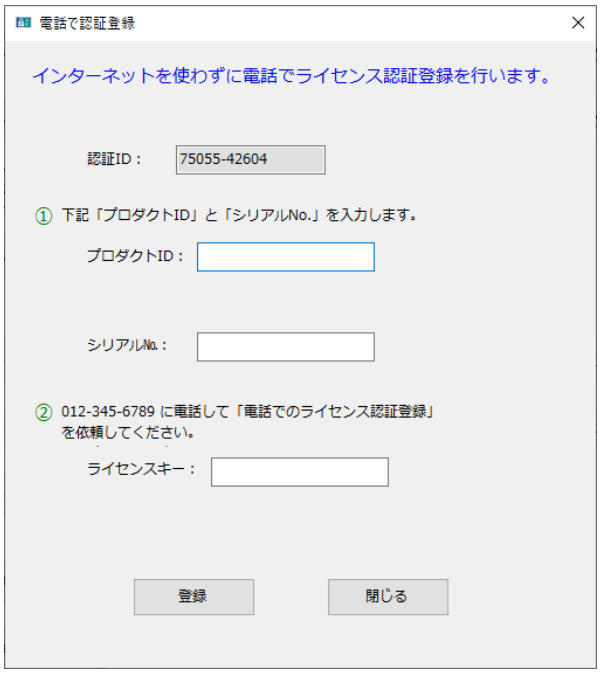
<VC++>
public：**VARIANT_BOOL**

NewtoneNRDvcpp::INRD_Activation::ActivateRegisterTelephone()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

【解説】
「認証登録/電話」処理の呼び出しを行います。



ActivateRemoveInternet メソッド

【機能】
「認証解除/インターネット」処理の呼び出しを行います。

【構文】
<VB.NET>
Public Function **ActivateRemoveInternet**() As **Boolean**

<C#>
public **bool** **ActivateRemoveInternet**()

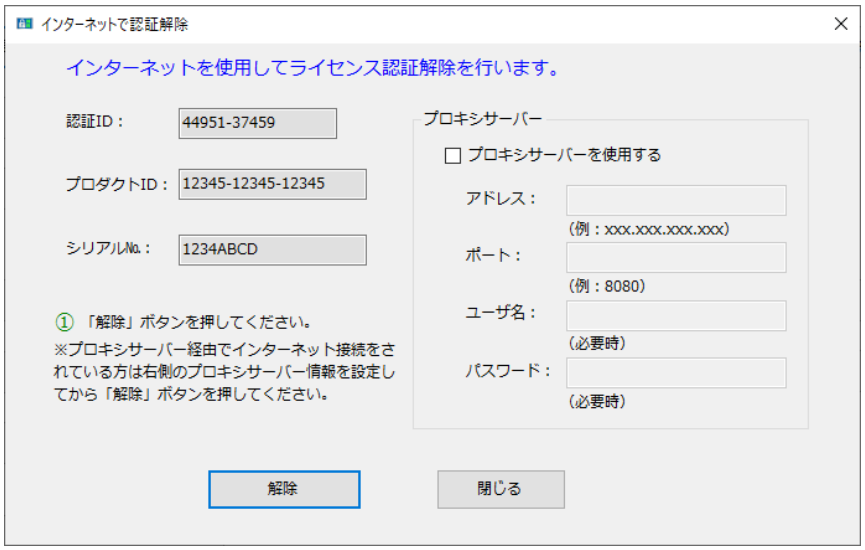
<VC++>
public：**VARIANT_BOOL**

NewtoneNRDvcpp::INRD_Activation::ActivateRemoveInternet()

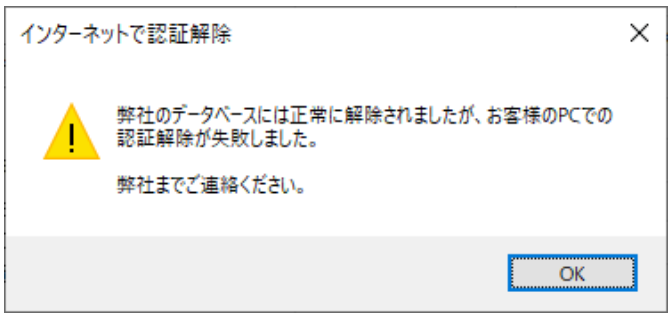
【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

【解説】
「認証解除/インターネット」処理の呼び出しを行います。



なお、何らかの原因で、データベースは解除済、エンドユーザ PC のレジストリが登録状態になった場合は、次のようなダイアログが表示されます。



この場合、エンドユーザ様がお使いのPCレジストリの状態をご確認の上、「代理認証解除−準備」処理をしていただくか、「認証解除状態回復メソッド」を行っていただくよう、お伝えください。

ActivateRemoveTelephone メソッド

【機能】
「認証解除/電話」処理の呼び出しを行います。

【構文】
<VB.NET>
Public Function **ActivateRemoveTelephone**() As **Boolean**

<C#>
public **bool** **ActivateRemoveTelephone**()

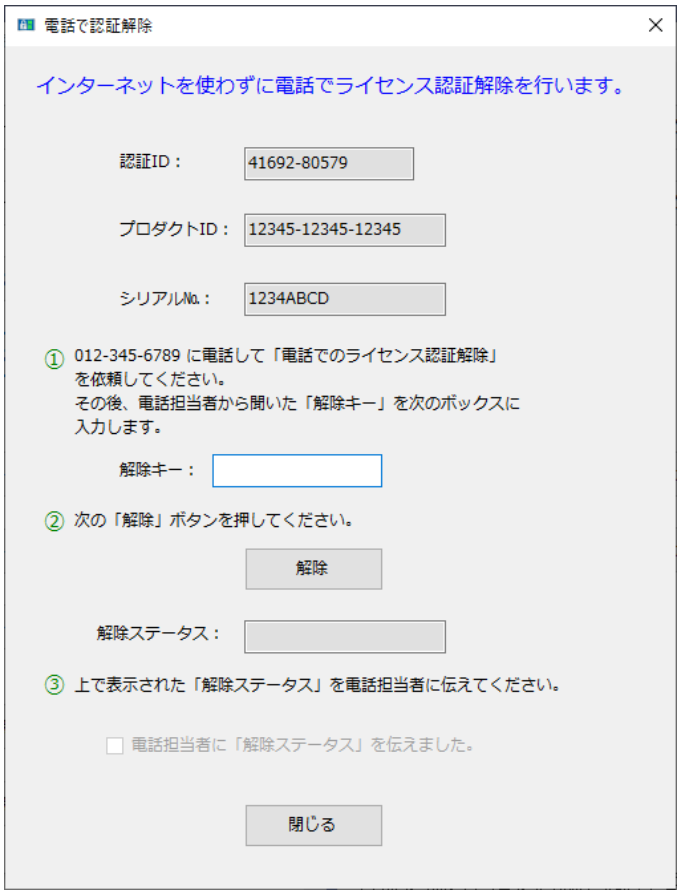
<VC++>
public：**VARIANT_BOOL**

NewtoneNRDvcpp::INRD_Activation::ActivateRemoveTelephone()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

【解説】
「認証解除/電話」処理の呼び出しを行います。



ActivateStatusCheck メソッド

【機能】
エンドユーザ PC 内での認証状態の確認を行います。

【構文】
<VB.NET>

```
Public Function ActivateStatusCheck() As Integer
```

<C#>

```
public int ActivateStatusCheck()
```

<VC++>

```
public : long NewtoneNRDvcpp::INRD_Activation::ActivateStatusCheck()
```

【引数】
なし

【戻り値】

0:	猶予期限切れ(猶予有効時)
1～365:	猶予日数有
366:	日付データの取得失敗(猶予)
400:	未認証(猶予無効時)
500:	認証済み
600:	フローティングライセンス登録済
1000:	欠番(旧レンタル機能関連)
1001～2100:	欠番(旧レンタル機能関連)
2101:	欠番(旧レンタル機能関連)
-999:	認証済ハードウェア情報不一致
-1:	エラー(未設定や範囲を超えているプロパティがある)
-21001231～-20000101:	認証済で終了した有効期限 (2000/01/01～2100/12/31)
-21001232:	日付データの取得失敗(有効期限)
-3:	PC の日付が変更されました。
20000101～21001231:	認証済でまだ有効な有効期限 (2000/01/01～2100/12/31)
200001010～210012310:	フローティングライセンスが登録済でまだ有効な有効期限 (2000/01/01～2100/12/31)
-210012310～-200001010:	フローティングライセンスが登録済で終了した有効期限 (2000/01/01～2100/12/31)

<戻り値が-1 の場合の補足説明>
本来設定が必須であるプロパティに値がセットされていない場合やセットした値が無効な場合などに(-1)が返ります。たとえば、「猶予(試用)期間の名称」用の TrialPeriodName プロパティに(猶予・試用期間機能を使用しないということで未設定にして結果として)空文字列を設定した場合や電話で認証時の電話番号用の TelephoneNumber プロパティに(電話対応機能を使用しないということで未設定にして結果として)空文字列を設定した場合などにこの戻り値(-1)が返ります。DLL のプロパティは機能の使用有無に関係なく DLL の起動時にそれらの値をチェックするようになっていて、上記のような空文字列などの場合その機能の利用時に問題が発生することをあらかじめ防止する目的で戻り値(-1)が返ります。

<戻り値が 200001010～210012310 または-210012310～-200001010 の場合>
戻り値を 10 で割って値が有効期限となります。
例:
戻り値が 202412200 の場合
202412200 ÷ 10 = 20241220
1～4 桁目が年→2024
5～6 桁目が月→12
7～8 桁目が日→20
よって、2024 年 12 月 20 日となります。

【解説】
この ActivateStatusCheck メソッドと ActivateStatusCheckOnline メソッドの相違点は ActivateStatusCheck メソッドがエンドユーザ PC 内での認証状況を返すのに対し、ActivateStatusCheckOnline メソッドはインターネットを介し貴社のデータベースにアクセスして取得した情報とエンドユーザ PC 内の情報とを照合して認証状況を返す点です。

ActivateStatusCheckOnline メソッド

【機能】
オンラインで認証状態の確認を行います。

【構文】
<VB.NET>
Public Function **ActivateStatusCheckOnline**() As **Integer**

<C#>
public **int** **ActivateStatusCheckOnline**()

<VC++>
public : **long NewtoneNRDvcpp::INRD_Activation::ActivateStatusCheckOnline()**

【引数】
なし

【戻り値】
0: PC レベルで認証されていない(PC のレジストリには認証登録情報がない)
1: OK(PC レベルと DB で認証登録情報が一致した)
2: NG(PC レベルと一致する認証登録情報が DB にない)
3: NG(認証済ハードウェア情報不一致)
4: NG(日付データの取得失敗(猶予))
5: 欠番(旧レンタル機能関連)
6: NG(日付データの取得失敗(有効期限))
-11: 接続できない(認証時は電話)
-12: 接続できない(認証時は代理)
-13: 接続できない(フローティングライセンス)
-999: その他エラー(接続できないなど)
-1: エラー(未設定や範囲を超えているプロパティがある)
-3: PC の日付が変更されました。
600: フローティングライセンス登録済

<戻り値が-1 の場合の補足説明>
本来設定が必須であるプロパティに値がセットされていない場合やセットした値が無効な場合などに(-1)が返ります。たとえば、「猶予(試用)期間の名称」用の TrialPeriodName プロパティに(猶予・試用期間機能を使用しないということで未設定にして結果として)空文字列を設定した場合や電話で認証時の電話番号用の TelephoneNumber プロパティに(電話対応機能を使用しないということで未設定にして結果として)空文字列を設定した場合などにこの戻り値(-1)が返ります。DLL のプロパティは機能の使用有無に関係なくDLL の起動時にそれらの値をチェックするようになっていて、上記のような空文字列などの場合その機能の利用時に問題が発生することをあらかじめ防止する目的で戻り値(-1)が返ります。

【解説】
この ActivateStatusCheckOnline メソッドと ActivateStatusCheck メソッドとの相違点は ActivateStatusCheck メソッドがエンドユーザ PC 内での認証状況を返すのに対し、ActivateStatusCheckOnline メソッドはインターネットを介し貴社のデータベースにアクセスして取得した情報とエンドユーザ PC 内の情報とを照合して認証状況を返す点です。

この ActivateStatusCheckOnline メソッドは、なんらかの理由で現在エンドユーザが使用している貴社のアプリケーションを使用不可としたい場合などに利用できます。

認証登録の場合、レジストリとハード情報だけの確認で OK となってしまい貴社がデータベース（DB）上の当該情報を削除してもエンドユーザの PC ではアプリケーションが継続して使用できてしまいます。そこで、任意のタイミングでインターネットを介し貴社のデータベースにアクセスしてそれらの情報を確認できる機能がこのメソッドです。貴社はたとえば、アプリケーションの起動時にそのメソッドを利用するコードを記述し、その戻り値によってアプリケーションを（強制的に）終了する、といった挙動を制御できます。

インターネットを介し Web サービスに接続できなかった場合は、次の「インターネットに接続」ダイアログを表示します。

このダイアログでのエンドユーザの選択肢は次の通りです。

・「接続」ボタン→現在の内容で再度接続する。
・「キャンセル」ボタン→このダイアログを閉じて、該当する戻り値を返してメソッド終了

なお、このダイアログ内で必要に応じてプロキシサーバーの接続設定が可能です。

ActivateStatusDisp メソッド

【機能】
「認証状態表示」処理の呼び出しを行います。

【構文】
<VB.NET>
Public Function **ActivateStatusDisp**() As **Boolean**

<C#>
public **bool** **ActivateStatusDisp**()

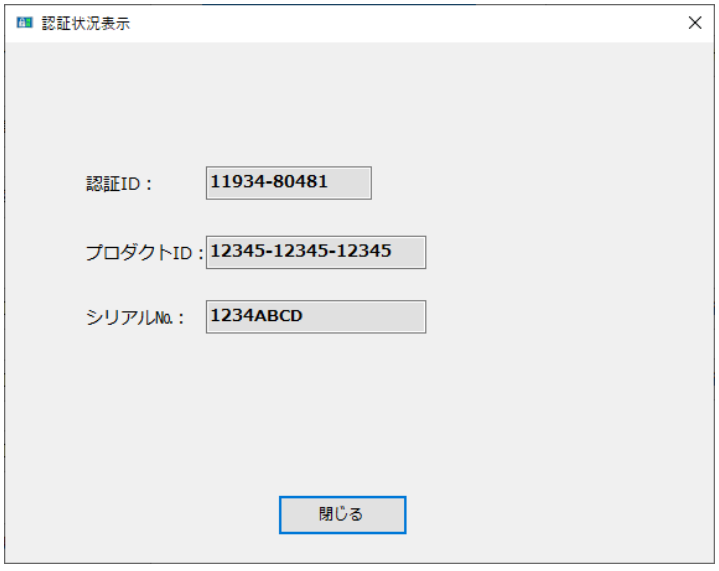
<VC++>
public : **VARIANT_BOOL NewtoneNRDvcpp::INRD_Activation::ActivateStatusDisp()**

【引数】
なし

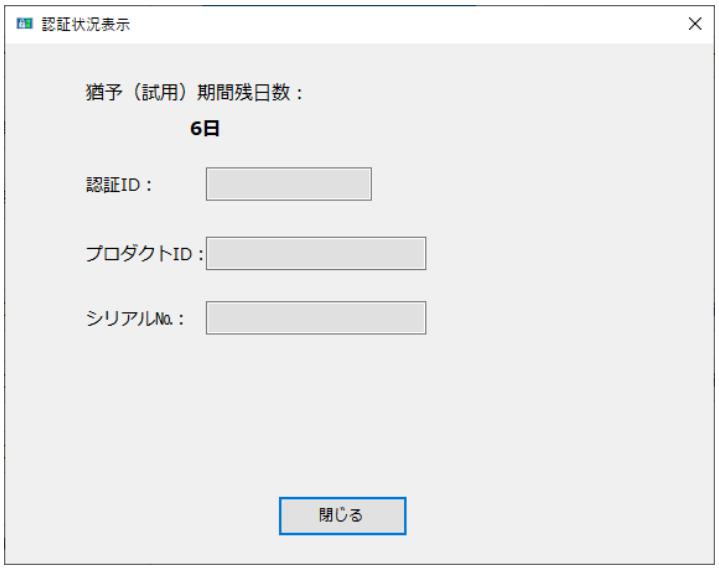
【戻り値】
True: 正常
False: エラー（未設定や範囲を超えているプロパティがある）

【解説】
エンドユーザ PC 内での認証状態の表示を行います。

呼び出したダイアログの例を以下に示します。

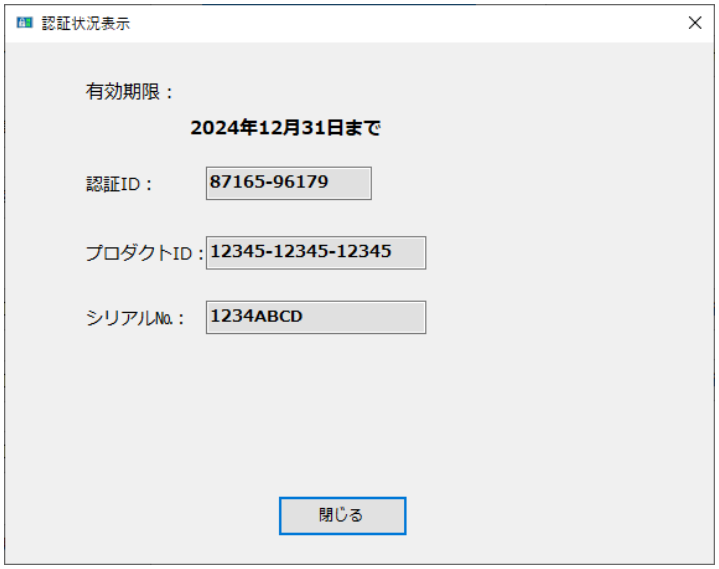


（猶予期間中）



（認証登録済み）

（有効期限中）



ActivateStatusOnlineVerify メソッド

【機能】
「認証状況オンライン表示」処理の呼び出しを行います。

【構文】
<VB.NET>
Public Function **ActivateStatusOnlineVerify**() As **Boolean**

<C#>
public **bool** **ActivateStatusOnlineVerify**()

【引数】
なし

<VC++>
public：**VARIANT_BOOL**

NewtoneNRDvcpp::INRD_Activation::ActivateStatusOnlineVerify()

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

【解説】
エンドユーザ PC 内でのフローティングライセンスの状況の表示を行います。

呼び出したダイアログの例を以下に示します。


認証状況オンライン表示
有効期限：
2024年12月31日まで
認証ID：
24003-25897
プロダクトID：
00005-00005-00005
シリアルNo.：
5555eeee
フローティングライセンス
このPCのフローティングライセンス状況：
登録済（使用中）
このライセンスの
使用状況
使用数：
2
上限数：
10
使用中　PC一覧
PC名
認証ID
1
PC00101
24003-25897
2
PC00245
97819-74573
閉じる

DeterminationOfProxyUpdateOfExpirationDate メソッド

【機能】
「代理有効期限更新/確定」処理の呼び出しを行います。
（代理有効期限更新機能については、「代理認証機能について」を参照）

【構文】
<VB.NET>
Public Function **DeterminationOfProxyUpdateOfExpirationDate**() As **Boolean**

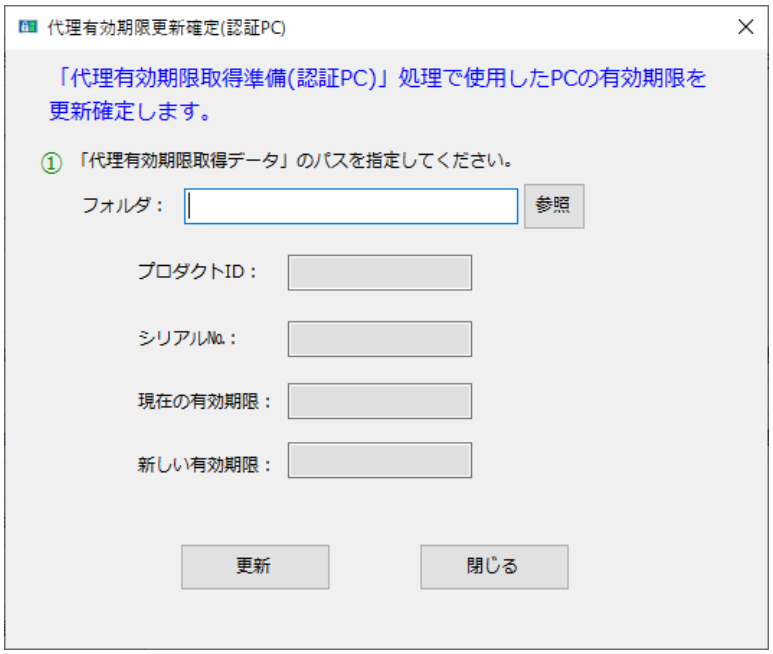
<C#>
public **bool** **DeterminationOfProxyUpdateOfExpirationDate**()

<VC++>
public：**VARIANT_BOOL**
NewtoneNRDvcpp::INRD_Activation::DeterminationOfProxyUpdateOfExpirationDate(
)

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

【解説】
「代理有効期限更新/確定」処理の呼び出しを行います。



FloatingLicenseFinish メソッド

【機能】
「フローティングライセンス使用終了」処理の呼び出しを行います。

【構文】
<VB.NET>
Public Function **FloatingLicenseFinish**() As **Boolean**

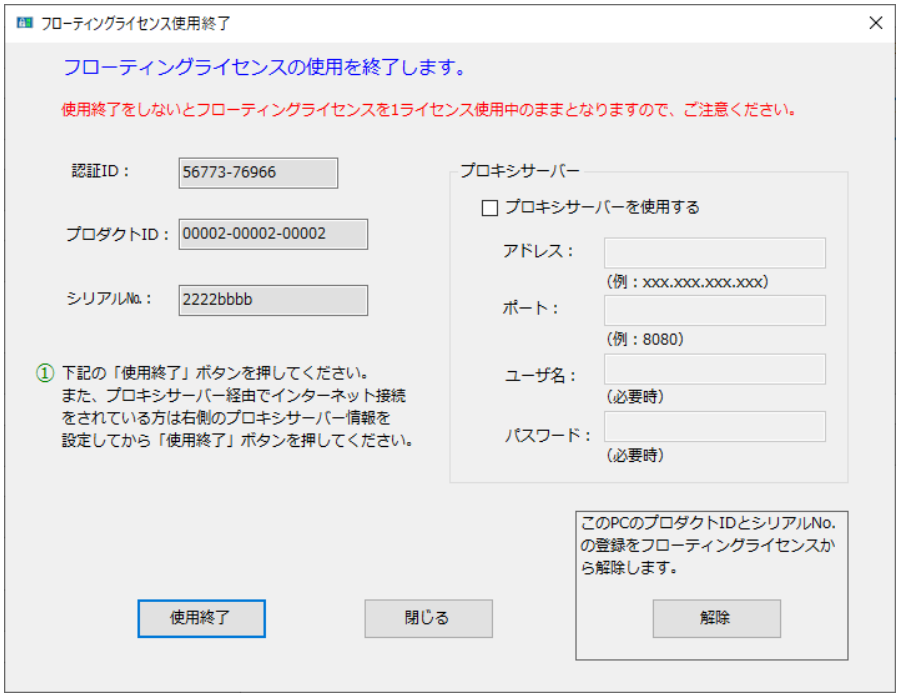
<C#>
public **bool** **FloatingLicenseFinish**()

<VC++>
public : **VARIANT_BOOL NewtoneNRDvcpp::INRD_Activation::FloatingLicenseFinish()**

【引数】
なし

【戻り値】
True: 正常
False: エラー（未設定や範囲を超えているプロパティがある）

【解説】
「フローティングライセンス使用終了」処理の呼び出しを行います。
この処理では、「使用終了」および「解除」が行えます。
「使用終了」のみ行い、「解除」を行わない場合、エンドユーザ PC にフローティングライセンス情報が登録済状態ですので、「フローティングライセンス使用開始」処理で、「プロダクト ID」と「シリアル No.」は自動的に表示されます。
また、「使用終了」を行わないとフローティングライセンスを 1 ライセンス使用中のままとなりますので、ご注意ください。



FloatingLicenseStart メソッド

【機能】
「フローティングライセンス使用開始」処理の呼び出しを行います。

【構文】
<VB.NET>
Public Function **FloatingLicenseStart**() As **Boolean**

<C#>
public **bool** **FloatingLicenseStart**()

<VC++>
public : **VARIANT_BOOL NewtoneNRDvcpp::INRD_Activation::FloatingLicenseStart()**

【引数】
なし

【戻り値】
True: 正常
False: エラー（未設定や範囲を超えているプロパティがある）

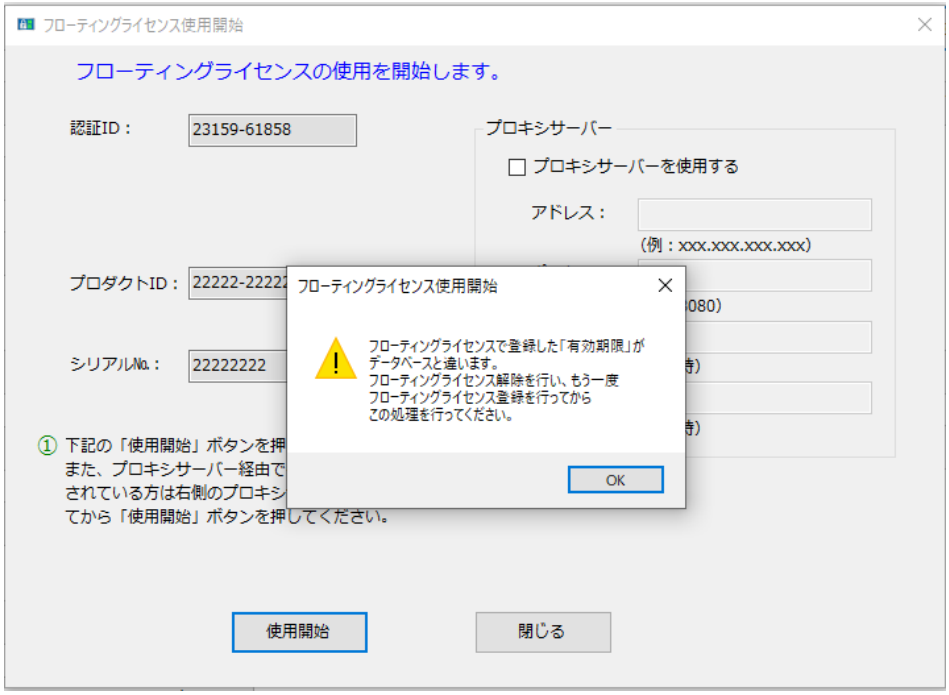
【解説】
「フローティングライセンス使用開始」処理の呼び出しを行います。
「フローティングライセンス使用開始」処理を初めて行う場合、「プロダクト ID」と「シリアル No.」の入力が必要になります。その後は、「フローティングライセンス解除」処理を行わない限り、入力は不要です。
入力されたプロダクト ID とシリアル No.がデータベースに存在するか確認後、エンドユーザ PC に情報を登録します。その後、使用開始します。
このメソッドは、TrialPeriod プロパティが 0 以上に設定されている場合、実行できません。

＜初めてのフローティングライセンス使用開始時＞


フローティングライセンス使用開始
フローティングライセンスの使用を開始します。
認証ID：
36328-19888
最初の使用開始です。
最初の1回だけ、プロダクトIDとシリアルNo.を登録します。
プロダクトID：
シリアルNo.：
① 上記の「プロダクトID」と「シリアルNo.」を入力後に、下記の「使用開始」ボタンを押してください。
また、プロキシサーバー経由でインターネット接続をされている方は右側のプロキシサーバー情報を設定してから「使用開始」ボタンを押してください。
プロキシサーバー
プロキシサーバーを使用する
アドレス：
isa01-out.example.local
(例：xxx.xxx.xxx.xxx)
ポート：
8080
(例：8080)
ユーザ名：
taro
(必要時)
パスワード：
******
(必要時)
使用開始
閉じる

＜フローティングライセンス解除がされていない場合＞

たとえば、「フローティングライセンス使用開始」処理を初めて行った時に設定されていた「有効期限」が 2024 年 7 月 31 日だとします。
その後、データベース側で「有効期限」を 2024 年 12 月 31 日に更新された場合、「フローティングライセンス使用開始」処理を実行すると次のメッセージが表示されます。
この場合、「フローティングライセンス解除」を行ってから、再度、当処理を行ってください。



GetProxyUpdateOfExpirationDate メソッド

【機能】
「代理有効期限/取得」処理の呼び出し(代理 PC で利用)を行います。
(代理有効期限更新機能については、「代理認証機能について」を参照)

【構文】
<VB.NET>
Public Function **GetProxyUpdateOfExpirationDate** () As **Boolean**

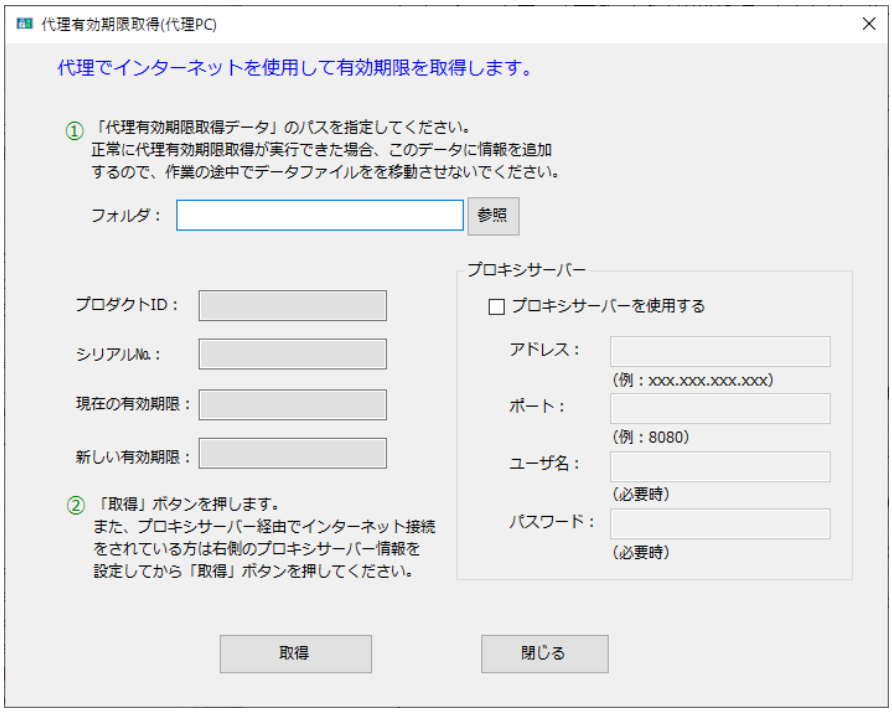
<C#>
public **bool** **GetProxyUpdateOfExpirationDate** ()

<VC++>
public : **VARIANT_BOOL**
NewtoneNRDvcpp::INRD_Activation::GetProxyUpdateOfExpirationDate()

【引数】
なし

【戻り値】
True: 正常
False: エラー(未設定や範囲を超えているプロパティがある)

【解説】
「代理有効期限/取得」処理の呼び出し(代理 PC で利用)を行います。



PreparationOfProxyUpdateOfExpirationDate メソッド

【機能】
「代理有効期限更新/準備」処理の呼び出しを行います。
（代理有効期限更新機能については、「代理認証機能について」を参照）

【構文】
<VB.NET>
Public Function **PreparationOfProxyUpdateOfExpirationDate**() As **Boolean**

<C#>
public **bool** **PreparationOfProxyUpdateOfExpirationDate** ()

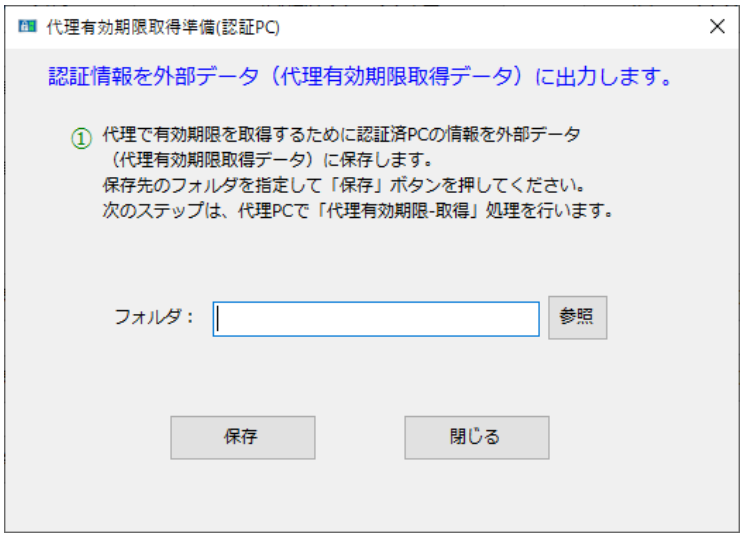
<VC++>
public：**VARIANT_BOOL**

NewtoneNRDvcpp::INRD_Activation::PreparationOfProxyUpdateOfExpirationDate()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

【解説】
「代理有効期限更新/準備」処理の呼び出しを行います。



ProxyActivateRegisterExecute メソッド

【機能】
「代理認証登録/実行」処理の呼び出し(代理 PC で利用)を行います。
(代理認証機能については、「代理認証機能について」を参照)

【構文】
<VB.NET>
Public Function **ProxyActivateRegisterExecute**() As **Boolean**

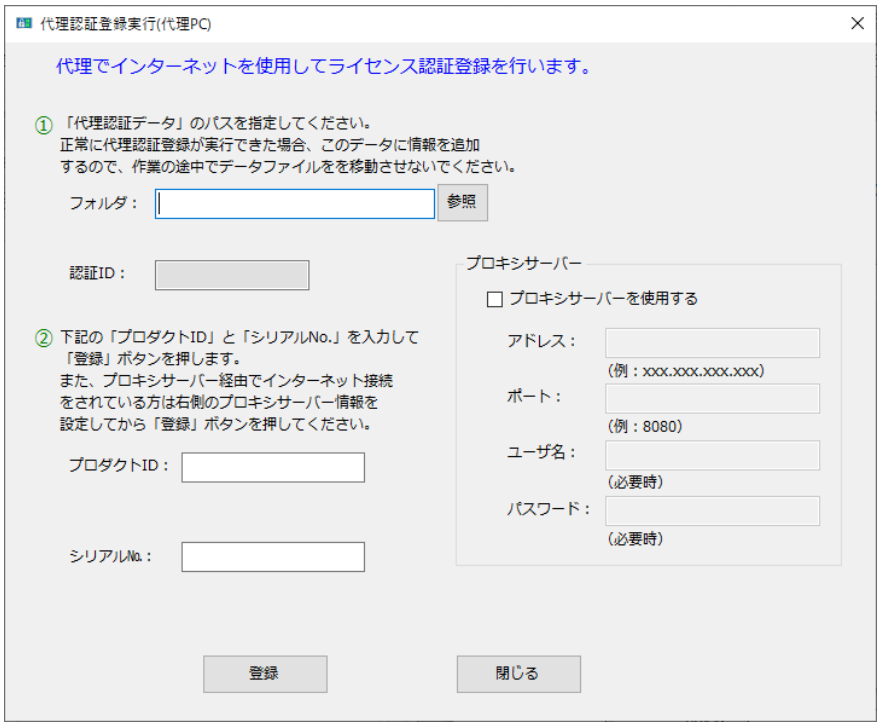
<C#>
public **bool** **ProxyActivateRegisterExecute**()

<VC++>
public : **VARIANT_BOOL**
NewtoneNRDvcpp::INRD_Activation::ProxyActivateRegisterExecute()

【引数】
なし

【戻り値】
True: 正常
False: エラー(未設定や範囲を超えているプロパティがある)

【解説】
「代理認証登録/実行」処理の呼び出し(代理 PC で利用)を行います。



ProxyActivateRegisterFix メソッド

【機能】
「代理認証登録/確定」処理の呼び出しを行います。
（代理認証機能については、「代理認証機能について」を参照）

【構文】
<VB.NET>
Public Function **ProxyActivateRegisterFix**() As **Boolean**

<C#>
public **bool** **ProxyActivateRegisterFix**()

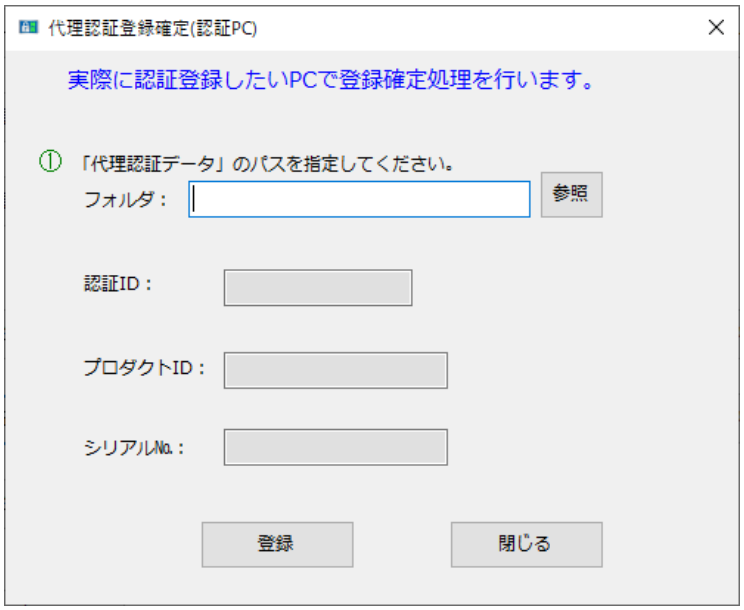
<VC++>
public：**VARIANT_BOOL**

NewtoneNRDvcpp::INRD_Activation::ProxyActivateRegisterFix()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

【解説】
「代理認証登録/確定」処理の呼び出しを行います。



ProxyActivateRegisterPrepare メソッド

【機能】
「代理認証登録/準備」処理の呼び出しを行います。
（代理認証機能については、「代理認証機能について」を参照）

【構文】
<VB.NET>
Public Function **ProxyActivateRegisterPrepare**() As **Boolean**

<C#>
public **bool** **ProxyActivateRegisterPrepare**()

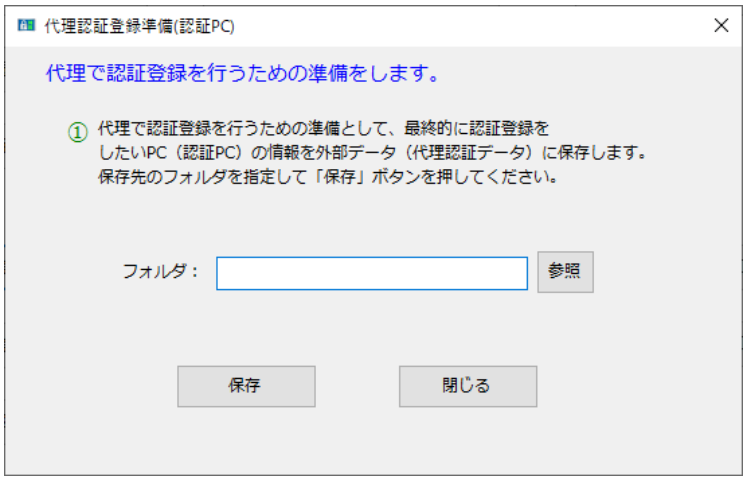
<VC++>
public：**VARIANT_BOOL**

NewtoneNRDvcpp::INRD_Activation::ProxyActivateRegisterPrepare()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

【解説】
「代理認証登録/準備」処理の呼び出しを行います。



ProxyActivateRemoveExecute メソッド

【機能】
「代理認証解除/実行」処理の呼び出し(代理 PC で利用)を行います。
(代理認証機能については、「代理認証機能について」を参照)

【構文】
<VB.NET>
Public Function **ProxyActivateRemoveExecute**() As **Boolean**

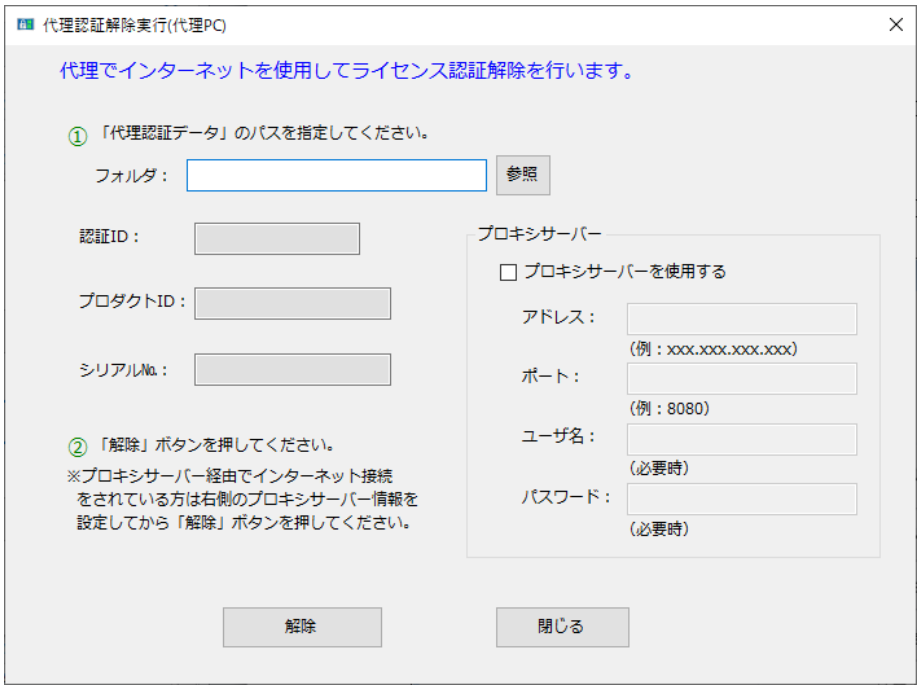
<C#>
public **bool** **ProxyActivateRemoveExecute**()

<VC++>
public : **VARIANT_BOOL**
NewtoneNRDvcpp::INRD_Activation::ProxyActivateRemoveExecute()

【引数】
なし

【戻り値】
True: 正常
False: エラー(未設定や範囲を超えているプロパティがある)

【解説】
「代理認証解除/実行」処理の呼び出し(代理 PC で利用)を行います。



ProxyActivateRemovePrepare メソッド

【機能】
「代理認証解除/準備」処理の呼び出しを行います。
（代理認証機能については、「代理認証機能について」を参照）

【構文】
<VB.NET>
Public Function **ProxyActivateRemovePrepare**() As **Boolean**

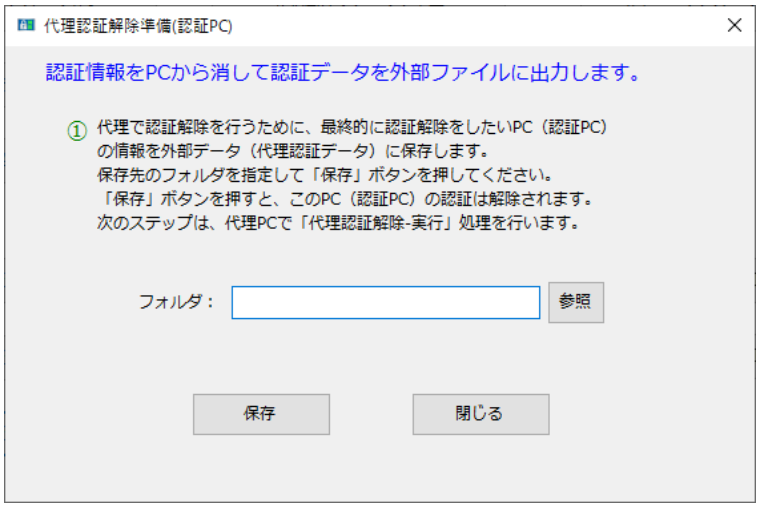
<C#>
public **bool** **ProxyActivateRemovePrepare**()

<VC++>
public：**VARIANT_BOOL**
NewtoneNRDvcpp::INRD_Activation::ProxyActivateRemovePrepare()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

【解説】
「代理認証解除/準備」処理の呼び出しを行います。



RestoreCancelStatus メソッド

【機能】
「認証解除状態回復」処理の呼び出しを行います。

【構文】
<VB.NET>
Public Function **RestoreCancelStatus**() As **Boolean**

<C#>
public **bool** **RestoreCancelStatus**()

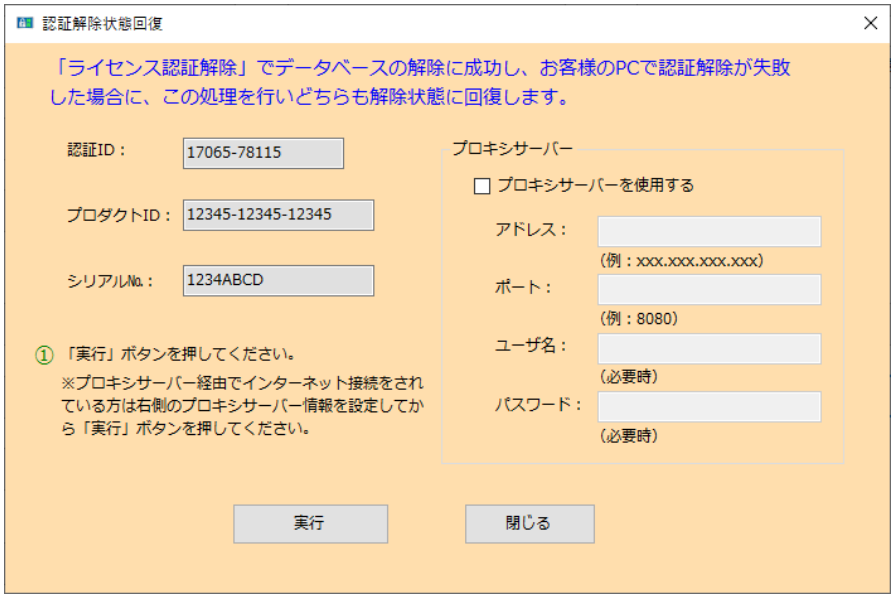
<VC++>
public：**VARIANT_BOOL NewtoneNRDvcpp::INRD_Activation::RestoreCancelStatus()**

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

【解説】
「認証解除状態回復」処理の呼び出しを行います。

「ライセンス認証解除」でデータベースの解除に成功したが、エンドユーザ PC での認証解除が失敗した状態になった場合に、このメソッドによる処理でエンドユーザの操作でその状態を回復しエンドユーザ PC を認証解除状態とします。



RestoreRegisterStatus メソッド

【機能】
「認証登録状態回復」処理の呼び出しを行います。

【構文】
<VB.NET>
Public Function **RestoreRegisterStatus**() As **Boolean**

<C#>
public **bool** **RestoreRegisterStatus**()

<VC++>
public：**VARIANT_BOOL**

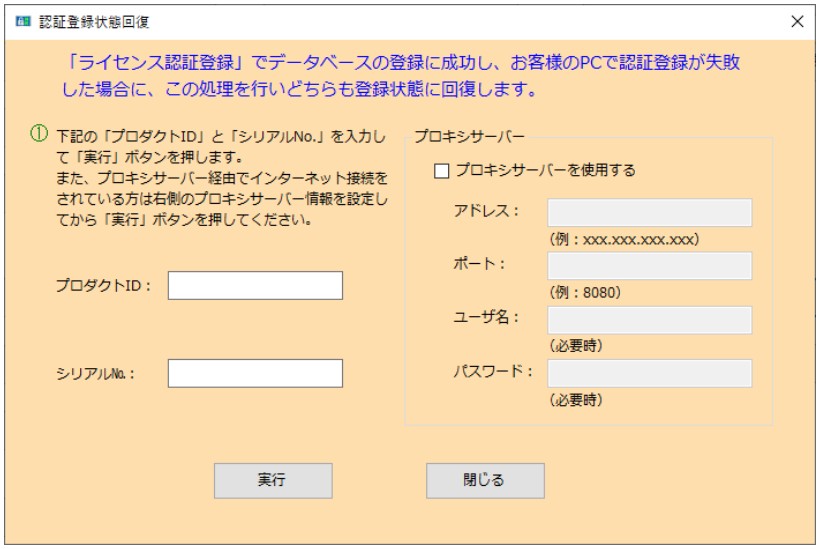
NewtoneNRDvcpp::INRD_Activation::RestoreRegisterStatus()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

【解説】
「認証登録状態回復」処理の呼び出しを行います。

「ライセンス認証登録」でデータベースの登録に成功したが、エンドユーザ PC での認証登録が失敗した状態になった場合に、このメソッドによる処理でエンドユーザの操作でその状態を回復しエンドユーザ PC を認証登録状態とします。



RunNR2AppDateRemove メソッド

【機能】
「アプリ起動日」を削除します。

【構文】
<VB.NET>
Public Function **RunNR2AppDateRemove**() As **Boolean**

<C#>
public **bool** **RunNR2AppDateRemove**()

<VC++>
public : **VARIANT_BOOL**

NewtoneNRDvcpp::INRD_Activation::RunNR2AppDateRemove()

【引数】
なし

【戻り値】
True: 正常
False: エラー（未設定や範囲を超えているプロパティがある）

【解説】
「アプリ起動日」を削除します。
認証レスキュー！を含む貴社のアプリを起動した際に、レジストリに記録されている前回の「アプリ起動日」を確認します。
（もし、「アプリ起動日」が存在しなかった場合は、起動した際にレジストリに記録します。）
その「アプリ起動日」より PC の日付が古い場合、故意に PC の日付が変更されたということで「PC の日付が変更されました。）」というメッセージが表示されます。

例:
たとえば、本日を 6 月 1 日とします。PC の日付を 6 月 2 日に設定しアプリを起動します。
次に、PC の日付を本日（つまり 6 月 1 日）に戻しても、「アプリ起動日」は 6 月 2 日で記録されており、PC の日付が古いので「PC の日付が変更されました。」というメッセージが表示されます。
この状態になった場合、PC の日付を 6 月 2 日に設定しない限り、認証登録等が実行できません。
このメソッドを実行後は、再度 PC の日付を本日（つまり 6 月 1 日）に設定して、アプリをご利用いただける状態にします。

TrialStartDateRemove メソッド

【機能】
「猶予日数」の「開始日」を削除します。

【構文】
<VB.NET>
Public Function **TrialStartDateRemove**() As **Boolean**

<C#>
public **bool** **TrialStartDateRemove**()

<VC++>
public : **VARIANT_BOOL NewtoneNRDvcpp::INRD_Activation::TrialStartDateRemove()**

【引数】
なし

【戻り値】
True:　正常
False:　エラー（未設定や範囲を超えているプロパティがある）

【解説】
「猶予日数」の「開始日」を削除します。
このメソッドを実行後は、エンドユーザは再度「猶予日数」分の利用が可能になります。

UpdateOfExpirationDate メソッド

【機能】
「有効期限の更新」処理の呼び出しを行います。

【構文】
<VB.NET>
Public Function **UpdateOfExpirationDate**() As **Boolean**

<C#>
public **bool** **UpdateOfExpirationDate**()

<VC++>
public：**VARIANT_BOOL**

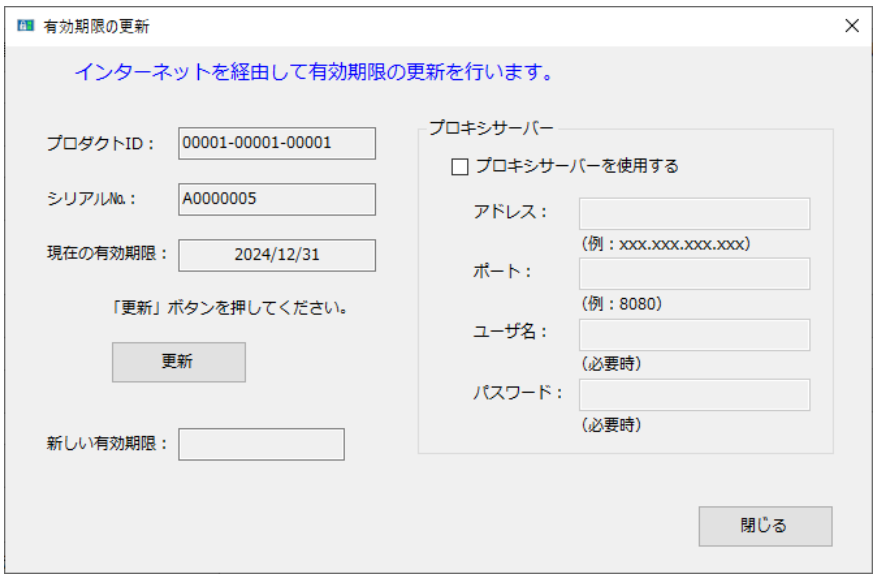
NewtoneNRDvcpp::INRD_Activation::UpdateOfExpirationDate()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

【解説】
「有効期限の更新」処理の呼び出しを行います。

プロダクト ID とシリアル No.が有効期限によるライセンスの場合、当処理を利用してエンドユーザに有効期限を更新させることができます。
この処理をエンドユーザに実行してもらう前に、貴社で新しい有効期限の設定をする必要があります。
有効期限の設定は認証管理システムの「認証キー編集（表形式）」処理を利用します。



なお、エンドユーザに有効期限を更新させる方法として当処理を実行させる他に、エンドユーザに一

度認証の解除後、再度認証の登録をしてもらうことでも更新が完了します。
エンドユーザが代理認証を利用している場合で、有効期限によるライセンスを更新する場合はその方法を使います。

<カスタム UI 系プロパティ>

プロパティー覧

プロパティ	機能
APIError 列挙体	エラー内容を表します。
APICertificationID	認証 ID（取得専用）
APICurrentExpirationDate	現在の有効期限（取得専用）
APIDisabledNICIgnore	処理を高速化するため、無効になっている NIC（Network Interface Card）を無視します。
APIEncryptionPassword	暗号化時のパスワードを設定
APIEncryptionSaltString	暗号化時の Salt 文字列を設定
APIErrorStatus	カスタム UI 系のメソッドを使用した際のエラーの内容を返す。（取得専用）
APIExternalLinkKey	APIGetProductIdSerialNoList メソッド 実行時の外部データベースとのリンク用キー項目を設定
APIFloatingLicenseDataInfo	フローティングライセンス使用中の PC 一覧を文字列として取得します。（取得専用）
APIFloatingLicenseMaxCount	フローティングライセンスの最大ライセンス数（取得専用）
APIFloatingLicenseState	フローティングライセンス状況（取得専用）
APIFloatingLicenseUsedCount	フローティングライセンスのライセンス使用数（取得専用）
APIFreeItem1～5	APIGetFreeItem メソッド 実行後に自由入力項目 1～5 が設定される（取得専用）
APILicenseKey	ライセンスキー
APINewExpirationDate	新しい有効期限を取得します。（取得専用）
APIOverwriteModeOfExpirationDateUpdate	「有効期限の更新」実行時の有効期限新旧上書きの設定
APIProductID	プロダクト ID
APIProductIdSerialNoList	APIGetProductIdSerialNoList メソッド実行後にプロダクト ID とシリアル No.のペアをデリミタで列挙したの文字列が設定される（取得専用）
APIProxyDataPath	代理認証データパス
APIProxyServerAddress	プロキシサーバーのアドレス
APIProxyServerPassword	プロキシサーバーのパスワード
APIProxyServerPort	プロキシサーバーのポート
APIProxyServerUserName	プロキシサーバーのユーザ名
APIReleaseKey	解除キー
APIReleaseStatus	解除ステータス（取得専用）
APISelectRunAppDatePathFlag	「アプリ起動日」の読み込み/書き込み場所の切り替えフラグ
APISerialNo	シリアル No.
APITrialPeriod	猶予（試用）日数を設定
APIUseCpuInfo	CPU 情報の使用を設定
APIUseMacAddress	MAC アドレスの使用を設定
APIUseProxyServer	プロキシサーバーの使用区分
APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパスを設定

APIWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワードを設定
APIWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名を設定
APIWebServiceCheckPassword	Web サービス確認パスワードを設定
APIWebServiceTimeout	Web サービスのタイムアウトを設定
APIWebServiceURL	Web サービスの URL を設定
APIWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用を設定

APIError 列挙体

エラー内容を表します。
public enum APIError
パブリックメンバ

メンバ名	値	内容	関連するメソッド
None	0	エラーなし	
OutOfMemoryException	11	プログラムの実行を継続するためのメモリが不足している場合にスローされる例外。	
StackOverflowException	12	入れ子になったメソッド呼び出しが多くなりすぎ、実行スタックがオーバーフローした場合にスローされる例外。 このクラスは継承できません。	
UnauthorizedAccessException	13	オペレーティング システムが I/O エラーまたは特定の種類のセキュリティ エラーのためにアクセスを拒否する場合、スローされる例外。	
IoDirectoryNotFoundException	14	ファイルまたはディレクトリの一部が見つからない場合にスローされる例外。	
IoDriveNotFoundException	15	使用できないドライブまたは共有にアクセスしようとするとスローされる例外。	
IoEndOfStreamException	16	ストリームの末尾を越えて読み取ろうとしたときにスローされる例外。	
IoFileLoadException	17	マネージ アセンブリが見つかったが、読み込むことができない場合にスローされる例外。	
IoFileNotFoundException	18	ディスク上に存在しないファイルにアクセスしようとして失敗したときにスローされる例外。	
IoIOException	19	I/O エラーが発生したときにスローされる例外。	
IoPathTooLongException	20	パス名またはファイル名がシステム定義の最大長を超えている場合にスローされる例外。	
BadStrInProductID	101	プロダクト ID に不正な文字が含まれています。	APIActivateRegisterInternet APIActivateRegisterTelephone APIProxyActivateRegisterExecute APIRestoreRegisterStatus APIUpdateOfExpirationDate APIGetFreeItem APIGetRegisteredInfoFromWeb

			APIGetProxyUpdateOfExpirationDate APIFloatingLicenseRegister APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicenseFinish
BadStrInSerialNo	102	シリアル No.に不正な文字が含まれています。	APIActivateRegisterInternet APIActivateRegisterTelephone APIProxyActivateRegisterExecute APIRestoreRegisterStatus APIUpdateOfExpirationDate APIGetFreeItem APIGetRegisteredInfoFromWeb APIGetProxyUpdateOfExpirationDate APIFloatingLicenseRegister APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicenseFinish
BadInputData	103	入力されたデータが不正です。	APIActivateRegisterInternet APIActivateRegisterTelephone APIProxyActivateRegisterExecute APIProxyActivateRegisterFix APIProxyActivateRemoveExecute APIFloatingLicenseRegister APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicense

			Finish
FailedEnableNIC	104	NIC の設定に失敗しました。(有効化)	APIActivateRegisterInternet APIActivateRegisterTelephone APIActivateRemoveInternet APIProxyActivateRegisterFix APIProxyActivateRegisterPrepare APIRestoreRegisterStatus APIGetProxyDataForRegister APIFloatingLicenseRegister
FailedDisableNIC	105	NIC の設定に失敗しました。(無効化)	APIActivateRegisterInternet APIActivateRegisterTelephone APIActivateRemoveInternet APIProxyActivateRegisterFix APIProxyActivateRegisterPrepare APIRestoreRegisterStatus APIGetProxyDataForRegister APIFloatingLicenseRegister
NoMACAdress	106	MAC アドレスが 1 つも取得できませんでした。	APIActivateRegisterInternet APIActivateRegisterTelephone APIActivateRemoveInternet APIProxyActivateRegisterFix APIProxyActivateRegisterPrepare APIRestoreRegisterStatus APIGetProxyDataForRegister APIFloatingLicenseRegister
ActivationFailedException	107	弊社のデータベースには正常に登録	APIActivateRegist

		されましたが、お客様の PC での認証登録が失敗しました。	erInternet
ExceedLicense	108	ライセンス数を超えています。	APIActivateRegisterInternet APIProxyActivateRegisterExecute APIFloatingLicenseStart
UnregisteredException	109	指定されたプロダクト ID とシリアル№は弊社データベースに登録されていません。	APIActivateRegisterInternet APIProxyActivateRegisterExecute APIUpdateOfExpirationDate APIGetFreeItem APIGetRegisteredInfoFromWeb APIGetProxyUpdateOfExpirationDate APIFloatingLicenseRegister APIFloatingLicenseStart
NotRegisteredException	110	何らかの原因で登録できませんでした。	APIActivateRegisterInternet APIProxyActivateRegisterExecute APIUpdateOfExpirationDate APIGetRegisteredInfoFromWeb APIGetProxyUpdateOfExpirationDate
AlreadyRegistered	111	そのプロダクト ID + シリアル№. + 認証 ID は既に登録されているので登録できませんでした。	APIActivateRegisterInternet APIProxyActivateRegisterExecute APIFloatingLicenseStart
BadCheckPassword	112	確認パスワードが不正です。	APIActivateRegisterInternet APIProxyActivateRegisterExecute APIActivateRemoveInternet APIProxyActivateRemoveExecute APIRestoreCancelStatus APIRestoreRegiste

			rStatus APIUpdateOfExpirationDate APIGetProductIdSerialNoList APIGetFreeItem APIGetRegisteredInfoFromWeb APIGetProxyUpdateOfExpirationDate APIFloatingLicenseRegister APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicenseFinish APIActivateStatusOnlineVerify
BadWaiFile	113	WebServEnv.wai ファイルに問題があります。 認証 Web サービス実行 PC 上で「認証レスキュー！.NET Web 環境設定」（WebAdmin.exe）が行われていない可能性があります。	APIActivateRegisterInternet APIProxyActivateRegisterExecute APIActivateRemoveInternet APIProxyActivateRemoveExecute APIRestoreCancelStatus APIRestoreRegisterStatus APIUpdateOfExpirationDate APIGetProductIdSerialNoList APIGetFreeItem APIGetRegisteredInfoFromWeb APIGetProxyUpdateOfExpirationDate APIFloatingLicenseRegister APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicenseFinish APIActivateStatus

			OnlineVerify
欠番	114	旧登録ライセンス関連。	
欠番	115	旧登録ライセンス関連	
ExpirationDateIsOutOfRange	116	弊社のデータベース上の「有効期限」が無効（範囲外など）になっていて更新できませんでした。 弊社にご連絡ください。	APIActivateRegisterInternet APIProxyActivateRegisterExecute APIRestoreRegisterStatus APIUpdateOfExpirationDate APIGetProxyUpdateOfExpirationDate APIFloatingLicenseStart
ExpirationDateIsOldDate	117	弊社のデータベース上の「有効期限」が無効（前日以前）になっていて更新できませんでした。 弊社にご連絡ください。	APIActivateRegisterInternet APIProxyActivateRegisterExecute APIRestoreRegisterStatus APIUpdateOfExpirationDate APIGetProxyUpdateOfExpirationDate APIFloatingLicenseStart
Authenticated	118	既に認証済のため、この処理は無効です。	APIActivateRegisterInternet APIActivateRegisterTelephone APIProxyActivateRegisterFix APIProxyActivateRegisterPrepare APIRestoreRegisterStatus APIGenerationOfNewCertificationID APIFloatingLicenseRegister APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicenseFinish
UnhandledProcess	119	何らかの原因のため、この処理は無効です。	APIActivateRegisterInternet APIActivateRegist

			erTelephone APIActivateRemoveInternet APIActivateRemoveTelephone APIProxyActivateRegisterFix APIProxyActivateRegisterPrepare APIProxyActivateRemovePrepare APIRestoreCancelStatus APIRestoreRegisterStatus APIUpdateOfExpirationDate APIGenerationOfNewCertificationID APIGetRegisteredInfoFromRegistry APIGetProductIdSerialNoList APIGetFreeItem APIPreparationOfProxyUpdateOfExpirationDate APIDeterminationOfProxyUpdateOfExpirationDate APIFloatingLicenseRegister APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicenseFinish
UnexpectedError	120	認証状況確認中に想定外のエラーが発生しました″	APIActivateRegisterInternet APIActivateRegisterTelephone APIActivateRemoveInternet APIActivateRemoveTelephone APIProxyActivateRegisterFix APIProxyActivateRegisterPrepare

			APIProxyActivateRemovePrepare APIRestoreCancelStatus APIRestoreRegisterStatus APIUpdateOfExpirationDate APIGenerationOfNewCertificationID APIGetRegisteredInfoFromRegistry APIPreparationOfProxyUpdateOfExpirationDate APIDeterminationOfProxyUpdateOfExpirationDate APIFloatingLicenseRegister APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicenseFinish
NoExpirationDateIsUsed	121	このプロダクト ID とシリアル No.には有効期限の利用設定はありません。	APIUpdateOfExpirationDate APIPreparationOfProxyUpdateOfExpirationDate APIGetProxyUpdateOfExpirationDate APIDeterminationOfProxyUpdateOfExpirationDate
欠番	122	旧レンタル機能関連	
BadLicenseKey	123	入力されたデータが不正です。	APIActivateRegisterTelephone
ActivationFailed	124	お客様の PC での認証登録が失敗しました。 再度、当処理を実行してください。	APIActivateRegisterTelephone APIProxyActivateRegisterFix
DeauthorizeFailedException	125	弊社のデータベースには正常に解除されましたが、お客様の PC での認証解除が失敗しました。	APIActivateRemoveInternet
ProductIDAndSerialNoAndCertificationIDNotFound	126	指定されたプロダクト ID/シリアル№/認証 ID は弊社データベースに登録されていません。	APIActivateRemoveInternet APIProxyActivateRemoveExecute

			APIFloatingLicense Finish
NotDeauthorized	127	何らかの原因で解除できませんでした。	APIActivateRemoveInternet APIProxyActivateRemoveExecute
UnauthenticatedByTrialPeriodOut	128	認証登録されていないため、この処理は無効です。	APIActivateRemoveInternet APIActivateRemoveTelephone APIProxyActivateRemovePrepare APIRestoreCancelStatus APIUpdateOfExpirationDate APIGetRegisteredInfoFromRegistry APIPreparationOfProxyUpdateOfExpirationDate APIDeterminationOfProxyUpdateOfExpirationDate
UnauthenticatedByTrialPeriod	129	認証登録されていないため、この処理は無効です。	APIActivateRemoveInternet APIActivateRemoveTelephone APIProxyActivateRemovePrepare APIRestoreCancelStatus APIUpdateOfExpirationDate APIGetRegisteredInfoFromRegistry APIPreparationOfProxyUpdateOfExpirationDate APIDeterminationOfProxyUpdateOfExpirationDate
Unauthenticated	130	認証登録されていないため、この処理は無効です。	APIActivateRemoveInternet APIActivateRemoveTelephone APIProxyActivateRemovePrepare APIRestoreCancelStatus

			APIUpdateOfExpirationDate APIGetRegisteredInfoFromRegistry APIPreparationOfProxyUpdateOfExpirationDate APIDeterminationOfProxyUpdateOfExpirationDate
HardwareInformationMismatch	131	認証済ハードウェア情報が不一致のため、この処理は無効です。	APIActivateRemoveInternet APIActivateRemoveTelephone APIProxyActivateRemovePrepare APIRestoreCancelStatus APIUpdateOfExpirationDate APIGetRegisteredInfoFromRegistry APIPreparationOfProxyUpdateOfExpirationDate APIDeterminationOfProxyUpdateOfExpirationDate APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicenseFinish
ReleasekeyIsEmpty	132	解除キーが不正です。	APIActivateRemoveTelephone
ReleasekeyDigitsProblem	133	解除キーが不正です。	APIActivateRemoveTelephone
BadReleasekey	134	解除キーが不正です。	APIActivateRemoveTelephone
DeauthorizeFailed	135	お客様の PC での認証解除が失敗しました。 再度、当処理を実行してください。	APIActivateRemoveTelephone
ProxyAuthenticationDataFileNotFound	136	代理認証データファイルが存在しません。	APIProxyActivateRegisterExecute APIProxyActivateRegisterFix APIProxyActivateRemoveExecute APIProxyActivateR

			egisterExecute APIProxyActivateR emoveExecute APIGetProxyDataF orExpirationDate APIGetProxyUpdat eOfExpirationDate APIDetermination OfProxyUpdateOfE xpirationDate APIGetProxyDataF orRegister APIGetProxyDataF orRemove
NoDataInProxydatafile	137	代理認証データファイルにデータが存在しません。	APIProxyActivateR egisterExecute APIProxyActivateR egisterFix APIProxyActivateR emoveExecute APIGetProxyDataF orRegister APIGetProxyDataF orRemove
IncorrectProxyData	138	代理認証データファイルの内容が不正です。	APIProxyActivateR egisterExecute APIProxyActivateR egisterFix APIGetProxyDataF orRegister
FailedReadProxydata	139	代理認証データの読み込みに失敗しました。	APIProxyActivateR egisterExecute APIProxyActivateR emoveExecute APIProxyActivateR egisterFix APIGetProxyDataF orRegister APIGetProxyDataF orRemove
FailedToWriteProxyData	140	弊社のデータベースには正常に登録されましたが、代理認証データの書き込みが失敗しました。 弊社のデータベースの登録を解除した上で、再度、当処理を実行していただく必要がありますので弊社までご連絡ください。	APIProxyActivateR egisterExecute
FailedWithEncryption	141	暗号化で失敗しました。	APIProxyActivateR egisterExecute APIProxyActivateR

			egisterPrepare APIProxyActivateR emoveExecute APIProxyActivateR emovePrepare APIWriteProxyServ erInfoToRegistry APIPreparationOfP roxyUpdateOfExpir ationDate APIGetProxyUpdat eOfExpirationDate
ProxyDataInconsistency	142	「代理認証データ」と現在お使いの PC の内容が異なるので確定処理が実行できません。	APIProxyActivateR egisterFix APIProxyActivateR egisterExecute APIGetProxyDataFor Register
欠番	143		
AlreadyUsedProxyData	144	この代理認証データファイルは、過去に使用されているので登録に利用できません。	APIProxyActivateR egisterFix
FailToRegisterConfirm	145	登録確定に失敗しました。	APIProxyActivateR egisterFix
FailToRegisterPreparation	146	登録準備に失敗しました。	APIProxyActivateR egisterPrepare
UnableToWriteProxyData	147	弊社のデータベースには正常に解除されましたが、代理認証データの書き込みが失敗しました。 「認証解除/電話」でお客様の PC での認証解除を実行していただく必要がありますので弊社までご連絡ください。	APIProxyActivateR emoveExecute
ThePathHasNotBeenSet	148	パスが設定されていません。	APIProxyActivateR egisterPrepare APIProxyActivateR emovePrepare APIPreparationOfP roxyUpdateOfExpir ationDate
PathNotFound	149	設定されたパスが存在しません。	APIProxyActivateR egisterPrepare APIProxyActivateR emovePrepare APIPreparationOfP roxyUpdateOfExpir ationDate
EmptyAnyOfTheInputData	150	この PC は認証登録されていません。	APIProxyActivateR emovePrepare
FailedToRemoveFromRegi	151	お客様の PC での認証解除準備が失	APIProxyActivateR

story		敗しました。 再度、当処理を実行してください。	emovePrepare
FailedToDeauthenticationPreparation	152	解除準備が失敗しました。	APIProxyActivateRemovePrepare
NotRestoreException	153	何らかの原因で回復できませんでした。	APIRestoreCancelStatus APIRestoreRegisterStatus
PleasePerformTheDeauthorizeProcess	154	弊社のデータベースにもデータが存在しますので当処理ではなく、「認証解除」処理を行ってください。	APIRestoreCancelStatus
FailedToDeauthorizeRecovery	155	お客様の PC で認証解除の回復に失敗しました。 「認証解除/電話」でお客様の PC での認証解除を実行して いただく必要がありますので弊社までご連絡ください。	APIRestoreCancelStatus
ProductIDIsEmpty	156	プロダクト ID を入力してください。	APIRestoreRegisterStatus
SerialNoIsEmpty	157	シリアル No.を入力してください。	APIRestoreRegisterStatus
FailedToRestoreActivation	158	お客様の PC で認証登録の回復に失敗しました。 弊社のデータベースの登録を解除した上で、再度、認証登録を実行していただく必要がありますので弊社までご連絡ください。	APIRestoreRegisterStatus
InputDataNotFound	159	入力された「プロダクト ID」や「シリアル No.」、PC 情報などが弊社のデータベースに登録されていません。 弊社のデータベースの登録を解除した上で、再度、認証登録を実行していただく必要がありますので弊社までご連絡ください。	APIRestoreRegisterStatus APIActivateStatusOnlineVerify
欠番	160	旧レンタル機能関連	
欠番	161	旧レンタル機能関連	
InvalidUseOfMACAdressOrCPU	162	MAC アドレスか CPU 情報の使用が無効なためこの処理は実行できません。 弊社までご連絡ください。	APIRestoreRegisterStatus
FailedToUpdateTheExpirationDate	163	「有効期限」の更新に失敗しました。	APIUpdateOfExpirationDate APIDeterminationOfProxyUpdateOfExpirationDate
CanNotUseTheString	164	使用できない文字列が含まれています。	APIActivateRegisterInternet APIActivateRemoveInternet APIProxyActivateR

			egisterExecute APIProxyActivateRemoveExecute APIRestoreCancelStatus APIRestoreRegisterStatus APIUpdateOfExpirationDate APIGetProductIdSerialNoList APIGetFreeItem APIGetRegisteredInfoFromWeb APIGetProxyUpdateOfExpirationDate APIFloatingLicenseRegister APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicenseFinish APIActivateStatusOnlineVerify
PropertyValueNotFound	165	「必須設定プロパティ」に値が設定されていません。	APIActivateRegisterInternet APIGetRegisteredInfoFromRegistry APIActivateRegisterTelephone APIActivateRemoveInternet APIActivateRemoveTelephone APIActivateStatusCheck APIActivateStatusCheckOnline APIGenerationOfNewCertificationID APIGetProxyDataForRemove APIProxyActivateRegisterExecute APIProxyActivateRegisterFix APIProxyActivateRegisterPrepare

			APIProxyActivateRemoveExecute APIProxyActivateRemovePrepare APIRestoreCancelStatus APIRestoreRegisterStatus APIUpdateOfExpirationDate APIGetProductIdSerialNoList APIGetFreeItem APITrialStartDateRemove APIRunNR2AppDateRemove APIRunNR2AppDateRemove2 APIActivateStatusCheck2 APIGetRegisteredInfoFromRegistry2 APIActivateStatusCheckOnline2 APIGetProxyDataForRegister APIWriteProxyServerInfoToRegistry APIReadProxyServerInfoFromRegistry APIGetRegisteredInfoFromWeb APIPreparationOfProxyUpdateOfExpirationDate APIGetProxyDataForExpirationDate APIGetProxyUpdateOfExpirationDate APIDeterminationOfProxyUpdateOfExpirationDate APIFloatingLicenseRegister APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicenseFinish

			APIActivateStatusOnlineVerify
NotUpdateTheExpirationDate	166	新しい有効期限が同じか古い場合「更新しない」ように設定されているので更新されませんでした。	APIUpdateOfExpirationDate APIGetProxyUpdateOfExpirationDate APIDeterminationOfProxyUpdateOfExpirationDate
CanNotReadTheTrialStartDay	167	日付データの取得に失敗しました。（猶予）	APIActivateRegisterInternet APIActivateRegisterTelephone APIActivateRemoveInternet APIActivateRemoveTelephone APIGenerationOfNewCertificationID APIGetRegisteredInfoFromRegistry APIProxyActivateRegisterFix APIProxyActivateRegisterPrepare APIProxyActivateRemovePrepare APIRestoreCancelStatus APIRestoreRegisterStatus APIUpdateOfExpirationDate APIPreparationOfProxyUpdateOfExpirationDate APIDeterminationOfProxyUpdateOfExpirationDate APIFloatingLicenseRegister APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicenseFinish APIActivateStatusOnlineVerify
欠番	168	旧レンタル機能関連	

CanNotReadTheExpirationStartDate	169	日付データの取得に失敗しました。（有効期限）	APIActivateRegisterInternet APIActivateRegisterTelephone APIActivateRemoveInternet APIActivateRemoveTelephone APIGenerationOfNewCertificationID APIGetProxyDataForRegister APIGetRegisteredInfoFromRegistry APIProxyActivateRegisterFix APIProxyActivateRegisterPrepare APIProxyActivateRegisterExecute APIProxyActivateRemovePrepare APIRestoreCancelStatus APIRestoreRegisterStatus APIUpdateOfExpirationDate APIPreparationOfProxyUpdateOfExpirationDate APIGetProxyDataForExpirationDate APIGetProxyUpdateOfExpirationDate APIDeterminationOfProxyUpdateOfExpirationDate APIDeterminationOfProxyUpdateOfExpirationDate APIFloatingLicenseRegister APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicenseFinish APIActivateStatusOnlineVerify

CanNotReadProxyServerInfoFromRegistry	170	プロキシサーバー情報の読み込みに失敗しました。	APIReadProxyServerInfoFromRegistry
CanNotWriteProxyServerInfoToRegistry	171	プロキシサーバー情報の書き込みに失敗しました。	APIWriteProxyServerInfoToRegistry
ExternalLinkKeyNotFound	172	指定されたリンク用キーは弊社データベースに登録されていません。	APIGetProductIdSerialNoList
FailToGetRegisterInfo	173	認証情報の取得に失敗しました。	APIPreparationOfProxyUpdateOfExpirationDate
PCDateChanged	174	PC の日付が変更されました。	APIActivateRegisterInternet APIActivateRegisterTelephone APIActivateRemoveInternet APIActivateRemoveTelephone APIGenerationOfNewCertificationID APIGetRegisteredInfoFromRegistry APIProxyActivateRegisterFix APIProxyActivateRegisterPrepare APIProxyActivateRemovePrepare APIRestoreCancelStatus APIRestoreRegisterStatus APIUpdateOfExpirationDate APIPreparationOfProxyUpdateOfExpirationDate APIDeterminationOfProxyUpdateOfExpirationDate APIFloatingLicenseRegister APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicenseFinish
ProxyExpirationDateDataFileNotFound	175	代理有効期限取得データファイルが存在しません。	APIGetProxyDataForExpirationDate APIGetProxyUpdat

			eOfExpirationDate APIDetermination OfProxyUpdateOfE xpirationDate
NoDataInProxyExpirationD ateDataFile	176	代理有効期限取得データファイルに データが存在しません。	APIGetProxyDataF orExpirationDate APIGetProxyUpdat eOfExpirationDate APIDetermination OfProxyUpdateOfE xpirationDate
FailedReadProxyExpiration DateData	177	代理有効期限取得データの読み込み に失敗しました。	APIGetProxyDataF orExpirationDate APIGetProxyUpdat eOfExpirationDate APIDetermination OfProxyUpdateOfE xpirationDate
FailedToWriteProxyExpirati onDateData	178	代理有効期限取得データの書き込み が失敗しました。 もう一度、「代理有効期限取得準備 (認証 PC)」から行ってください。	APIGetProxyUpdat eOfExpirationDate
NoExpirationDateSettingF orProxyExpirationDateDat aFile	179	代理有効期限取得データは有効期限 の利用設定はありません。	APIDetermination OfProxyUpdateOfE xpirationDate
HardwareInfoAndProxyExp irationDateDataIsMismatch	180	「代理有効期限取得データ」と現在お 使いの PC の内容が一致しません。	APIDetermination OfProxyUpdateOfE xpirationDate
FailedRemoveValueFromR egistory	181	削除処理が失敗しました。(レジスト リ)	APITrialStartDate Remove APIRunNR2AppDat eRemove APIRunNR2AppDat eRemove2
FailedRemoveValueFromF older	182	削除処理が失敗しました。(フォルダ)	APIRunNR2AppDat eRemove2
FailedReadRunAppDateVal ueFromRegistory	183	「アプリ起動日」の読み込みに失敗し ました。(レジストリ)	APIActivateRegist erInternet APIActivateRegist erTelephone APIActivateRemov eInternet APIActivateRemov eTelephone APIActivateStatus Check APIActivateStatus Check2 APIGenerationOfN ewCertificationID

			APIGetRegisteredInfoFromRegistry APIProxyActivateRegisterFix APIProxyActivateRegisterPrepare APIProxyActivateRemovePrepare APIRestoreCancelStatus APIRestoreRegisterStatus APIUpdateOfExpirationDate APIPreparationOfProxyUpdateOfExpirationDate APIDeterminationOfProxyUpdateOfExpirationDate APIActivateStatusCheckOnline APIActivateStatusCheckOnline2 APIGetRegisteredInfoFromRegistry2 APIFloatingLicenseRegister APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicenseFinish APIActivateStatusOnlineVerify
FailedWriteRunAppDateValueFromRegistory	184	「アプリ起動日」の書き込みに失敗しました。(レジストリ)	APIActivateRegisterInternet APIActivateRegisterTelephone APIActivateRemoveInternet APIActivateRemoveTelephone APIActivateStatusCheck APIActivateStatusCheck2 APIGenerationOfNewCertificationID

			APIGetRegisteredInfoFromRegistry APIProxyActivateRegisterFix APIProxyActivateRegisterPrepare APIProxyActivateRemovePrepare APIRestoreCancelStatus APIRestoreRegisterStatus APIUpdateOfExpirationDate APIPreparationOfProxyUpdateOfExpirationDate APIDeterminationOfProxyUpdateOfExpirationDate APIActivateStatusCheckOnline APIActivateStatusCheckOnline2 APIGetRegisteredInfoFromRegistry2 APIFloatingLicenseRegister APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicenseFinish APIActivateStatusOnlineVerify
FailedReadRunAppDateValueFromFolder	185	「アプリ起動日」の読み込みに失敗しました。(フォルダ)	APIActivateRegisterInternet APIActivateRegisterTelephone APIActivateRemoveInternet APIActivateRemoveTelephone APIActivateStatusCheck APIActivateStatusCheck2 APIGenerationOfNewCertificationID

			APIGetRegisteredInfoFromRegistry APIProxyActivateRegisterFix APIProxyActivateRegisterPrepare APIProxyActivateRemovePrepare APIRestoreCancelStatus APIRestoreRegisterStatus APIUpdateOfExpirationDate APIPreparationOfProxyUpdateOfExpirationDate APIDeterminationOfProxyUpdateOfExpirationDate APIActivateStatusCheckOnline APIActivateStatusCheckOnline2 APIGetRegisteredInfoFromRegistry2 APIFloatingLicenseRegister APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicenseFinish APIActivateStatusOnlineVerify
FailedWriteRunAppDateValueFromFolder	186	「アプリ起動日」の書き込みに失敗しました。(フォルダ)	APIActivateStatusCheck2
ProductIDAndSerialNoIsFloatingLicense	187	そのプロダクト ID + シリアル No.は、フローティングライセンスのため、この処理は無効です。	APIActivateRegisterInternet APIProxyActivateRegisterExecute APIRestoreRegisterStatus
ReturnValueNotFound	188	メソッドの値が見つかりません。	APIActivateRegisterInternet APIActivateRegisterTelephone APIProxyActivateRegisterExecute

			APIProxyActivateRemoveExecute APIRestoreCancelStatus APIRestoreRegisterStatus APIUpdateOfExpirationDate APIGetProductIdSerialNoList APIGetFreeItem APIGetRegisteredInfoFromWeb APIGetProxyUpdateOfExpirationDate APIFloatingLicenseRegister APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicenseFinish APIActivateStatusCheckOnline APIActivateStatusCheckOnline2 APIActivateStatusOnlineVerify
CanNotUseWithTrial	189	猶予(試用)でご利用の場合は、この処理はできません。	APIFloatingLicenseRegister APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicenseFinish
FailedReadRegistorydata	190	レジストリデータの読み込みで失敗しました。	APIActivateRemoveInternet APIProxyActivateRemovePrepare APIUpdateOfExpirationDate APIPreparationOfProxyUpdateOfExpirationDate APIDeterminationOfProxyUpdateOfExpirationDate APIFloatingLicense

			CancelAPIFloatingLicenseStart APIFloatingLicenseFinish
AlreadyFloatingLicenseRegistered	191	フローティングライセンス登録済のため、この処理は無効です。	APIActivateRegisterInternet APIActivateRegisterTelephone APIActivateRemoveInternet APIActivateRemoveTelephone APIGenerationOfNewCertificationID APIProxyActivateRegisterFix APIProxyActivateRegisterPrepare APIProxyActivateRemovePrepare APIRestoreCancelStatus APIRestoreRegisterStatus APIFloatingLicenseRegister
FloatingLicenseRegistrationFailed	192	フローティングライセンス登録が失敗しました。	APIFloatingLicenseRegister
CanNotUseFloatingLicense	193	指定されたプロダクト ID とシリアル№.は、フローティングライセンスは使用できません。	APIFloatingLicenseRegister
FloatingLicenseUnauthenticatedByTrialPeriodOut	194	フローティングライセンス登録されていないため、この処理は無効です。	APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicenseFinish
FloatingLicenseUnauthenticatedByTrialPeriod	195	フローティングライセンス登録されていないため、この処理は無効です。	APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicenseFinish
FloatingLicenseUnauthenticated	196	フローティングライセンス登録されていないため、この処理は無効です。	APIFloatingLicenseCancel APIFloatingLicenseStart APIFloatingLicenseFinish

FloatingLicenseInUse	197	フローティングライセンス終了処理を行ってから再度、当処理を実行してください。	APIFloatingLicenseCancel
FloatingLicenseCancellationFailed	198	フローティングライセンス解除が失敗しました。	APIFloatingLicenseCancel
ExpirationDateHasExpired	199	有効期限が切れているため、この処理は無効です。	APIFloatingLicenseStart
AlreadyFloatingLicenseStarted	200	既にフローティングライセンスが開始されています。	APIFloatingLicenseStart
LicenseKeyIsDifferent	201	フローティングライセンスで登録した「ライセンスキー」がデータベースと違います。フローティングライセンス解除を行い、もう一度フローティングライセンス登録を行ってからこの処理を行ってください。	APIFloatingLicenseStart
ExpirationDateIsDifferent	202	フローティングライセンスで登録した「有効期限」がデータベースと違います。フローティングライセンス解除を行い、もう一度フローティングライセンス登録を行ってからこの処理を行ってください。	APIFloatingLicenseStart
FailedToCancelFloatingLicense	203	何らかの原因でフローティングライセンスを解除できませんでした。	APIFloatingLicenseCancel
FailedToStartFloatingLicense	204	何らかの原因でフローティングライセンスを開始できませんでした。	APIFloatingLicenseStart
FloatingLicenseNotStarted	205	フローティングライセンスが開始されていないため、この処理は無効です。	APIFloatingLicenseFinish
RegisteredProductIDAndSerialNoIsNotFound	206	レジストリに登録されているプロダクトID+シリアル No.に該当する認証キーデータが存在しません。	APIActivateStatusOnlineVerify
FailedToFinishFloatingLicense	207	何らかの原因でフローティングライセンスを終了できませんでした。	APIFloatingLicenseFinish
FailedToRegisterFloatingLicense	208	何らかの原因でフローティングライセンスを登録できませんでした。	APIFloatingLicenseRegister
ErrorCallingWebServiceMethod	209	Web サービスのメソッド呼び出しでエラーが発生しました。	APIActivateRegisterInternet APIActivateRemoveInternet APIProxyActivateRegisterExecute APIProxyActivateRemoveExecute APIRestoreCancelStatus APIRestoreRegisterStatus APIUpdateOfExpirationDate APIGetProductIdSerialNoList

			APIGetFreeItem APIGetRegisteredInfoFromWeb APIGetProxyUpdateOfExpirationDate APIFloatingLicenseRegister APIFloatingLicenseStart APIFloatingLicenseFinish APIFloatingLicenseCancel APIActivateStatusOnlineVerify APIActivateStatusCheckOnline APIActivateStatusCheckOnline2
ErrorInWebServiceMethod	210	Web サービスのメソッド内でエラーが発生しました。	APIActivateStatusOnlineVerify
ProxyServerConnectionFailed	211	プロキシサーバーの接続に失敗しました。	APIActivateRegisterInternet APIActivateRemoveInternet APIProxyActivateRegisterExecute APIProxyActivateRemoveExecute APIRestoreCancelStatus APIRestoreRegisterStatus APIUpdateOfExpirationDate APIGetProductIdSerialNoList APIGetFreeItem APIGetRegisteredInfoFromWeb APIGetProxyUpdateOfExpirationDate APIFloatingLicenseRegister APIFloatingLicenseStart APIFloatingLicenseFinish APIFloatingLicenseCancel

			APIActivateStatusCheckOnline APIActivateStatusCheckOnline2 APIActivateStatusOnlineVerify
ServerConnectionFailed	212	サーバーの接続に失敗しました。	APIActivateRegisterInternet APIActivateRemoveInternet APIProxyActivateRegisterExecute APIProxyActivateRemoveExecute APIRestoreCancelStatus APIRestoreRegisterStatus APIUpdateOfExpirationDate APIGetProductIdSerialNoList APIGetFreeItem APIGetRegisteredInfoFromWeb APIGetProxyUpdateOfExpirationDate APIFloatingLicenseRegister APIFloatingLicenseStart APIFloatingLicenseFinish APIFloatingLicenseCancel APIActivateStatusCheckOnline APIActivateStatusCheckOnline2 APIActivateStatusOnlineVerify
	9999	その他エラー	

APICertificationID プロパティ

【機能】
認証 ID を取得します。(取得専用)

【構文】
<VB.NET>

```
Public Property APICertificationID As String
```

<C#>

```
public string APICertificationID { get; }
```

<VC++>

```
public : _bstr_t NewtoneNRDvcpp::INRD_APIActivation::APICertificationID
```

【解説】
認証 ID を取得します。(取得専用)
次のメソッドを実行すると当プロパティ(APICertificationID プロパティ)に()内の各値が設定されます。
・APIGenerationOfNewCertificationID メソッド(新しく生成された認証 ID)
・APIGetRegisteredInfoFromRegistry メソッド(認証登録済みの認証 ID)
・APIGetProxyDataForRegister メソッド(代理認証登録データから取得した認証 ID)
・APIGetProxyDataForRemove メソッド(代理認証解除データから取得した認証 ID)

APICurrentExpirationDate プロパティ

【機能】
現在の有効期限を取得します。(取得専用)

【構文】
<VB.NET>
Public Property **APICurrentExpirationDate** As **String**

<C#>
public **string** **APICurrentExpirationDate** { get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_APIActivation::APICurrentExpirationDate**

【解説】
現在の有効期限を取得します。(取得専用)
次のメソッドの成功時のみ、当プロパティに値が設定されます。
・APIGetRegisteredInfoFromRegistry メソッド(レジストリからの証登録済み情報の取得)

当プロパティの値を利用するメソッドは次の通りです。
・APIUpdateOfExpirationDate メソッド(「有効期限の更新」を実行)

APIDisabledNICIgnore プロパティ

【機能】
処理を高速化するため、無効になっている NIC（Network Interface Card）を無視します。
デフォルト：True。

【構文】
<VB.NET>
Public Property **APIDisabledNICIgnore** As **Boolean**

<C#>
public **bool** **APIDisabledNICIgnore** { set; get; }

<VC++>
public : **VARIANT_BOOL**
NewtoneNRDvcpp::INRD_APIActivation::APIDisabledNICIgnore

【解説】
認証レスキュー！の認証 UI ライブラリでは、認証登録など各処理時にエンドユーザ PC の MAC アドレスを最大で 5 個記録します。MAC アドレスは LAN ポートなどのネットワーク機器に固有な物理アドレスです。
このプロパティが False の場合、無効になっている NIC を一度有効にし、情報を取得後に再度 NIC を無効にします。この際に少し時間が掛かります。
このプロパティを True に設定すると、この無効の NIC は無視し、有効の NIC のみ情報を取得して記録します。これにより、情報の取得が高速化します。
認証レスキュー！の各処理では、最初に認証状況を確認しており、認証済みの場合、記録されている情報と、使用されている PC 情報が合致しているか確認しています。
このプロパティを True に設定することで、各処理が高速化します。

＜ご注意＞
たとえば、エンドユーザ PC の NIC が 2 つ有効で 3 つが無効の状態とします。

MACアドレスの例：
①A11111111111 → NIC が有効
②B22222222222 → NIC が有効
③C33333333333 → NIC が無効
④D44444444444 → NIC が無効
⑤E55555555555 → NIC が無効

このプロパティが False の場合、5 個の MAC アドレスを取得するために、2 つ有効（①②）になっている MAC アドレスと、3 つの無効（③④⑤）の NIC を一度有効にし、MAC アドレスを取得後に再度無効に設定します。
この無効の NIC を有効にし、再度無効にすることで、少々時間が掛かります。

このプロパティが True の場合、この 3 つの無効の NIC は無視し、2 つの有効な MAC アドレスのみ取得します。これにより、MACアドレスの取得時間が短縮されます。
しかし、次の点にご注意ください。
たとえば、このプロパティが True で、2 つの有効な（①②）になっている MAC アドレスで認証登録をします。

次に、2 つの有効な（①②）NIC を無効にし、3 つの無効な（③④⑤）を有効にして認証解除は、MAC アドレスが一致しないため「-999：認証済みハードウエア情報不一致」となり実行できません。

APIEncryptionPassword プロパティ

【機能】
暗号化時のパスワードを設定します。

【構文】
<VB.NET>
Public Property **APIEncryptionPassword** As **String**

<C#>
public **string** **APIEncryptionPassword** { set; get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_APIActivation::APIEncryptionPassword**

【解説】
認証 UI ライブラリ(DLL)はエンドユーザ PC のレジストリやファイルにデータを出力する際に必要に応じてデータを暗号化しています。その際の暗号化の方法は貴社では指定できませんが、暗号化する時の 2 つのパラメータ、パスワードと Salt 文字列は貴社で指定できます。

当プロパティには暗号化時のパスワードを指定します。全角でも指定できます。
(例)"認証レスキュー！"
当プロパティの文字数は、空文字列は不可で 1～65535 文字ですが、8 文字から 15 文字程度が妥当と思われます。

APIEncryptionSaltString プロパティ

【機能】
暗号化時の Salt 文字列(8 文字以上)を設定します。

【構文】
<VB.NET>
Public Property **APIEncryptionSaltString** As **String**

<C#>
public **string** **APIEncryptionSaltString** { set; get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_APIActivation::APIEncryptionSaltString**

【解説】
認証 UI ライブラリ(DLL)はエンドユーザ PC のレジストリやファイルにデータを出力する際に必要に応じてデータを暗号化しています。その際の暗号化の方法は貴社では指定できませんが、暗号化する時の 2 つのパラメータ、パスワードと Salt 文字列は貴社で指定できます。

当プロパティには暗号化時の Salt 文字列を指定します。
必ず 8 文字以上で指定します。
(例)"12345678ABCDEFGH"

APIErrorStatus プロパティ

【機能】
カスタム UI 系のメソッドを使用した際のエラーの内容を返します。

【構文】
<VB.NET>
Public ReadOnly Property **APIErrorStatus** As **Integer**

<C#>
public **int** **APIErrorStatus** { get; }

<VC++>
public : **long NewtoneNRDvcpp::INRD_APIActivation::APIErrorStatus**

【解説】
カスタム UI 系のメソッドは戻り値として正常（True）かエラー（False）を返しますが、この APIErrorStatus プロパティはそのエラーの内容を APIError 列挙体の参照で返します。この APIErrorStatus プロパティは取得専用です。

APIExternalLinkKey プロパティ

【機能】
APIGetProductIdSerialNoList メソッド 実行時の外部データベースとのリンク用キー項目を設定します。

【構文】
<VB.NET>
Public Property **APIExternalLinkKey** As **String**

<C#>
public **string** **APIExternalLinkKey** { set; get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_APIActivation::APIExternalLinkKey**

【解説】
認証レスキュー！の ActivationKey テーブルよりプロダクト ID とシリアル No.のペア文字列の列挙を取得する APIGetProductIdSerialNoList メソッドの実行時に、外部データベースとのリンク用キー項目を当プロパティ（APIExternalLinkKey プロパティ）設定します。

認証レスキュー！とは別の外部データベースとのリンク用キーを当プロパティ（APIExternalLinkKey プロパティ）に設定し、APIGetProductIdSerialNoList メソッドを実行すると、認証レスキュー！のデータベースの ActivationKey テーブルより該当するプロダクト ID とシリアル No.のペア文字列の列挙をデリミタ付きで APIProductIdSerialNoList プロパティに設定して返します。

APINewExpirationDate プロパティ

【機能】
新しい有効期限を取得します。(取得専用)

【構文】
<VB.NET>
Public Property **APINewExpirationDate** As **String**

<C#>
public **string** **APINewExpirationDate** { get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_APIActivation::APINewExpirationDate**

【解説】
新しい有効期限を取得します。(取得専用)
次のメソッドの成功時のみ、当プロパティに値が設定されます。
・APIGetProxyDataForExpirationDate メソッド(代理有効期限取得データの取得)
・APIGetProxyUpdateOfExpirationDate メソッド(「代理有効期限/取得」を実行(代理 PC で利用))

当プロパティの値を利用するメソッドは次の通りです。
・APIDeterminationOfProxyUpdateOfExpirationDate メソッド(「代理有効期限更新/確定」を実行)

APIFloatingLicenseDataInfo プロパティ

【機能】
フローティングライセンス使用中の PC 一覧を文字列として取得します。(取得専用)

【構文】
<VB.NET>
Public Property **APIFloatingLicenseDataInfo** As **String**

<C#>
public **string** **APIFloatingLicenseDataInfo** { get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_APIActivation::APIFloatingLicenseDataInfo**

【解説】
フローティングライセンス登録されている「プロダクト ID」と「シリアル No.」に該当する、使用中の PC 一覧を文字列として取得します。(取得専用)
文字列は、「PC 名」 デリミタのカンマ(,) 「認証 ID」として設定されます。
(例)"PCName1,50533-21818,PCName2,17958-26503,PCName3,78359-54685"
次のメソッドの成功時のみ、当プロパティに値が設定されます。
・APIActivateStatusOnlineVerify メソッド

APIFloatingLicenseMaxCount プロパティ

【機能】
フローティングライセンスの最大ライセンス数を取得します。(取得専用)

【構文】
<VB.NET>
Public ReadOnly Property **APIFloatingLicenseMaxCount** As **Integer**

<C#>
public **int** **APIFloatingLicenseMaxCount** { get; }

<VC++>
public : **long NewtoneNRDvcpp::INRD_APIActivation::APIFloatingLicenseMaxCount**

【解説】
フローティングライセンスの最大ライセンス数を取得します。(取得専用)
最大ライセンス数とは、レジストリからのフローティングライセンス登録済情報に該当する認証キーテーブルのプロダクト ID とシリアル No.に設定されている「ライセンス数」と「プラス許可数」を足した数です。
次のメソッドの成功時のみ、当プロパティに値が設定されます。
・APIActivateStatusOnlineVerify メソッド

APIFloatingLicenseState プロパティ

【機能】
フローティングライセンス状況(0:未登録　1:登録済　2:使用中)を取得します。(取得専用)

【構文】
<VB.NET>
Public ReadOnly Property **APIFloatingLicenseState** As **Byte**

<C#>
public **byte** **APIFloatingLicenseState** { get; }

<VC++>
public : **unsigned char**

NewtoneNRDvcpp::INRD_APIActivation::APIFloatingLicenseState

【解説】
フローティングライセンス状況(0:未登録　1:登録済　2:使用中)を取得します。(取得専用)
次のメソッドの成功時のみ、当プロパティに値が設定されます。
・APIActivateStatusOnlineVerify メソッド(0:未登録　1:登録済　2:使用中)
・APIGetRegisteredInfoFromRegistry メソッド(0:未登録　1:登録済)

APIActivateStatusOnlineVerify メソッドは、オンラインで貴社のデータベースに接続しフローティングライセンスが使用中かどうか確認しているのに対し、APIGetRegisteredInfoFromRegistry メソッドは、レジストリ情報のみの確認ですので「2:使用中」は返しません。

APIFloatingLicenseUsedCount プロパティ

【機能】
フローティングライセンスのライセンス使用数を取得します。(取得専用)

【構文】
<VB.NET>
Public ReadOnly Property **APIFloatingLicenseUsedCount** As **Integer**

<C#>
public **int** **APIFloatingLicenseUsedCount** { get; }

<VC++>
public : **long NewtoneNRDvcpp::INRD_APIActivation::APIFloatingLicenseUsedCount**

【解説】
フローティングライセンスの使用数を取得します。(取得専用)
使用数とは、フローティングライセンスの最大ライセンス数(APIFloatingLicenseMaxCount プロパティ)のうち、いくつ使用されているかの数です。
次のメソッドの成功時のみ、当プロパティに値が設定されます。
・APIActivateStatusOnlineVerify メソッド

APIFreeItem1～5 プロパティ

【機能】
APIGetFreeItem メソッドを実行して ActivationKey テーブルより取得した自由入力項目の値が当プロパティ(APIFreeItem1～APIFreeItem5 プロパティ)に設定されます。(取得専用)

【構文】
<VB.NET>

```
Public Property APIFreeItem1 As String
Public Property APIFreeItem2 As String
Public Property APIFreeItem3 As String
Public Property APIFreeItem4 As String
Public Property APIFreeItem5 As String
```

<C#>

```
public string APIFreeItem1 { get; }
public string APIFreeItem2 { get; }
public string APIFreeItem3 { get; }
public string APIFreeItem4 { get; }
public string APIFreeItem5 { get; }
```

<VC++>

```
public : _bstr_t NewtoneNRDvcpp::INRD_APIActivation::APIFreeItem1
public : _bstr_t NewtoneNRDvcpp::INRD_APIActivation::APIFreeItem2
public : _bstr_t NewtoneNRDvcpp::INRD_APIActivation::APIFreeItem3
public : _bstr_t NewtoneNRDvcpp::INRD_APIActivation::APIFreeItem4
public : _bstr_t NewtoneNRDvcpp::INRD_APIActivation::APIFreeItem5
```

【解説】
APIGetFreeItem メソッドを実行して ActivationKey テーブルより取得した自由入力項目の値が当プロパティ(APIFreeItem1～APIFreeItem5 プロパティ)に設定されます。

APILicenseKey プロパティ

【機能】
ライセンスキーを設定・取得します。

【構文】
<VB.NET>
Public Property **APILicenseKey** As **String**

<C#>
public **string** **APILicenseKey** { set; get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_APIActivation::APILicenseKey**

【解説】
ライセンスキーを設定・取得します。
次のメソッドを実行すると当プロパティ(APILicenseKey プロパティ)に()内の値が設定されます。
・APIGetRegisteredInfoFromRegistry メソッド(認証登録済みのライセンスキー)

APIOverwriteModeOfExpirationDateUpdate プロパティ

【機能】
「有効期限の更新」(APIUpdateOfExpirationDate)メソッドを実行する際に、データベース上の新しい有効期限が現在のエンドユーザ PC のレジストリ内有効期限が同じか古い場合の対応を設定します。

【構文】
<VB.NET>
Public Property **APIOverwriteModeOfExpirationDateUpdate** As **Boolean**

<C#>
public **bool** **APIOverwriteModeOfExpirationDateUpdate** { set; get; }

<VC++>
public : **VARIANT_BOOL**

NewtoneNRDvcpp::INRD_APIActivation::APIOverwriteModeOfExpirationDateUpdate

【解説】
当プロパティには次のいづれか値を設定します。
True:新しい有効期限が現在と同じか古い場合でも更新します。
False:新しい有効期限が現在と同じか古い場合は更新しません。

「有効期限の更新」の実行については、APIUpdateOfExpirationDate メソッドをご覧ください。

APIProductID プロパティ

【機能】
プロダクト ID を設定・取得します。

【構文】
<VB.NET>
Public Property **APIProductID** As **String**

<C#>
public **string** **APIProductID** { set; get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_APIActivation::APIProductID**

【解説】
プロダクト ID を設定・取得します。
次のメソッドを実行すると当プロパティ（APIProductID プロパティ）に（）内の各値が設定されます。
・APIGetRegisteredInfoFromRegistry メソッド（認証登録済みのプロダクト ID）
・APIGetProxyDataForRegister メソッド（代理認証登録データから取得したプロダクト ID）
・APIGetProxyDataForRemove メソッド（代理認証解除データから取得したプロダクト ID）

APIProductIdSerialNoList プロパティ

【機能】
APIGetProductIdSerialNoList メソッド実行後にプロダクト ID とシリアル No.のペアをデリミタで列挙したの文字列が設定されます。(取得専用)

【構文】
<VB.NET>
Public Property **APIProductIdSerialNoList** As **String**

<C#>
public **string** **APIProductIdSerialNoList** { get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_APIActivation::APIProductIdSerialNoList**

【解説】
APIGetProductIdSerialNoList メソッド実行後にプロダクト ID とシリアル No.のペアをデリミタで列挙した文字列が当プロパティ(APIProductIdSerialNoList プロパティ)設定されます。

認証レスキュー！とは別の外部データベースとのリンク用キーを APIExternalLinkKey プロパティに設定し、APIGetProductIdSerialNoList メソッドを実行すると、認証レスキュー！のデータベースの ActivationKey テーブルより該当するプロダクト ID とシリアル No.のペア文字列の列挙をデリミタ付きで当プロパティ(APIProductIdSerialNoList プロパティ)に設定して返します。

プロダクト ID とシリアル No.のペア文字列の列挙は、たとえば次のような文字列です。
デリミタはカンマ(,)です。
1111-1111-1111,aaaa1111,22222-22222-22222,bbbb2222,33333-33333-33333,cccc3333

この例では次の 3 組のプロダクト ID とシリアル No.が返されました。
プロダクト ID =”11111-11111-11111” シリアル No.=”aaaa1111”
プロダクト ID =”22222-22222-22222” シリアル No.=”bbbb2222”
プロダクト ID =”33333-33333-33333” シリアル No.=”cccc3333”

APIProxyDataPath プロパティ

【機能】
代理認証データパスを設定・取得します。
（代理認証機能については、「代理認証機能について」を参照）

【構文】
<VB.NET>
Public Property **APIProxyDataPath** As **String**

<C#>
public **string** **APIProxyDataPath** { set; get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_APIActivation::APIProxyDataPath**

【解説】
代理認証データパスを設定・取得します。ファイルの拡張子は「.NRS」とします。
当プロパティを利用するメソッドは次の通りです。
・APIGetProxyDataForRegister メソッド（代理認証登録データの取得）
・APIGetProxyDataForRemove メソッド（代理認証解除データの取得）
・APIProxyActivateRegisterExecute メソッド（代理認証登録/実行）
・APIProxyActivateRegisterFix メソッド（代理認証登録/確定）
・APIProxyActivateRegisterPrepare メソッド（代理認証登録/準備）
・APIProxyActivateRemoveExecute メソッド（代理認証解除/実行）
・APIProxyActivateRemovePrepare メソッド（代理認証解除/準備）

APIProxyServerAddress プロパティ

【機能】
プロキシサーバーのアドレスを設定します。

【構文】
<VB.NET>
Public Property **APIProxyServerAddress** As **String**

<C#>
public **string** **APIProxyServerAddress** { set; get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_APIActivation::APIProxyServerAddress**

【解説】
プロキシサーバーのアドレスを設定します。
インターネットに接続する際にプロキシサーバーを使用する場合のみ設定が必要です。
プロキシサーバーを使用する場合は、当プロパティを含む次のプロパティに適切な設定が必要です。
・APIProxyServerAddress プロパティ(プロキシサーバーのアドレス)→**当プロパティ**
・APIProxyServerPort プロパティ(プロキシサーバーのポート)
・APIProxyServerUserName プロパティ(プロキシサーバーのユーザ名)
・APIProxyServerPassword プロパティ(プロキシサーバーのパスワード)

プロキシサーバーを使用する場合、当プロパティの値を利用するメソッドは次の通りです。
・APIActivateRegisterInternet メソッド(インターネットによる認証登録を実行)
・APIActivateRemoveInternet メソッド(インターネットによる認証解除を実行)
・APIProxyActivateRegisterExecute メソッド(代理認証登録/実行)
・APIProxyActivateRemoveExecute メソッド(代理認証解除/実行)
・APIRestoreRegisterStatus メソッド(認証登録状態回復)
・APIRestoreCancelStatus メソッド(認証解除状態回復)
・APIUpdateOfExpirationDate メソッド(有効期限の更新)

APIProxyServerPassword プロパティ

【機能】
プロキシサーバーのパスワードを設定します。

【構文】
<VB.NET>
Public Property **APIProxyServerPassword** As **String**

<C#>
public **string** **APIProxyServerPassword** { set; get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_APIActivation::APIProxyServerPassword**

【解説】
プロキシサーバーのパスワードを設定します。
インターネットに接続する際にプロキシサーバーを使用する場合のみ設定が必要です。
プロキシサーバーを使用する場合は、当プロパティを含む次のプロパティに適切な設定が必要です。
・APIProxyServerAddress プロパティ(プロキシサーバーのアドレス)
・APIProxyServerPort プロパティ(プロキシサーバーのポート)
・APIProxyServerUserName プロパティ(プロキシサーバーのユーザ名)
・APIProxyServerPassword プロパティ(プロキシサーバーのパスワード)→**当プロパティ**

プロキシサーバーを使用する場合、当プロパティの値を利用するメソッドは次の通りです。
・APIActivateRegisterInternet メソッド(インターネットによる認証登録を実行)
・APIActivateRemoveInternet メソッド(インターネットによる認証解除を実行)
・APIProxyActivateRegisterExecute メソッド(代理認証登録/実行)
・APIProxyActivateRemoveExecute メソッド(代理認証解除/実行)
・APIRestoreRegisterStatus メソッド(認証登録状態回復)
・APIRestoreCancelStatus メソッド(認証解除状態回復)
・APIUpdateOfExpirationDate メソッド(有効期限の更新)

APIProxyServerPort プロパティ

【機能】
プロキシサーバーのポートを設定します。

【構文】
<VB.NET>
Public Property **APIProxyServerPort** As **String**

<C#>
public **string** **APIProxyServerPort** { set; get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_APIActivation::APIProxyServerPort**

【解説】
プロキシサーバーのポートを設定します。
インターネットに接続する際にプロキシサーバーを使用する場合のみ設定が必要です。
プロキシサーバーを使用する場合は、当プロパティを含む次のプロパティに適切な設定が必要です。
・APIProxyServerAddress プロパティ(プロキシサーバーのアドレス)
・APIProxyServerPort プロパティ(プロキシサーバーのポート)→**当プロパティ**
・APIProxyServerUserName プロパティ(プロキシサーバーのユーザ名)
・APIProxyServerPassword プロパティ(プロキシサーバーのパスワード)

プロキシサーバーを使用する場合、当プロパティの値を利用するメソッドは次の通りです。
・APIActivateRegisterInternet メソッド(インターネットによる認証登録を実行)
・APIActivateRemoveInternet メソッド(インターネットによる認証解除を実行)
・APIProxyActivateRegisterExecute メソッド(代理認証登録/実行)
・APIProxyActivateRemoveExecute メソッド(代理認証解除/実行)
・APIRestoreRegisterStatus メソッド(認証登録状態回復)
・APIRestoreCancelStatus メソッド(認証解除状態回復)
・APIUpdateOfExpirationDate メソッド(有効期限の更新)

APIProxyServerUserName プロパティ

【機能】
プロキシサーバーのユーザ名を設定します。

【構文】
<VB.NET>
Public Property **APIProxyServerUserName** As **String**

<C#>
public **string** **APIProxyServerUserName** { set; get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_APIActivation::APIProxyServerUserName**

【解説】
プロキシサーバーのユーザ名を設定します。
インターネットに接続する際にプロキシサーバーを使用する場合のみ設定が必要です。
プロキシサーバーを使用する場合は、当プロパティを含む次のプロパティに適切な設定が必要です。
・APIProxyServerAddress プロパティ(プロキシサーバーのアドレス)
・APIProxyServerPort プロパティ(プロキシサーバーのポート)
・APIProxyServerUserName プロパティ(プロキシサーバーのユーザ名)→**当プロパティ**
・APIProxyServerPassword プロパティ(プロキシサーバーのパスワード)

プロキシサーバーを使用する場合、当プロパティの値を利用するメソッドは次の通りです。
・APIActivateRegisterInternet メソッド(インターネットによる認証登録を実行)
・APIActivateRemoveInternet メソッド(インターネットによる認証解除を実行)
・APIProxyActivateRegisterExecute メソッド(代理認証登録/実行)
・APIProxyActivateRemoveExecute メソッド(代理認証解除/実行)
・APIRestoreRegisterStatus メソッド(認証登録状態回復)
・APIRestoreCancelStatus メソッド(認証解除状態回復)
・APIUpdateOfExpirationDate メソッド(有効期限の更新)

APIReleaseKey プロパティ

【機能】
解除キーを設定・取得します。

【構文】
<VB.NET>
Public Property **APIReleaseKey** As **String**

<C#>
public **string** **APIReleaseKey** { set; get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_APIActivation::APIReleaseKey**

【解説】
解除キーを設定・取得します。
当プロパティは、電話による認証解除メソッド（APIActivateRemoveTelephone メソッド）で使用します。
解除キーは通常、たとえば"956-11749-60711"といったハイフン付きの数字列です。

APIReleaseStatus プロパティ

【機能】
解除ステータスを取得します。(取得専用)

【構文】
<VB.NET>
Public Property **APIReleaseStatus** As **String**

<C#>
public **string** **APIReleaseStatus** { get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_APIActivation::APIReleaseStatus**

【解説】
解除ステータスを取得します。(取得専用)
当プロパティは、電話による認証解除メソッド(APIActivateRemoveTelephone メソッド)の実行が成功した場合のみ自動的に設定されます。
解除ステータスは通常、たとえば"162-20797-49245"といったハイフン付きの数字列です。

APISelectRunAppDatePathFlag プロパティ

【機能】
「アプリ起動日」の読み込み/書き込み場所の切り替えフラグを設定します。
APIActivateStatusCheck2 メソッドと APIActivateStatusCheckOnline2 メソッドで使用されます。

【構文】
<VB.NET>
Public Property **APISelectRunAppDatePathFlag** As **Integer**

<C#>
public **int** **APISelectRunAppDatePathFlag** { set; get; }

<VC++>
public : **long NewtoneNRDvcpp::INRD_APIActivation::APISelectRunAppDatePathFlag**

【解説】
このプロパティ値により「アプリ起動日」の読み込み/書き込み場所を切り替えます。
0 を指定した場合：HKEY_CURRENT_USER（レジストリ）
1 を指定した場合：ProgramData（フォルダ）
APIActivateStatusCheck2 メソッドと APIActivateStatusCheckOnline2 メソッドで使用されます。

APISerialNo プロパティ

【機能】
シリアル No.を設定・取得します。

【構文】
<VB.NET>
Public Property **APISerialNo** As **String**

<C#>
public **string** **APISerialNo** { set; get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_APIActivation::APISerialNo**

【解説】
シリアル No.を設定・取得します。
次のメソッドを実行すると当プロパティ(APIProductID プロパティ)に()内の各値が設定されます。
・APIGetRegisteredInfoFromRegistry メソッド(認証登録済みのシリアル No.)
・APIGetProxyDataForRegister メソッド(代理認証登録データから取得したシリアル No.)
・APIGetProxyDataForRemove メソッド(代理認証解除データから取得したシリアル No.)

APITrialPeriod プロパティ

【機能】
猶予(試用)日数(デフォルト:0 日、設定可能範囲:1～365)を設定します。

【構文】
<VB.NET>
Public Property **APITrialPeriod** As **Integer**

<C#>
public **int** **APITrialPeriod** { set; get; }

<VC++>
public : **long NewtoneNRDvcpp::INRD_APIActivation::APITrialPeriod**

【解説】
エンドユーザがライセンス認証登録をしなくてもアプリケーションが動作する期間を指定します。
1(日)～365(日)の間で設定できます。
当プロパティを 0 に設定すると、猶予期間は無いことになり、ライセンス認証登録をしない限りアプリケーション(またはアプリケーションの主機能など)が動作しないようにできます。

APIUseCpuInfo プロパティ

【機能】
CPU 情報の使用(デフォルト:True)を設定します。
次の 3 つのメソッドの処理において、認証情報とエンドユーザ PC 内の CPU 情報とを照合するかどうかを設定します。
・APIActivateStatusCheck メソッド(エンドユーザ PC 内での認証状態の確認)
・APIActivateStatusCheckOnline メソッド(オンラインで認証状態の確認)
・APIRestoreRegisterStatus メソッド(「認証登録状態回復」処理の実行)

【構文】
<VB.NET>
Public Property **APIUseCpuInfo** As **Boolean**

<C#>
public **bool** **APIUseCpuInfo** { set; get; }

<VC++>
public : **VARIANT_BOOL NewtoneNRDvcpp::INRD_APIActivation::APIUseCpuInfo**

【解説】
認証レスキュー！の認証 UI ライブラリでは、認証登録など各処理時にエンドユーザ PC の CPU 情報を記録します。当プロパティを True にするとエンドユーザ PC の CPU 情報を認証識別情報として利用します。これを利用することでエンドユーザ PC の不正利用時の認証識別情報の精度を上げます。
※APIRestoreRegisterStatus メソッド(「認証登録状態回復」処理の実行)を使用する際は、当プロパティを必ず True に設定してください。

APIUseMacAddress プロパティ

【機能】
MAC アドレスの使用（デフォルト：True）を設定します。
次の 3 つのメソッドの処理において、認証情報とエンドユーザ PC 内の MAC アドレスとを照合するかどうかを設定します。
・APIActivateStatusCheck メソッド（エンドユーザ PC 内での認証状態の確認）
・APIActivateStatusCheckOnline メソッド（オンラインで認証状態の確認）
・APIRestoreRegisterStatus メソッド（「認証登録状態回復」処理の実行）

【構文】
<VB.NET>
Public Property **APIUseMacAddress** As **Boolean**

<C#>
public **bool** **APIUseMacAddress** { set; get; }

<VC++>
public : **VARIANT_BOOL NewtoneNRDvcpp::INRD_APIActivation::APIUseMacAddress**

【解説】
認証レスキュー！の認証 UI ライブラリでは、認証登録など各処理時にエンドユーザ PC の MAC アドレスを最大で 5 個記録します。MAC アドレスは LAN ポートなどのネットワーク機器に固有な物理アドレスです。MAC アドレスは、たとえば"E840F260C430"といった文字列です
当プロパティを True にするとエンドユーザ PC の MAC アドレスを認証識別情報として利用します。これを利用することでエンドユーザ PC の不正利用時の認証識別情報の精度を上げます。
※APIRestoreRegisterStatus メソッド（「認証登録状態回復」処理の実行）を使用する際は、当プロパティを必ず True に設定してください。

APIUseProxyServer プロパティ

【機能】
プロキシサーバーの使用区分（デフォルト：False）を設定します。

【構文】
<VB.NET>
Public Property **APIUseProxyServer** As **Boolean**

<C#>
public **bool** **APIUseProxyServer** { set; get; }

<VC++>
public : **VARIANT_BOOL NewtoneNRDvcpp::INRD_APIActivation::APIUseProxyServer**

【解説】
プロキシサーバーの使用区分（デフォルト：False）を設定します。
インターネットに接続する際にプロキシサーバーを使用するかどうかを設定します。
プロキシサーバーを使用する場合は当プロパティに True を、使用しない場合は False をそれぞれ設定します。
プロキシサーバーを使用する場合は、次のプロパティに適切な設定が必要です。
・APIProxyServerAddress プロパティ（プロキシサーバーのアドレス）
・APIProxyServerPort プロパティ（プロキシサーバーのポート）
・APIProxyServerUserName プロパティ（プロキシサーバーのユーザ名）
・APIProxyServerPassword プロパティ（プロキシサーバーのパスワード）

プロキシサーバーを使用する場合、当プロパティの値を利用するメソッドは次の通りです。
・APIActivateRegisterInternet メソッド（インターネットによる認証登録を実行）
・APIActivateRemoveInternet メソッド（インターネットによる認証解除を実行）
・APIProxyActivateRegisterExecute メソッド（代理認証登録/実行）
・APIProxyActivateRemoveExecute メソッド（代理認証解除/実行）
・APIRestoreRegisterStatus メソッド（認証登録状態回復）
・APIRestoreCancelStatus メソッド（認証解除状態回復）
・APIUpdateOfExpirationDate メソッド（有効期限の更新）

APIVendorsProductStartRegistryKeyPath プロパティ

【機能】
ベンダアプリケーション開始レジストリキーパスを設定します。

【構文】
<VB.NET>
Public Property **APIVendorsProductStartRegistryKeyPath** As **String**

<C#>
public **string** **APIVendorsProductStartRegistryKeyPath** { set; get; }

<VC++>
public : **_bstr_t**

NewtoneNRDvcpp::INRD_APIActivation::APIVendorsProductStartRegistryKeyPath

【解説】
エンドユーザに配布したアプリケーションが認証 UI ライブラリ（DLL）の機能を使う場合、その各種情報の記録先として使うエンドユーザ PC 上のレジストリの開始キーのパスを当プロパティに指定します。レジストリ内の「HKEY_LOCAL_MACHINE」以降を指定します。
（例）"Software¥Newtone¥NinshoRescue¥NRD¥SampleProject"

なお、物理的なレジストリの位置は動作する貴社のアプリケーションが32bitなのか64bitなのかと、動作するPCのOSが32bitなのか64bitなのかによって異なります。
たとえば、HKEY_LOCAL_MACHINE¥SOFTWARE¥ABCというレジストリパスは次のようになります。
32bitOS上で32bitアプリケーションが動作する場合、または64bitOS上で64bitアプリケーションが動作する場合は、
HKEY_LOCAL_MACHINE¥SOFTWARE¥ABC
そのままです。
しかし、64bitOS上で32bitアプリケーションが動作する場合は、
HKEY_LOCAL_MACHINE¥SOFTWARE**¥Wow6432Node**¥ABC
となります。

また、貴社の異なる複数のアプリケーションをエンドユーザの同一PC上で使用させるには、当プロパティにアプリケーションごとのそれぞれ異なるレジストリパスを設定してください。たとえば、次のように設定します。
アプリケーションA内での設定コード例：
APIVendorsProductStartRegistryKeyPath = "Software¥Company¥A"
アプリケーションB内での設定コード例：
APIVendorsProductStartRegistryKeyPath = "Software¥Company¥B"

APIWebServiceBasicAuthenticationPassword プロパティ

【機能】
Web サービス時の基本認証パスワードを設定します。

【構文】
<VB.NET>
Public Property **APIWebServiceBasicAuthenticationPassword** As **String**

<C#>
public **string** **APIWebServiceBasicAuthenticationPassword** { set; get; }

<VC++>
public : **_bstr_t**
NewtoneNRDvcpp::INRD_APIActivation::APIWebServiceBasicAuthenticationPassword

【解説】
Web サーバーで基本認証を使用する場合に、そのパスワードを指定します。
基本認証に関しては、APIWebServiceUseBasicAuthentication プロパティを参照してください。

APIWebServiceBasicAuthenticationUserName プロパティ

【機能】
Web サービス時の基本認証ユーザ名を設定します。

【構文】
<VB.NET>
Public Property **APIWebServiceBasicAuthenticationUserName** As **String**

<C#>
public **string** **APIWebServiceBasicAuthenticationUserName** { set; get; }

<VC++>
public : **_bstr_t**
NewtoneNRDvcpp::INRD_APIActivation::APIWebServiceBasicAuthenticationUserName

【解説】
Web サーバーで基本認証を使用する場合に、そのユーザ名を指定します。
基本認証に関しては、APIWebServiceUseBasicAuthentication プロパティを参照してください。

APIWebServiceCheckPassword プロパティ

【機能】
Web サービス確認パスワード(8 文字以上)を設定します。

【構文】
<VB.NET>
Public Property **APIWebServiceCheckPassword** As **String**

<C#>
public **string** **APIWebServiceCheckPassword** { set; get; }

<VC++>
public : **_bstr_t**

NewtoneNRDvcpp::INRD_APIActivation::APIWebServiceCheckPassword

【解説】
Web サービスを利用する場合の確認用のパスワードを設定します。ここでは、Web サーバー側設定アプリケーション「認証 Web サービス」の環境設定処理の Web サービスの確認パスワードと同じものを設定します。
確認パスワードは必須項目です、省略はできません。8 文字以上で半角の次の文字が使用できます。
・大文字の英字（A～Z）
・小文字の英字（a～z）
・数字（0～9）
・記号（+-*/!#$%&()=¥@<>?）

APIWebServiceTimeout プロパティ

【機能】
Web サービスのタイムアウト（デフォルト：60 秒）を設定します。

【構文】
<VB.NET>
Public Property **APIWebServiceTimeout** As **Integer**

<C#>
public **int APIWebServiceTimeout** { set; get; }

<VC++>
public : **long NewtoneNRDvcpp::INRD_APIActivation::APIWebServiceTimeout**

【解説】
Web サービスに接続して応答を待つ最大時間を秒単位で指定します。初期値は 60 秒です。

APIWebServiceURL プロパティ

【機能】
Web サービスの URL を設定します。

【構文】
<VB.NET>
Public Property **APIWebServiceURL** As **String**

<C#>
public **string** **APIWebServiceURL** { set; get; }

<VC++>
public : **_bstr_t NewtoneNRDvcpp::INRD_APIActivation::APIWebServiceURL**

【解説】
認証に関するシステムを Web サービスとして提供する Web サーバーの URL を指定します。
以下に例を示します。

自 PC のローカルホストの Web サーバー(IIS)にアクセスする例:
"http://localhost/NRDWebService/Service.asmx"
(注)この例は実際の運用ではありえません。貴社のアプリケーションで認証 UI ライブラリ(DLL)を使用した開発時に、テスト用に使用される URL です。

自社 Web サーバー(IIS)にアクセスさせる例:
"http://www.newtone.co.jp/NRDWebService/Service.asmx"

クラウドサービス Microsoft Azure の Web アプリ(App Service)に配置した Web サービスを利用する例:
"http://newtonecojp.azurewebsites.net/Service.asmx"

APIWebServiceUseBasicAuthentication プロパティ

【機能】
Web サービス時の基本認証の使用（デフォルト：False）を設定します。

【構文】
<VB.NET>
Public Property **APIWebServiceUseBasicAuthentication** As **Boolean**

<C#>
public **bool** **APIWebServiceUseBasicAuthentication** { set; get; }

<VC++>
public : **VARIANT_BOOL**

NewtoneNRDvcpp::INRD_APIActivation::APIWebServiceUseBasicAuthentication

【解説】
Web サーバーで基本認証を使用する場合は、当プロパティを True にします。
初期値は、False（基本認証を使用しない）です。

当プロパティは、Web サーバー（IIS）側で特定のアカウント（ユーザ名とパスワード）でアクセスできるフォルダにこの Web サービスが配置してある場合に使用できるセキュリティ設定です。

Web サーバーで基本認証を使用する一般的な手順は次の通りです。
1.サーバーPC 上でユーザを作成。
　この際のユーザ名とパスワードがそのまま基本認証に使われます。
2.基本認証フォルダのセキュリティ設定
　フォルダのプロパティを開き、セキュリティタブで上記 1 のユーザ名を　追加し、「読み取り」権限を付与します。
3.IIS でのセキュリティ設定
　IIS　で該当フォルダに対し「認証」の設定で「匿名認証」を無効にして　「基本認証」を有効にします。

なお、Web サーバーでの基本認証の詳細につきましてはマイクロソフト社の関連ドキュメントをご覧ください。

<カスタム UI 系メソッド>

メソッド一覧

メソッド	機能
APIActivateRegisterInternet	インターネットによる認証登録の実行
APIActivateRegisterTelephone	電話による認証登録の実行
APIActivateRemoveInternet	インターネットによる認証解除の実行
APIActivateRemoveTelephone	電話による認証解除の実行
APIActivateStatusCheck	エンドユーザ PC 内での認証状態の確認
APIActivateStatusCheck2	エンドユーザ PC 内での認証状態の確認（Windows の管理者でなくても実行可能）
APIActivateStatusCheckOnline	オンラインで認証状態の確認
APIActivateStatusCheckOnline2	オンラインで認証状態の確認（Windows の管理者でなくても実行可能）
APIActivateStatusOnlineVerify	オンラインで認証状態の確認
APIDeterminationOfProxyUpdateOfExpirationDate	「代理有効期限更新/確定」処理の実行
APIFloatingLicenseCancel	フローティングライセンス登録解除の実行
APIFloatingLicenseFinish	フローティングライセンス使用終了の実行
APIFloatingLicenseRegister	フローティングライセンス登録の実行
APIFloatingLicenseStart	フローティングライセンス使用開始の実行
APIGenerationOfNewCertificationID	新規認証 ID の生成
APIGetFreeItem	プロダクト ID とシリアル No.を指定して、ActivationKey テーブルより自由入力項目を取得
APIGetProductIdSerialNoList	外部データベースとのリンク用キー項目を指定し ActivationKey テーブルよりプロダクト ID とシリアル No.のペア文字列の列挙を取得
APIGetProxyDataForExpirationDate	代理有効期限取得データの取得
APIGetProxyDataForRegister	代理認証登録データの取得
APIGetProxyDataForRemove	代理認証解除データの取得
APIGetProxyUpdateOfExpirationDate	「代理有効期限/取得」処理の実行（代理 PC で利用）
APIGetRegisteredInfoFromRegistry	（レジストリからの）認証登録済み情報の取得
APIGetRegisteredInfoFromRegistry2	（レジストリからの）認証登録済み情報の取得（Windows の管理者でなくても実行可能）
APIGetRegisteredInfoFromWeb	インターネットを使用して ActivationKeyTable から「ライセンス数」「有効期限の利用」「有効期限」を、ActivationDataTable から「認証 ID」「認証日時（作成日）」を取得します。
APIPreparationOfProxyUpdateOfExpirationDate	「代理有効期限取得/準備」処理の実行
APIProxyActivateRegisterExecute	「代理認証登録/実行」処理の実行（代理 PC で利用）
APIProxyActivateRegisterFix	「代理認証登録/確定」処理の実行
APIProxyActivateRegisterPrepare	「代理認証登録/準備」処理の実行
APIProxyActivateRemoveExecute	「代理認証解除/実行」処理の実行（代理 PC で利用）
APIProxyActivateRemovePrepare	「代理認証解除/準備」処理の実行
APIReadProxyServerInfoFromRegistry	レジストリからプロキシサーバーの情報を取得
APIRestoreCancelStatus	「認証解除状態回復」処理の実行
APIRestoreRegisterStatus	「認証登録状態回復」処理の実行

APIRunNR2AppDateRemove	「アプリ起動日」を削除
APIRunNR2AppDateRemove2	「アプリ起動日」を削除
APITrialStartDateRemove	「猶予日数」の「開始日」を削除
APIUpdateOfExpirationDate	有効期限更新の実行
APIWriteProxyServerInfoToRegistry	レジストリへプロキシサーバーの情報を書き込み

APIActivateRegisterInternet メソッド

【機能】
インターネットによる認証登録を実行します。

【構文】
<VB.NET>
Public Function **APIActivateRegisterInternet**() As **Boolean**

<C#>
public **bool** **APIActivateRegisterInternet**()

<VC++>
public：**VARIANT_BOOL**

NewtoneNRDvcpp::INRD_APIActivation::APIActivateRegisterInternet()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

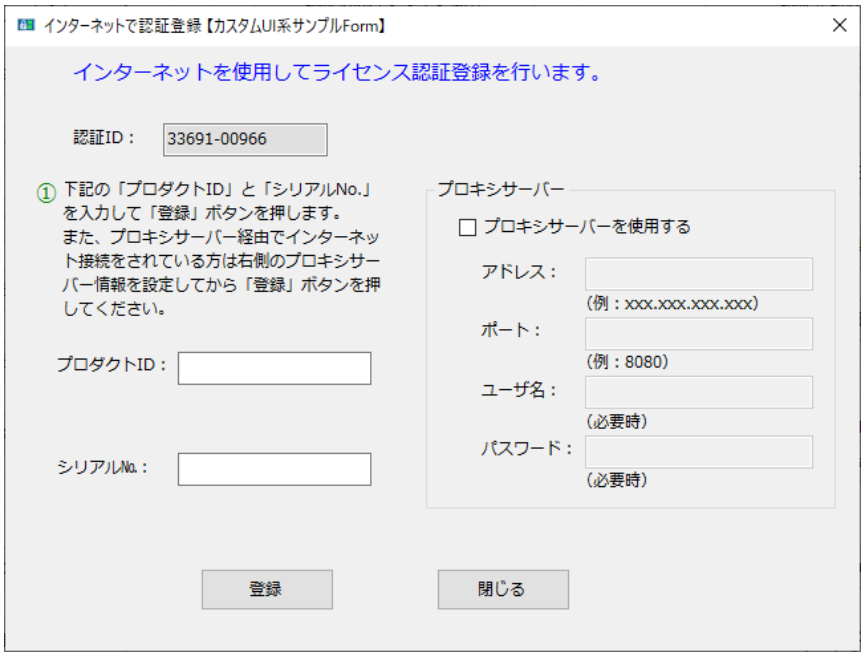
設定区分※	プロパティ	設定する内容	
UI	APICertificationID	（認証登録する）認証 ID（取得専用）。新規認証 ID 生成メソッド（APIGenerationOfNewCertificationID メソッド）の実行により、新しい認証 ID が当プロパティに設定されます。	
UI	APIProductID	（認証登録する）プロダクト ID	
UI	APISerialNo	（認証登録する）シリアル No.	
UI	APIUseProxyServer	プロキシサーバーの使用区分	
UI	APIProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時
UI	APIProxyServerPort	プロキシサーバーのポート	
UI	APIProxyServerUserName	プロキシサーバーのユーザ名	
UI	APIProxyServerPassword	プロキシサーバーのパスワード	
Code	APIDisabledNICIgnore	処理を高速化するため、無効になっている NIC（Network Interface Card）を無視します。	
Code	APIEncryptionPassword	暗号化時のパスワード	

Code	APIEncryptionSaltString	暗号化時の Salt 文字列
Code	APITrialPeriod	猶予(試用)日数(0:猶予期間なし)
Code	APIUseCpuInfo	CPU 情報の使用区分
Code	APIUseMacAddress	MAC アドレスの使用区分
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス
Code	APIWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード(基本認証使用時)
Code	APIWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名(基本認証使用時)
Code	APIWebServiceCheckPassword	Web サービス確認パスワード
Code	APIWebServiceTimeout	Web サービスのタイムアウト
Code	APIWebServiceURL	Web サービスの URL
Code	APIWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分

※設定区分:UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム(コード)で設定されるプロパティ

【解説】
インターネットによる認証登録を実行します。

◆想定 UI 画面
当メソッドの利用時に、想定する貴社作成の UI(ユーザインターフェース)画面は次の通りです。



この画面上の「認証 ID」は、新規認証 ID 生成メソッド(APIGenerationOfNewCertificationID メソッド)の実行で設定される認証 ID プロパティ(APICertificationID プロパティ)で取得できます。

◆処理手順
一連の処理の手順は次の通りです。

[1]新規認証 ID 生成メソッド(APIGenerationOfNewCertificationID メソッド)を実行します。
これにより、認証 ID プロパティ(APICertificationID プロパティ)に新しい認証 ID が設定されます。
その内容を画面に表示します。この内容表示そのものは必須ではありません。

[2]プロダクト ID とシリアル No.をエンドユーザに画面より入力させ、プロダクト ID プロパティ（APIProductID プロパティ）とシリアル No. プロパティ（APISerialNo プロパティ）にそれぞれ設定します。あるいは、別の方法で結果としてプロダクト ID プロパティ（APIProductID プロパティ）とシリアル No. プロパティ（APISerialNo プロパティ）にそれぞれ認証登録するプロダクト ID とシリアル No.を設定します。

[3]上記の【必須設定プロパティ】一覧にある「設定区分」が「Code」のプロパティに適切な値を設定します。

★これらのプロパティは、一度設定した内容で変更がなければ再設定する必要はありません。

[4]上記の「想定 UI 画面」の「登録」ボタンに連動させるなどして、当メソッド（APIActivateRegisterInternet メソッド）を実行します。

APIActivateRegisterTelephone メソッド

【機能】
電話による認証登録を実行します。

【構文】
<VB.NET>
Public Function **APIActivateRegisterTelephone**() As **Boolean**

<C#>
public **bool** **APIActivateRegisterTelephone**()

<VC++>
public：**VARIANT_BOOL**

NewtoneNRDvcpp::INRD_APIActivation::APIActivateRegisterTelephone()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

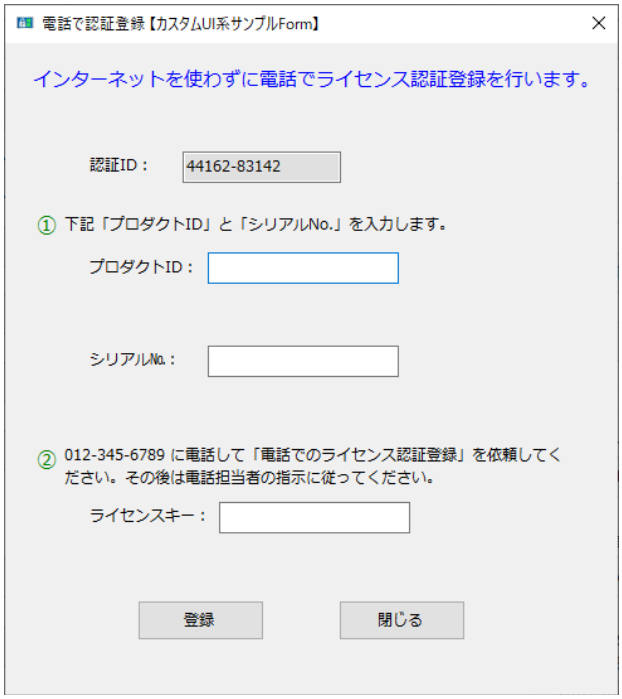
設定区分※	プロパティ	設定する内容
UI	APICertificationID	（認証登録する）認証 ID（取得専用）。 新規認証 ID 生成メソッド（APIGenerationOfNewCertificationID メソッド）の実行により、新しい認証 ID が当プロパティに設定されます。
UI	APIProductID	（認証登録する）プロダクト ID
UI	APISerialNo	（認証登録する）シリアル No.
UI	APILicenseKey	ライセンスキー （貴社電話担当者より告げられたライセンスキーを設定）
Code	APIDisabledNICIgnore	処理を高速化するため、無効になっている NIC（Network Interface Card）を無視します。
Code	APIEncryptionPassword	暗号化時のパスワード
Code	APIEncryptionSaltString	暗号化時の Salt 文字列
Code	APITrialPeriod	猶予（試用）日数（0：猶予期間なし）
Code	APIUseCpuInfo	CPU 情報の使用区分
Code	APIUseMacAddress	MAC アドレスの使用区分

Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス
Code	APIWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード（基本認証使用時）
Code	APIWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名（基本認証使用時）
Code	APIWebServiceCheckPassword	Web サービス確認パスワード
Code	APIWebServiceTimeout	Web サービスのタイムアウト
Code	APIWebServiceURL	Web サービスの URL
Code	APIWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分

※設定区分：UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム（コード）で設定されるプロパティ

【解説】
電話による認証登録を実行します。

◆想定 UI 画面
当メソッドの利用時に、想定する貴社作成の UI（ユーザインターフェース）画面は次の通りです。



この画面上の「認証 ID」は、新規認証 ID 生成メソッド（APIGenerationOfNewCertificationID メソッド）の実行で設定される認証 ID プロパティ（APICertificationID プロパティ）で取得できます。

◆処理手順
一連の処理の手順は次の通りです。

[1]新規認証 ID 生成メソッド（APIGenerationOfNewCertificationID メソッド）を実行します。
これにより、認証 ID プロパティ（APICertificationID プロパティ）に新しい認証 ID が設定されます。
その内容を画面に表示します。

[2]プロダクト ID とシリアル No.をエンドユーザに画面より入力させ、プロダクト ID プロパティ（APIProductID プロパティ）とシリアル No. プロパティ（APISerialNo プロパティ）にそれぞれ設定します。あるいは、別の方法で結果としてプロダクト ID プロパティ（APIProductID プロパティ）とシリ

アル No. プロパティ(APISerialNo プロパティ)にそれぞれ認証登録するプロダクト ID とシリアル No. を設定します。

[3]エンドユーザに貴社に電話をしてもらいます。貴社のオペレータより聞いたライセンスキーを画面より入力させ、その値をライセンスキープロパティ(APILicenseKey プロパティ)に設定します。

[4]上記の【必須設定プロパティ】一覧にある「設定区分」が「Code」のプロパティに適切な値を設定します。
★これらのプロパティは、一度設定した内容で変更がなければ再設定する必要はありません。

[5]上記の「想定 UI 画面」の「登録」ボタンに連動させるなどして、当メソッド(APIActivateRegisterTelephone メソッド)を実行します。

APIActivateRemoveInternet メソッド

【機能】
インターネットによる認証解除を実行します。

【構文】
<VB.NET>
Public Function **APIActivateRemoveInternet**() As **Boolean**

<C#>
public **bool** **APIActivateRemoveInternet**()

<VC++>
public：**VARIANT_BOOL**

NewtoneNRDvcpp::INRD_APIActivation::APIActivateRemoveInternet()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	機能
UI	APICertificationID	（登録解除する）認証 ID（取得専用）。（レジストリからの）認証登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行により、登録済みの認証 ID が当プロパティに設定されます。
UI	APIProductID	（認証解除する）プロダクト ID（レジストリからの）認証登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行により、登録済みのプロダクト ID が当プロパティに設定されます。
UI	APISerialNo	（認証解除する）シリアル No.（レジストリからの）認証登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行により、登録済みのシリアル No. が当プロパティに設定されます。

UI	APIUseProxyServer	プロキシサーバーの使用区分	
UI	APIProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時有効
UI	APIProxyServerPort	プロキシサーバーのポート	
UI	APIProxyServerUserName	プロキシサーバーのユーザ名	
UI	APIProxyServerPassword	プロキシサーバーのパスワード	
Code	APIDisabledNICIgnore	処理を高速化するため、無効になっている NIC(Network Interface Card)を無視します。	
Code	APIEncryptionPassword	暗号化時のパスワード	
Code	APIEncryptionSaltString	暗号化時の Salt 文字列	
Code	APITrialPeriod	猶予(試用)日数	
Code	APIUseCpuInfo	CPU 情報の使用区分	
Code	APIUseMacAddress	MAC アドレスの使用区分	
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス	
Code	APIWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード	
Code	APIWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名	
Code	APIWebServiceCheckPassword	Web サービス確認パスワード	
Code	APIWebServiceTimeout	Web サービスのタイムアウト	
Code	APIWebServiceURL	Web サービスの URL	
Code	APIWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分	

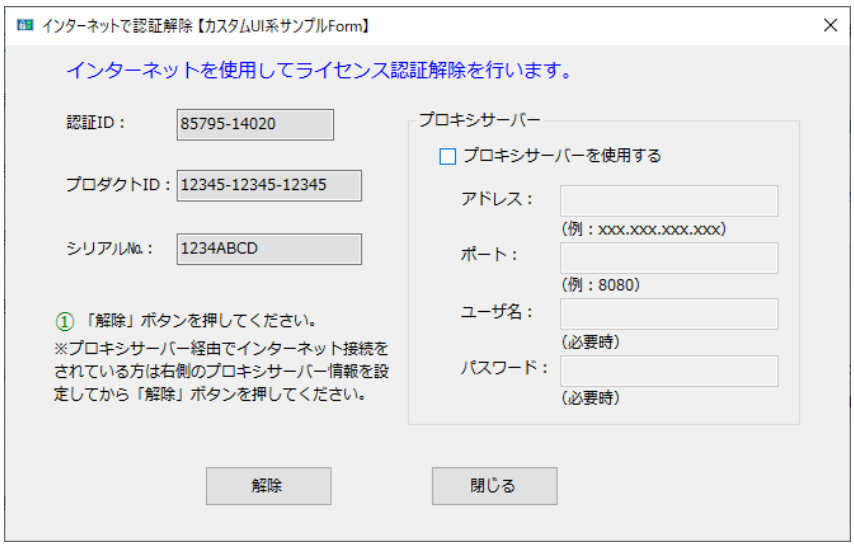
※設定区分:UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム(コード)で設定されるプロパティ

【解説】

インターネットによる認証解除を実行します。

◆想定 UI 画面

当メソッドの利用時に、想定する貴社作成の UI(ユーザインターフェース)画面は次の通りです。



この画面上の「認証 ID」、「プロダクト ID」、「シリアル No.」は、（レジストリからの）認証登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行で設定される認証 ID プロパティ（APICertificationID プロパティ）、プロダクト ID プロパティ（APIProductID プロパティ）、シリアル No.プロパティ（APISerialNo プロパティ）でそれぞれ取得できます。

◆処理手順

一連の処理の手順は次の通りです。

[1]（レジストリからの）認証登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）を実行します。これにより、認証 ID プロパティ、プロダクト ID プロパティ、シリアル No.プロパティに各値が設定されます。

[2]認証 ID プロパティ、プロダクト ID プロパティ、シリアル No.プロパティの各値を画面に表示します。

[3]上記の【必須設定プロパティ】一覧にある「設定区分」が「Code」のプロパティに適切な値を設定します。

★これらのプロパティは、一度設定した内容で変更がなければ再設定する必要はありません。

[4]上記の「想定 UI 画面」の「解除」ボタンに連動させるなどして、当メソッド（APIActivateRemoveInternet メソッド）を実行します。

APIActivateRemoveTelephone メソッド

【機能】
電話による認証解除を実行します。

【構文】
<VB.NET>
Public Function **APIActivateRemoveTelephone**() As **Boolean**

<C#>
public **bool** **APIActivateRemoveTelephone**()

<VC++>
public：**VARIANT_BOOL**

NewtoneNRDvcpp::INRD_APIActivation::APIActivateRemoveTelephone()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

戻り値とは別に、正常終了時には、解除ステータスプロパティ（APIReleaseStatus プロパティ）に解除ステータスが設定されます。
また、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

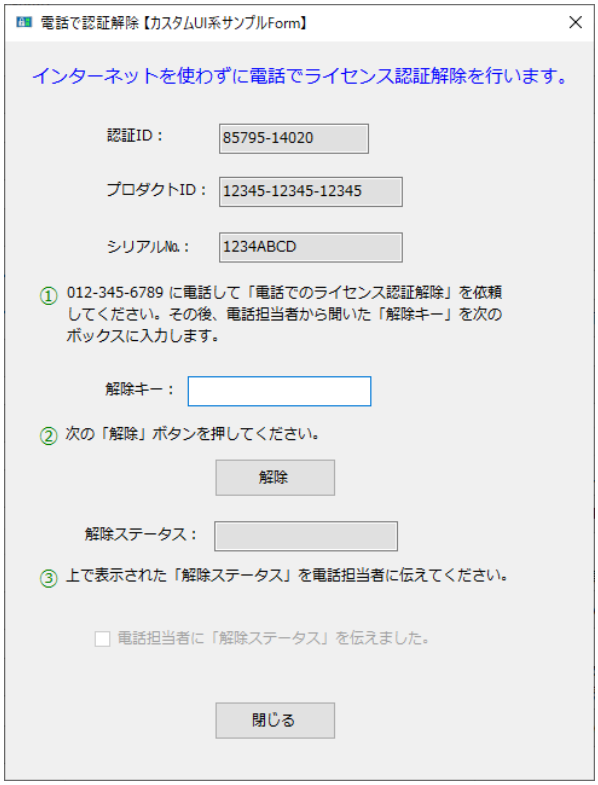
設定区分※	プロパティ	機能
UI	APICertificationID	（登録解除する）認証 ID（取得専用）。（レジストリからの）認証登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行により、登録済みの認証 ID が当プロパティに設定されます。
UI	APIProductID	（認証解除する）プロダクト ID（レジストリからの）認証登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行により、登録済みのプロダクト ID が当プロパティに設定されます。
UI	APISerialNo	（認証解除する）シリアル No.（レジストリからの）認証登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソ

		ッド）の実行により、登録済みのシリアルNo.が当プロパティに設定されます。
UI	APIReleaseKey	解除キー
UI	APIReleaseStatus	解除ステータス（取得専用）
Code	APIDisabledNICIgnore	処理を高速化するため、無効になっている NIC（Network Interface Card）を無視します。
Code	APIEncryptionPassword	暗号化時のパスワード
Code	APIEncryptionSaltString	暗号化時の Salt 文字列
Code	APITrialPeriod	猶予（試用）日数
Code	APIUseCpuInfo	CPU 情報の使用区分
Code	APIUseMacAddress	MAC アドレスの使用区分
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス
Code	APIWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード
Code	APIWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名
Code	APIWebServiceCheckPassword	Web サービス確認パスワード
Code	APIWebServiceTimeout	Web サービスのタイムアウト
Code	APIWebServiceURL	Web サービスの URL
Code	APIWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分

※設定区分：UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム（コード）で設定されるプロパティ

【解説】
電話による認証解除を実行します。

◆想定 UI 画面
当メソッドの利用時に、想定する貴社作成の UI（ユーザインターフェース）画面は次の通りです。



この画面上の「認証 ID」、「プロダクト ID」、「シリアル No.」は、（レジストリからの）認証登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行で設定される認証 ID プロパティ（APICertificationID プロパティ）、プロダクト ID プロパティ（APIProductID プロパティ）、シリアル No.プロパティ（APISerialNo プロパティ）でそれぞれ取得できます。

この画面上の「解除キー」は、貴社オペレータより電話経由でエンドユーザが聞いて画面より入力した解除キーの値を解除キープロパティに設定します。
この画面上の「解除ステータス」は、当メソッド（APIActivateRemoveTelephone メソッド）の実行後の正常終了時に解除ステータスプロパティ（APIReleaseStatus プロパティ）で取得できます。

◆処理手順
一連の処理の手順は次の通りです。

[1]（レジストリからの）認証登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）を実行します。これにより、認証 ID プロパティ、プロダクト ID プロパティ、シリアル No.プロパティに各値が設定されます。

[2]認証 ID プロパティ、プロダクト ID プロパティ、シリアル No.プロパティの各値を画面に表示します。

[3]解除キープロパティ（APIReleaseKey プロパティ）に、貴社オペレータより電話経由でエンドユーザが聞いて画面より入力した解除キーの値を設定します。

[4]上記の【必須設定プロパティ】一覧にある「設定区分」が「Code」のプロパティに適切な値を設定します。
★これらのプロパティは、一度設定した内容で変更がなければ再設定する必要はありません。

[5]上記の「想定 UI 画面」の「解除」ボタンに連動させるなどして、当メソッド（APIActivateRemoveTelephone メソッド）を実行します。

[6]当メソッド（APIActivateRemoveTelephone メソッド）の実行後、正常終了時には、解除ステータスプロパティ（APIReleaseStatus プロパティ）に解除ステータス（文字列）が設定されますので、エンドユーザにその解除ステータスを電話で貴社のオペレータに伝えてもらいます。

APIActivateStatusCheck メソッド

【機能】
エンドユーザ PC 内での認証状態の確認を行います。

【構文】
<VB.NET>
Public Function **APIActivateStatusCheck**() As **Integer**

<C#>
public **int** **APIActivateStatusCheck**()

<VC++>
public : **long NewtoneNRDvcpp::INRD_APIActivation::APIActivateStatusCheck()**

【引数】
なし

【戻り値】

0:	猶予期限切れ(猶予有効時)
1～365:	猶予日数有
366:	日付データの取得失敗(猶予)
400:	未認証(猶予無効時)
500:	認証済み
600:	フローティングライセンス登録済
1000:	欠番(旧レンタル機能関連)
1001～2100:	欠番(旧レンタル機能関連)
2101:	欠番(旧レンタル機能関連)
-999:	認証済ハードウェア情報不一致
-1:	エラー(未設定や範囲を超えているプロパティがある)
-21001231～-20000101:	認証済で終了した有効期限 (2000/01/01～2100/12/31)
-21001232:	日付データの取得失敗(有効期限)
-3:	PC の日付が変更されました。
20000101～21001231:	認証済でまだ有効な有効期限 (2000/01/01～2100/12/31)
200001010～210012310:	フローティングライセンスが登録済でまだ有効な有効期限 (2000/01/01～2100/12/31)
-210012310～-200001010:	フローティングライセンスが登録済で終了した有効期限 (2000/01/01～2100/12/31)

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	設定する内容
Code	APIDisabledNICIgnore	処理を高速化するため、無効になってい

		る NIC(Network Interface Card)を無視します。
Code	APIEncryptionPassword	暗号化時のパスワード
Code	APIEncryptionSaltString	暗号化時の Salt 文字列
Code	APITrialPeriod	猶予(試用)日数(0:猶予期間なし)
Code	APIUseCpuInfo	CPU 情報の使用区分
Code	APIUseMacAddress	MAC アドレスの使用区分
Code	APIWebServiceCheckPassword	Web サービス確認パスワード
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス

※設定区分:UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム(コード)で設定されるプロパティ

<戻り値が 200001010～210012310 または-210012310～-200001010 の場合>
戻り値を 10 で割って値が有効期限となります。
例:
戻り値が 202412200 の場合
202412200 ÷ 10 = 20241220
1～4 桁目が年→2024
5～6 桁目が月→12
7～8 桁目が日→20
よって、2024 年 12 月 20 日となります。

【解説】
この APIActivateStatusCheck メソッドと APIActivateStatusCheckOnline メソッドの相違点は APIActivateStatusCheck メソッドがエンドユーザ PC 内での認証状況を返すのに対し、APIActivateStatusCheckOnline メソッドはインターネットを介し貴社のデータベースにアクセスして取得した情報とエンドユーザ PC 内の情報とを照合して認証状況を返す点です。

APIActivateStatusCheck2 メソッド

【機能】
エンドユーザ PC 内での認証状態の確認を行います。
Windows の管理者でなくても実行可能です。

【構文】
<VB.NET>
Public Function **APIActivateStatusCheck2**() As **Integer**

<C#>
public **int** **APIActivateStatusCheck2**()

<VC++>
public : **long NewtoneNRDvcpp::INRD_APIActivation::APIActivateStatusCheck2()**

【引数】
なし

【戻り値】

0:	猶予期限切れ(猶予有効時)
1～365:	猶予日数有
366:	日付データの取得失敗(猶予)
400:	未認証(猶予無効時)
500:	認証済み
600:	フローティングライセンス登録済
1000:	欠番(旧レンタル機能関連)
1001～2100:	欠番(旧レンタル機能関連)
2101:	欠番(旧レンタル機能関連)
-999:	認証済ハードウェア情報不一致
-1:	エラー(未設定や範囲を超えているプロパティがある)
-21001231～-20000101:	認証済で終了した有効期限 (2000/01/01～2100/12/31)
-21001232:	日付データの取得失敗(有効期限)
-3:	PC の日付が変更されました。
20000101～21001231:	認証済でまだ有効な有効期限 (2000/01/01～2100/12/31)
200001010～210012310:	フローティングライセンスが登録済でまだ有効な有効期限 (2000/01/01～2100/12/31)
-210012310～-200001010:	フローティングライセンスが登録済で終了した有効期限 (2000/01/01～2100/12/31)

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	設定する内容

Code	APIDisabledNICIgnore	処理を高速化するため、無効になっている NIC（Network Interface Card）を無視します。
Code	APIEncryptionPassword	暗号化時のパスワード
Code	APIEncryptionSaltString	暗号化時の Salt 文字列
Code	APISelectRunAppDatePathFlag	「アプリ起動日」の読み込み/書き込み場所の切り替えフラグ
Code	APITrialPeriod	猶予（試用）日数（0：猶予期間なし）
Code	APIUseCpuInfo	CPU 情報の使用区分
Code	APIUseMacAddress	MAC アドレスの使用区分
Code	APIWebServiceCheckPassword	Web サービス確認パスワード
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス

※設定区分：UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム（コード）で設定されるプロパティ

<戻り値が 200001010～210012310 または-210012310～-200001010 の場合>
戻り値を 10 で割って値が有効期限となります。
例：
戻り値が 202412200 の場合
202412200 ÷ 10 = 20241220
1～4 桁目が年→2024
5～6 桁目が月→12
7～8 桁目が日→20
よって、2024 年 12 月 20 日となります。

【解説】
以下の機能以外は従来の APIActivateStatusCheck メソッドと同等です。
APISelectRunAppDatePathFlag プロパティ値による場所+従来の APIVendorsProductStartRegistryKeyPath プロパティ値の場所に「アプリ起動日」の読み込み/書き込みを行います。

このメソッドは、ユーザ様からのご要望で追加されました。
認証レスキュー！は、認証情報などをレジストリの HKEY_LOCAL_MACHINE + APIVendorsProductStartRegistryKeyPath プロパティ値の場所へ書き込み/読み込みを行っています。

APITrialPeriod プロパティ（猶予（試用）日数）が 0 以上に設定されていて、このメソッドが初めて実行された場合、残日数を確認するための開始日をレジストリに書き込みます。
その後、残日数が切れるまではレジストリへの書き込み処理はありませんでした。

しかし、バージョン 2.6.2 以降は、故意に PC の日付が変更されることを防ぐために、このメソッドを起動する度にレジストリに情報を書き込むよう変更されました。
前出のユーザ様は、Windows の管理者でなくても認証状況を確認したいとのご要望で、この故意に PC の日付が変更されることを防ぐための書き込み部分のみ「HKEY_CURRENT_USER（レジストリ）」あるいは「ProgramData（フォルダ）」へ書き込みを変更するため、このメソッドが追加されました。

APIActivateStatusCheckOnline メソッド

【機能】
オンラインで認証状態の確認を行います。

【構文】
<VB.NET>
Public Function **APIActivateStatusCheckOnline**() As **Integer**

<C#>
public **int** **APIActivateStatusCheckOnline**()

<VC++>
public：**long**
NewtoneNRDvcpp::INRD_APIActivation::APIActivateStatusCheckOnline()

【引数】
なし

【戻り値】
0: PC レベルで認証されていない（PC のレジストリには認証登録情報がない）
1: OK（PC レベルと DB で認証登録情報が一致した）
2: NG（PC レベルと一致する認証登録情報が DB にない）
3: NG（認証済ハードウェア情報不一致）
4: NG（日付データの取得失敗（猶予））
5: 欠番（旧レンタル機能関連）
6: NG（日付データの取得失敗（有効期限））
-11: 接続できない（認証時は電話）
-12: 接続できない（認証時は代理）
-13: 接続できない（フローティングライセンス）
-999: その他エラー（接続できないなど）
-1: エラー（未設定や範囲を超えているプロパティがある）
-3: PC の日付が変更されました。
600: フローティングライセンス登録済

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	機能	
UI	APIUseProxyServer	プロキシサーバーの使用区分	
UI	APIProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時
UI	APIProxyServerPort	プロキシサーバーのポート	
UI	APIProxyServerUserName	プロキシサーバーのユーザ名	
UI	APIProxyServerPassword	プロキシサーバーのパスワ	

		ード	
Code	APIDisabledNICIgnore	処理を高速化するため、無効になっている NIC（Network Interface Card）を無視します。	
Code	APIEncryptionPassword	暗号化時のパスワード	
Code	APIEncryptionSaltString	暗号化時の Salt 文字列	
Code	APITrialPeriod	猶予（試用）日数（0：猶予期間なし）	
Code	APIUseCpuInfo	CPU 情報の使用区分	
Code	APIUseMacAddress	MAC アドレスの使用区分	
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス	
Code	APIWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード（基本認証使用時）	
Code	APIWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名（基本認証使用時）	
Code	APIWebServiceCheckPassword	Web サービス確認パスワード	
Code	APIWebServiceTimeout	Web サービスのタイムアウト	
Code	APIWebServiceURL	Web サービスの URL	
Code	APIWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分	

※設定区分：UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム（コード）で設定されるプロパティ

【解説】

この APIActivateStatusCheckOnline メソッドと APIActivateStatusCheck メソッドとの相違点は APIActivateStatusCheck メソッドがエンドユーザ PC 内での認証状況を返すのに対し、APIActivateStatusCheckOnline メソッドはインターネットを介し貴社のデータベースにアクセスして取得した情報とエンドユーザ PC 内の情報とを照合して認証状況を返す点です。

この APIActivateStatusCheckOnline メソッドは、なんらかの理由で現在エンドユーザが使用している貴社のアプリケーションを使用不可としたい場合などに利用できます。

認証登録の場合、レジストリとハード情報だけの確認で OK となってしまい貴社がデータベース（DB）上の当該情報を削除してもエンドユーザの PC ではアプリケーションが継続して使用できてしまいます。そこで、任意のタイミングでインターネットを介し貴社のデータベースにアクセスしてそれらの情報を確認できる機能がこのメソッドです。貴社はたとえば、アプリケーションの起動時にそのメソッドを利用するコードを記述し、その戻り値によってアプリケーションを（強制的に）終了する、といった挙動を制御できます。

APIActivateStatusCheckOnline2 メソッド

【機能】
オンラインで認証状態の確認を行います。
Windows の管理者でなくても実行可能です。

【構文】
<VB.NET>
Public Function **APIActivateStatusCheckOnline2**() As **Integer**

<C#>
public **int** **APIActivateStatusCheckOnline2**()

<VC++>
public：**long**

NewtoneNRDvcpp::INRD_APIActivation::APIActivateStatusCheckOnline2()

【引数】
なし

【戻り値】
0: PC レベルで認証されていない（PC のレジストリには認証登録情報がない）
1: OK（PC レベルと DB で認証登録情報が一致した）
2: NG（PC レベルと一致する認証登録情報が DB にない）
3: NG（認証済ハードウェア情報不一致）
4: NG（日付データの取得失敗（猶予））
5: 欠番（旧レンタル機能関連）
6: NG（日付データの取得失敗（有効期限））
-11: 接続できない（認証時は電話）
-12: 接続できない（認証時は代理）
-13: 接続できない（フローティングライセンス）
-999: その他エラー（接続できないなど）
-1: エラー（未設定や範囲を超えているプロパティがある）
-3: PC の日付が変更されました。
600: フローティングライセンス登録済

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	機能	
UI	APIUseProxyServer	プロキシサーバーの使用区分	
UI	APIProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時
UI	APIProxyServerPort	プロキシサーバーのポート	
UI	APIProxyServerUserName	プロキシサーバーのユーザ名	

UI	APIProxyServerPassword	プロキシサーバーのパスワード	
Code	APIDisabledNICIgnore	処理を高速化するため、無効になっている NIC(Network Interface Card)を無視します。	
Code	APIEncryptionPassword	暗号化時のパスワード	
Code	APIEncryptionSaltString	暗号化時の Salt 文字列	
Code	APITrialPeriod	猶予(試用)日数(0:猶予期間なし)	
Code	APIUseCpuInfo	CPU 情報の使用区分	
Code	APIUseMacAddress	MAC アドレスの使用区分	
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス	
Code	APIWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード(基本認証使用時)	
Code	APIWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名(基本認証使用時)	
Code	APIWebServiceCheckPassword	Web サービス確認パスワード	
Code	APIWebServiceTimeout	Web サービスのタイムアウト	
Code	APIWebServiceURL	Web サービスの URL	
Code	APIWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分	

※設定区分:UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム(コード)で設定されるプロパティ

【解説】

以下の機能以外は従来の APIActivateStatusCheckOnline メソッドと同等です。
APISelectRunAppDatePathFlag プロパティ値による場所+従来の
APIVendorsProductStartRegistryKeyPath プロパティ値の場所に「アプリ起動日」の読み込み/書き込みを行います。

このメソッドは、ユーザ様からのご要望で追加されました。
認証レスキュー！は、認証情報などをレジストリの
HKEY_LOCAL_MACHINE + APIVendorsProductStartRegistryKeyPath プロパティ値の場所へ書き込み/読み込みを行っています。

APITrialPeriod プロパティ(猶予(試用)日数)が 0 以上に設定されていて、このメソッドが初めて実行された場合、残日数を確認するための開始日をレジストリに書き込みます。
その後、残日数が切れるまではレジストリへの書き込み処理はありませんでした。

しかし、バージョン 2.6.2 以降は、故意に PC の日付が変更されることを防ぐために、このメソッドを起動する度にレジストリに情報を書き込むよう変更されました。
前出のユーザ様は、Windows の管理者でなくても認証状況を確認したいとのご要望で、この故意に PC の日付が変更されることを防ぐための書き込み部分のみ「HKEY_CURRENT_USER(レジストリ)」あるいは「ProgramData(フォルダ)」へ書き込みを変更するため、このメソッドが追加されました。

APIActivateStatusOnlineVerify メソッド

【機能】
オンラインで認証状態の確認を行います。

【構文】
<VB.NET>
Public Function **APIActivateStatusOnlineVerify**() As **Integer**

<C#>
public **int** **APIActivateStatusOnlineVerify**()

<VC++>
public：**long**

NewtoneNRDvcpp::INRD_APIActivation::APIActivateStatusOnlineVerify()

【引数】
なし

【戻り値】

0:	猶予期限切れ（猶予有効時）
1～365:	猶予日数有
366:	日付データの取得失敗（猶予）
400:	未認証（猶予無効時）
500:	認証済み
600:	フローティングライセンス登録済
-999:	認証済ハードウェア情報不一致
-1:	エラー（未設定や範囲を超えているプロパティがある）
-21001231～-20000101:	認証済で終了した有効期限 （2000/01/01～2100/12/31）
-21001232:	日付データの取得失敗（有効期限）
-3:	PC の日付が変更されました。
20000101～21001231:	認証済でまだ有効な有効期限 （2000/01/01～2100/12/31）
200001010～210012310:	フローティングライセンスが登録済でまだ有効な有効期限 （2000/01/01～2100/12/31）
-210012310～-200001010:	フローティングライセンスが登録済で終了した有効期限 （2000/01/01～2100/12/31）
-4:	該当する認証キーが存在しない
-5:	認証データが存在しない
-6:	Web サービスのメソッド内でエラーが発生
-7:	確認パスワードが不正
-8:	使用できない文字列が含まれています。
-9:	WebServEnvNRD.wai ファイルに問題があります
7777:	Web サービスのメソッドの呼び出しでエラーが発生
8888:	Web サービスのメソッドの呼出し後に戻り値が見付からない

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	機能	
UI	APIUseProxyServer	プロキシサーバーの使用区分	
UI	APIProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時
UI	APIProxyServerPort	プロキシサーバーのポート	
UI	APIProxyServerUserName	プロキシサーバーのユーザ名	
UI	APIProxyServerPassword	プロキシサーバーのパスワード	
Code	APIDisabledNICIgnore	処理を高速化するため、無効になっている NIC(Network Interface Card)を無視します。	
Code	APIEncryptionPassword	暗号化時のパスワード	
Code	APIEncryptionSaltString	暗号化時の Salt 文字列	
Code	APITrialPeriod	猶予(試用)日数(0:猶予期間なし)	
Code	APIUseCpuInfo	CPU 情報の使用区分	
Code	APIUseMacAddress	MAC アドレスの使用区分	
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス	
Code	APIWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード(基本認証使用時)	
Code	APIWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名(基本認証使用時)	
Code	APIWebServiceCheckPassword	Web サービス確認パスワード	
Code	APIWebServiceTimeout	Web サービスのタイムアウト	
Code	APIWebServiceURL	Web サービスの URL	
Code	APIWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分	

※設定区分:UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム(コード)で設定されるプロパティ

<戻り値が 200001010～210012310 または-210012310～-200001010 の場合>
戻り値を 10 で割って値が有効期限となります。
例:
戻り値が 202412200 の場合
202412200 ÷ 10 = 20241220
1～4 桁目が年→2024
5～6 桁目が月→12
7～8 桁目が日→20
よって、2024 年 12 月 20 日となります。

【解説】
この APIActivateStatusOnlineVerify メソッドと APIActivateStatusCheckOnline メソッドとの相違点はフローティングライセンス上限数、使用数、状況の情報、使用中の PC 一覧データがプロパティに設定されます。
これにより、エンドユーザ PC の現在のフローティングライセンス状況を把握でき、不要なフローティングライセンス使用を防ぐことができます。

戻り値が「600」、「-210012310 ～ -200001010」、「200001010 ～ 210012310」の場合、
次のプロパティにフローティングライセンスの各情報が設定されます。

・フローティングライセンス状況（APIFloatingLicenseState プロパティ）
（0：未登録　1：登録済　2：使用中）
・フローティングライセンス上限数（APIFloatingLicenseMaxCount プロパティ）
・フローティングライセンス使用数（APIFloatingLicenseUsedCount プロパティ）
・フローティングライセンス使用中の PC 一覧データ（APIFloatingLicenseDataInfo プロパティ）
これらのプロパティは、正常に読み込みが成功しない限り、初期値（0）が設定されます。

APIDeterminationOfProxyUpdateOfExpirationDate メソッド

【機能】
「代理有効期限更新/確定」を実行します。(有効期限を更新したい PC で利用)

【構文】
<VB.NET>
Public Function **APIDeterminationOfProxyUpdateOfExpirationDate** () As **Boolean**

<C#>
public **bool** **APIDeterminationOfProxyUpdateOfExpirationDate**()

<VC++>
public : **VARIANT_BOOL**
NewtoneNRDvcpp::INRD_APIActivation::APIDeterminationOfProxyUpdateOfExpirationDate()

【引数】
なし

【戻り値】
True: 正常
False: エラー(未設定や範囲を超えているプロパティがある)

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

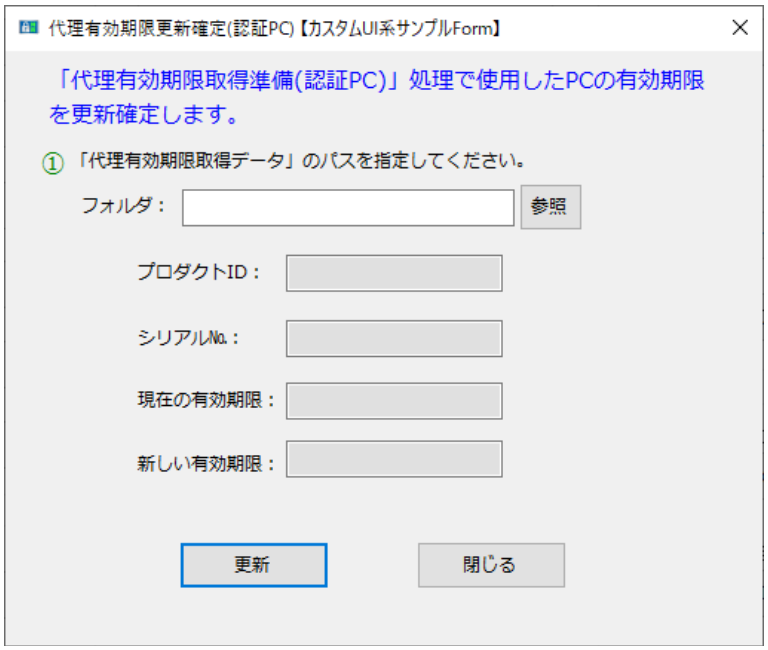
設定区分※	プロパティ	設定する内容
UI	APIProxyDataPath	(有効期限を更新する)代理有効期限取得データのパス ファイルの拡張子は「.NRS」です。
UI	APIProductID	(有効期限を更新する)プロダクト ID 代理有効期限取得データの読み込みメソッド(APIGetProxyDataForExpirationDate メソッド)の実行により、代理有効期限取得データから取得したプロダクト ID が当プロパティに設定されます。
UI	APISerialNo	(有効期限を更新する)シリアル No. 代理有効期限取得データの読み込みメソッド(APIGetProxyDataForExpirationDate メソッド)の実行により、代理有効期限取得データから取得したシリアル No.が当プロパティに設定されます。
UI	APICurrentExpirationDate	(有効期限を更新する)現在の有効期限

		代理有効期限取得データの読み込みメソッド（APIGetProxyDataForExpirationDate メソッド）の実行により、代理有効期限取得データから取得した現在の有効期限が当プロパティに設定されます。
UI	APINewExpirationDate	（有効期限を更新する）新しい有効期限. 代理有効期限取得データの読み込みメソッド（APIGetProxyDataForExpirationDate メソッド）の実行により、代理有効期限取得データから取得した新しい有効期限が当プロパティに設定されます。
Code	APIDisabledNICIgnore	処理を高速化するため、無効になっている NIC（Network Interface Card）を無視します。
Code	APIEncryptionPassword	暗号化時のパスワード
Code	APIEncryptionSaltString	暗号化時の Salt 文字列
Code	APITrialPeriod	猶予（試用）日数（0：猶予期間なし）
Code	APIUseCpuInfo	CPU 情報の使用区分
Code	APIUseMacAddress	MAC アドレスの使用区分
Code	APIWebServiceCheckPassword	Web サービス確認パスワード
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス

※設定区分：UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム（コード）で設定されるプロパティ

【解説】
「代理有効期限更新/確定」を実行します。（有効期限を更新したい PC で利用）

◆想定 UI 画面
当メソッドの利用時に、想定する貴社作成の UI（ユーザインターフェース）画面は次の通りです。



この画面上の「プロダクト ID」、「シリアル No.」、「現在の有効期限」、「新しい有効期限」は、代理有効期限取得データの取得メソッド（APIGetProxyDataForExpirationDate メソッド）の実行で設定される

プロダクトIDプロパティ(APIProductIDプロパティ)、シリアルNo.プロパティ(APISerialNoプロパティ)、現在の有効期限プロパティ(APICurrentExpirationDate プロパティ)、新しい有効期限プロパティ(APINewExpirationDateプロパティ)でそれぞれ取得できます。

◆処理手順
一連の処理の手順は次の通りです。

[1] 代理有効期限取得データパスをエンドユーザに画面より入力させ、代理有効期限取得データパスプロパティ(APIProxyDataPathプロパティ)に設定します。この際、ファイルの拡張子は「.NRS」とします。

[2] 代理有効期限取得データの取得メソッド(APIGetProxyDataForExpirationDate メソッド)を実行します。
これにより、プロダクト ID プロパティ(APIProductID プロパティ)、シリアル No.プロパティ(APISerialNoプロパティ)、現在の有効期限プロパティ(APICurrentExpirationDateプロパティ)、新しい有効期限プロパティ(APINewExpirationDate プロパティ)に代理有効期限取得データから取得したプロダクトID、シリアルNo.、現在の有効期限、新しい有効期限それぞれが設定されます。
それらの内容を画面に表示します。この内容表示そのものは必須ではありません。

[3]上記の【必須設定プロパティ】一覧にある「設定区分」が「Code」のプロパティに適切な値を設定します。
★これらのプロパティは、一度設定した内容で変更がなければ再設定する必要はありません。

[5]上記の「想定 UI 画面」の「更新」ボタンに連動させるなどして、当メソッド(APIDeterminationOfProxyUpdateOfExpirationDate メソッド)を実行します。

APIFloatingLicenseCancel メソッド

【機能】
フローティングライセンス登録解除を実行します。

【構文】
<VB.NET>
Public Function **APIFloatingLicenseCancel**() As **Boolean**

<C#>
public **bool** **APIFloatingLicenseCancel**()

<VC++>
public：**VARIANT_BOOL**

NewtoneNRDvcpp::INRD_APIActivation::APIFloatingLicenseCancel()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	機能
UI	APICertificationID	（解除する）認証 ID（取得専用）。 （レジストリからの）フローティングライセンス登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行により、登録済みの認証 ID が当プロパティに設定されます。
UI	APIProductID	（解除する）プロダクト ID （レジストリからの）フローティングライセンス登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行により、登録済みのプロダクト ID が当プロパティに設定されます。
UI	APISerialNo	（解除する）シリアル No. （レジストリからの）フローティングライセンス登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行により、登録済みのシリアル No. が当プロパティに設定されます。

UI	APIUseProxyServer	プロキシサーバーの使用区分	
UI	APIProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時有効
UI	APIProxyServerPort	プロキシサーバーのポート	
UI	APIProxyServerUserName	プロキシサーバーのユーザ名	
UI	APIProxyServerPassword	プロキシサーバーのパスワード	
Code	APIDisabledNICIgnore	処理を高速化するため、無効になっている NIC(Network Interface Card)を無視します。	
Code	APIEncryptionPassword	暗号化時のパスワード	
Code	APIEncryptionSaltString	暗号化時の Salt 文字列	
Code	APITrialPeriod	猶予(試用)日数(0:猶予期間なし)	
Code	APIUseCpuInfo	CPU 情報の使用区分	
Code	APIUseMacAddress	MAC アドレスの使用区分	
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス	
Code	APIWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード	
Code	APIWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名	
Code	APIWebServiceCheckPassword	Web サービス確認パスワード	
Code	APIWebServiceTimeout	Web サービスのタイムアウト	
Code	APIWebServiceURL	Web サービスの URL	
Code	APIWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分	

※設定区分:UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム(コード)で設定されるプロパティ

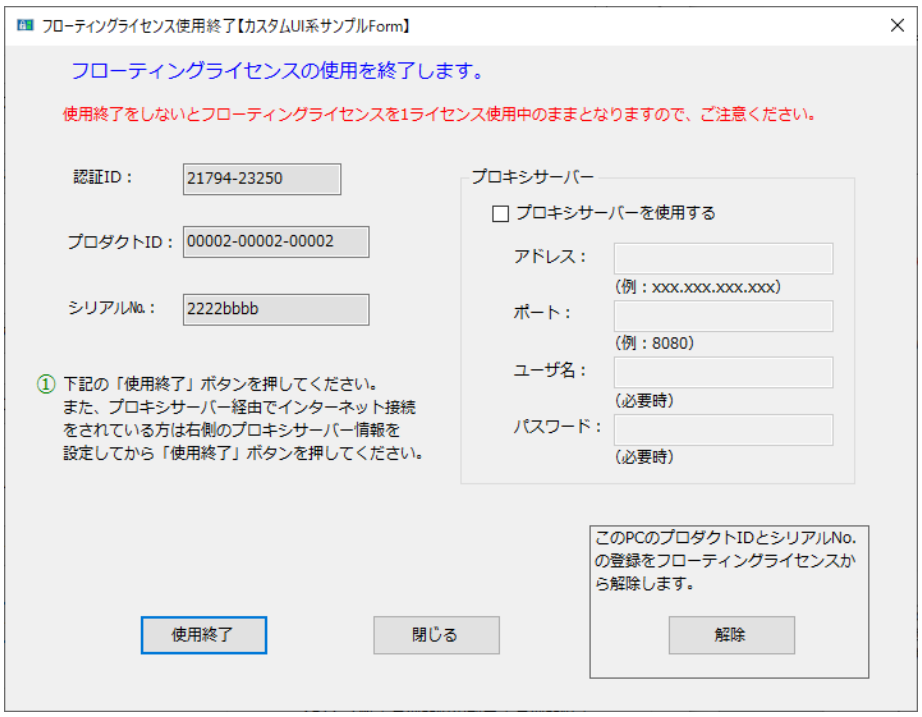
【解説】

フローティングライセンス登録解除を実行します。

このメソッドは、APITrialPeriod プロパティが 0 以上に設定されている場合、実行できません。

◆想定 UI 画面

当メソッドの利用時に、想定する貴社作成の UI(ユーザインターフェース)画面は次の通りです。



この画面上の「認証 ID」、「プロダクト ID」、「シリアル No.」は、（レジストリからの）フローティングライセンス登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行で設定される認証 ID プロパティ（APICertificationID プロパティ）、プロダクト ID プロパティ（APIProductID プロパティ）、シリアル No.プロパティ（APISerialNo プロパティ）でそれぞれ取得できます。

◆処理手順

一連の処理の手順は次の通りです。

[1]（レジストリからの）フローティングライセンス登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）を実行します。これにより、認証 ID プロパティ、プロダクト ID プロパティ、シリアル No.プロパティに各値が設定されます。

[2]認証 ID プロパティ、プロダクト ID プロパティ、シリアル No.プロパティの各値を画面に表示します。

[3]上記の【必須設定プロパティ】一覧にある「設定区分」が「Code」のプロパティに適切な値を設定します。

★これらのプロパティは、一度設定した内容で変更がなければ再設定する必要はありません。

[4]上記の「想定 UI 画面」の「解除」ボタンに連動させるなどして、当メソッド（APIFloatingLicenseCancel メソッド）を実行します。

APIFloatingLicenseFinish メソッド

【機能】
フローティングライセンス使用終了を実行します。

【構文】
<VB.NET>
Public Function **APIFloatingLicenseFinish**() As **Boolean**

<C#>
public **bool** **APIFloatingLicenseFinish**()

<VC++>
public：**VARIANT_BOOL**

NewtoneNRDvcpp::INRD_APIActivation::APIFloatingLicenseFinish()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	機能
UI	APICertificationID	（解除する）認証 ID（取得専用）。 （レジストリからの）フローティングライセンス登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行により、登録済みの認証 ID が当プロパティに設定されます。
UI	APIProductID	（解除する）プロダクト ID （レジストリからの）フローティングライセンス登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行により、登録済みのプロダクト ID が当プロパティに設定されます。
UI	APISerialNo	（解除する）シリアル No. （レジストリからの）フローティングライセンス登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行により、登録済みのシリアル No. が当プロパティに設定されます。

UI	APIUseProxyServer	プロキシサーバーの使用区分	
UI	APIProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時有効
UI	APIProxyServerPort	プロキシサーバーのポート	
UI	APIProxyServerUserName	プロキシサーバーのユーザ名	
UI	APIProxyServerPassword	プロキシサーバーのパスワード	
Code	APIDisabledNICIgnore	処理を高速化するため、無効になっている NIC(Network Interface Card)を無視します。	
Code	APIEncryptionPassword	暗号化時のパスワード	
Code	APIEncryptionSaltString	暗号化時の Salt 文字列	
Code	APITrialPeriod	猶予(試用)日数(0:猶予期間なし)	
Code	APIUseCpuInfo	CPU 情報の使用区分	
Code	APIUseMacAddress	MAC アドレスの使用区分	
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス	
Code	APIWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード	
Code	APIWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名	
Code	APIWebServiceCheckPassword	Web サービス確認パスワード	
Code	APIWebServiceTimeout	Web サービスのタイムアウト	
Code	APIWebServiceURL	Web サービスの URL	
Code	APIWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分	

※設定区分:UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム(コード)で設定されるプロパティ

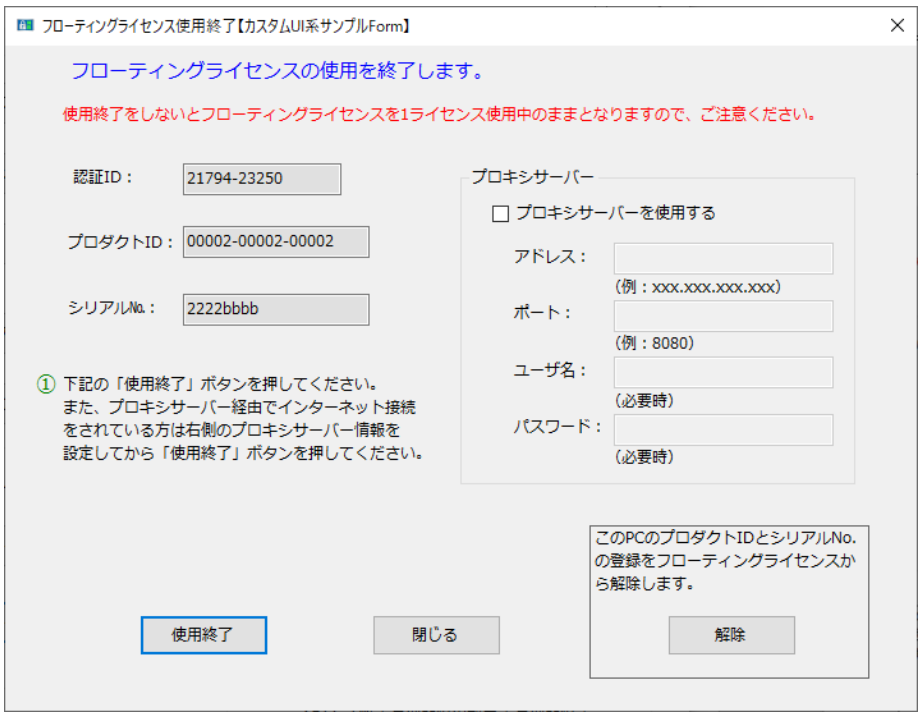
【解説】

フローティングライセンス使用終了を実行します。

このメソッドは、APITrialPeriod プロパティが 0 以上に設定されている場合、実行できません。

◆想定 UI 画面

当メソッドの利用時に、想定する貴社作成の UI(ユーザインターフェース)画面は次の通りです。



この画面上の「認証 ID」、「プロダクト ID」、「シリアル No.」は、(レジストリからの)フローティングライセンス登録済み情報の取得メソッド(APIGetRegisteredInfoFromRegistry メソッド)の実行で設定される認証 ID プロパティ(APICertificationID プロパティ)、プロダクト ID プロパティ(APIProductID プロパティ)、シリアル No.プロパティ(APISerialNo プロパティ)でそれぞれ取得できます。

◆処理手順

一連の処理の手順は次の通りです。

[1](レジストリからの)フローティングライセンス登録済み情報の取得メソッド(APIGetRegisteredInfoFromRegistry メソッド)を実行します。これにより、認証 ID プロパティ、プロダクト ID プロパティ、シリアル No.プロパティに各値が設定されます。

[2]認証 ID プロパティ、プロダクト ID プロパティ、シリアル No.プロパティの各値を画面に表示します。

[3]上記の【必須設定プロパティ】一覧にある「設定区分」が「Code」のプロパティに適切な値を設定します。

★これらのプロパティは、一度設定した内容で変更がなければ再設定する必要はありません。

[4]上記の「想定 UI 画面」の「使用終了」ボタンに連動させるなどして、当メソッド(APIFloatingLicenseFinish メソッド)を実行します。

APIFloatingLicenseRegister メソッド

【機能】
フローティングライセンス登録を実行します。

【構文】
<VB.NET>
Public Function **APIFloatingLicenseRegister**() As **Boolean**

<C#>
public **bool** **APIFloatingLicenseRegister**()

<VC++>
public：**VARIANT_BOOL**

NewtoneNRDvcpp::INRD_APIActivation::APIFloatingLicenseRegister()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	設定する内容	
UI	APICertificationID	（フローティングライセンス登録する）認証 ID（取得専用）。 新規認証 ID 生成メソッド（APIGenerationOfNewCertificationID メソッド）の実行により、新しい認証 ID が当プロパティに設定されます。	
UI	APIProductID	（フローティングライセンス登録する）プロダクト ID	
UI	APISerialNo	（フローティングライセンス登録する）シリアル No.	
UI	APIUseProxyServer	プロキシサーバーの使用区分	
UI	APIProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時
UI	APIProxyServerPort	プロキシサーバーのポート	
UI	APIProxyServerUserName	プロキシサーバーのユーザ名	
UI	APIProxyServerPassword	プロキシサーバーのパスワード	
Code	APIDisabledNICIgnore	処理を高速化するため、無効になってい	

		る NIC(Network Interface Card)を無視します。
Code	APIEncryptionPassword	暗号化時のパスワード
Code	APIEncryptionSaltString	暗号化時の Salt 文字列
Code	APITrialPeriod	猶予(試用)日数(0:猶予期間なし)
Code	APIUseCpuInfo	CPU 情報の使用区分
Code	APIUseMacAddress	MAC アドレスの使用区分
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス
Code	APIWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード(基本認証使用時)
Code	APIWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名(基本認証使用時)
Code	APIWebServiceCheckPassword	Web サービス確認パスワード
Code	APIWebServiceTimeout	Web サービスのタイムアウト
Code	APIWebServiceURL	Web サービスの URL
Code	APIWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分

※設定区分:UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム(コード)で設定されるプロパティ

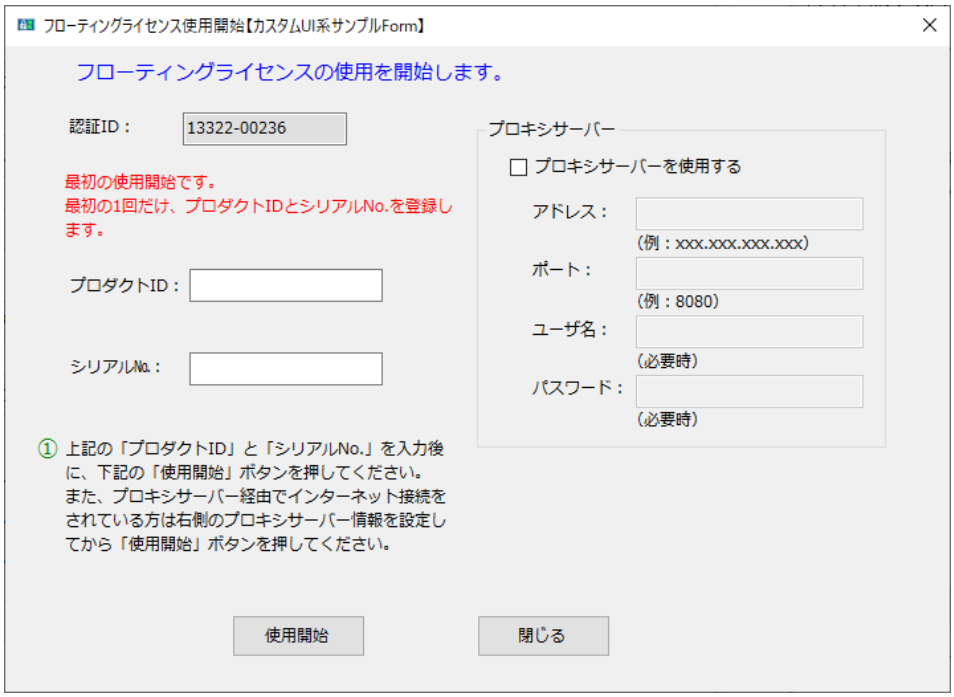
【解説】

フローティングライセンス登録を実行します。

このメソッドは、APITrialPeriod プロパティが 0 以上に設定されている場合、実行できません。

◆想定 UI 画面

当メソッドの利用時に、想定する貴社作成の UI(ユーザインターフェース)画面は次の通りです。



この画面上の「認証 ID」は、新規認証 ID 生成メソッド(APIGenerationOfNewCertificationID メソッド)の実行で設定される認証 ID プロパティ(APICertificationID プロパティ)で取得できます。

◆処理手順

一連の処理の手順は次の通りです。

[1]新規認証 ID 生成メソッド(APIGenerationOfNewCertificationID メソッド)を実行します。
これにより、認証 ID プロパティ(APICertificationID プロパティ)に新しい認証 ID が設定されます。
その内容を画面に表示します。この内容表示そのものは必須ではありません。

[2]プロダクト ID とシリアル No.をエンドユーザに画面より入力させ、プロダクト ID プロパティ(APIProductID プロパティ)とシリアル No. プロパティ(APISerialNo プロパティ)にそれぞれ設定します。あるいは、別の方法で結果としてプロダクト ID プロパティ(APIProductID プロパティ)とシリアル No. プロパティ(APISerialNo プロパティ)にそれぞれ認証登録するプロダクト ID とシリアル No.を設定します。

[3]上記の【必須設定プロパティ】一覧にある「設定区分」が「Code」のプロパティに適切な値を設定します。
★これらのプロパティは、一度設定した内容で変更がなければ再設定する必要はありません。

[4]上記の「想定 UI 画面」の「使用開始」ボタンに連動させるなどして、当メソッド(APIFloatingLicenseRegister メソッド)を実行します。

APIFloatingLicenseStart メソッド

【機能】
フローティングライセンス使用開始を実行します。

【構文】
<VB.NET>
Public Function **APIFloatingLicenseStart**() As **Boolean**

<C#>
public **bool** **APIFloatingLicenseStart**()

<VC++>
public：**VARIANT_BOOL**

NewtoneNRDvcpp::INRD_APIActivation::APIFloatingLicenseStart()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	設定する内容
UI	APICertificationID	（開始する）認証 ID（取得専用）。 （レジストリからの）フローティングライセンス登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行により、登録済みの認証 ID が当プロパティに設定されます。
UI	APIProductID	（開始する）プロダクト ID （レジストリからの）フローティングライセンス登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行により、登録済みのプロダクト ID が当プロパティに設定されます。
UI	APISerialNo	（開始する）シリアル No. （レジストリからの）フローティングライセンス登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行により、登録済みのシリアル No.が当プロパティに設定されます。

UI	APIUseProxyServer	プロキシサーバーの使用区分	
UI	APIProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時
UI	APIProxyServerPort	プロキシサーバーのポート	
UI	APIProxyServerUserName	プロキシサーバーのユーザ名	
UI	APIProxyServerPassword	プロキシサーバーのパスワード	
Code	APIDisabledNICIgnore	処理を高速化するため、無効になっている NIC(Network Interface Card)を無視します。	
Code	APIEncryptionPassword	暗号化時のパスワード	
Code	APIEncryptionSaltString	暗号化時の Salt 文字列	
Code	APITrialPeriod	猶予(試用)日数(0:猶予期間なし)	
Code	APIUseCpuInfo	CPU 情報の使用区分	
Code	APIUseMacAddress	MAC アドレスの使用区分	
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス	
Code	APIWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード(基本認証使用時)	
Code	APIWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名(基本認証使用時)	
Code	APIWebServiceCheckPassword	Web サービス確認パスワード	
Code	APIWebServiceTimeout	Web サービスのタイムアウト	
Code	APIWebServiceURL	Web サービスの URL	
Code	APIWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分	

※設定区分:UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム(コード)で設定されるプロパティ

【解説】

フローティングライセンス使用開始を実行します。

このメソッドは、APITrialPeriod プロパティが 0 以上に設定されている場合、実行できません。

◆想定 UI 画面

当メソッドの利用時に、想定する貴社作成の UI(ユーザインターフェース)画面は次の通りです。

この画面上の「認証 ID」、「プロダクト ID」、「シリアル No.」は、(レジストリからの)フローティングライセンス登録済み情報の取得メソッド(APIGetRegisteredInfoFromRegistry メソッド)の実行で設定される認証 ID プロパティ(APICertificationID プロパティ)、プロダクト ID プロパティ(APIProductID プロパティ)、シリアル No.プロパティ(APISerialNo プロパティ)でそれぞれ取得できます。

◆処理手順

一連の処理の手順は次の通りです。

[1](レジストリからの)フローティングライセンス登録済み情報の取得メソッド(APIGetRegisteredInfoFromRegistry メソッド)を実行します。これにより、認証 ID プロパティ、プロダクト ID プロパティ、シリアル No.プロパティに各値が設定されます。

[2]認証 ID プロパティ、プロダクト ID プロパティ、シリアル No.プロパティの各値を画面に表示します。

[3]上記の【必須設定プロパティ】一覧にある「設定区分」が「Code」のプロパティに適切な値を設定します。

★これらのプロパティは、一度設定した内容で変更がなければ再設定する必要はありません。

[4]上記の「想定 UI 画面」の「使用開始」ボタンに連動させるなどして、当メソッド(APIFloatingLicenseStart メソッド)を実行します。

APIGenerationOfNewCertificationID メソッド

【機能】
認証登録時に必要な、新しい認証 ID を生成します。

【構文】
<VB.NET>
Public Function **APIGenerationOfNewCertificationID**() As **Boolean**

<C#>
public **bool** **APIGenerationOfNewCertificationID**()

<VC++>
public：**VARIANT_BOOL**
NewtoneNRDvcpp::INRD_APIActivation::APIGenerationOfNewCertificationID()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	設定する内容
Code	APIDisabledNICIgnore	処理を高速化するため、無効になっている NIC（Network Interface Card）を無視します。
Code	APIEncryptionPassword	暗号化時のパスワード
Code	APIEncryptionSaltString	暗号化時の Salt 文字列
Code	APITrialPeriod	猶予（試用）日数（0：猶予期間なし）
Code	APIUseCpuInfo	CPU 情報の使用区分
Code	APIUseMacAddress	MAC アドレスの使用区分
Code	APIWebServiceCheckPassword	Web サービス確認パスワード
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス

※設定区分：UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム（コード）で設定されるプロパティ

【解説】
認証登録時に必要な、新しい認証 ID を生成します。

[1]このメソッドの戻り値は成功か失敗（True/False）です。

[2]認証登録済みの場合は、当メソッドは失敗します。

[3]このメソッドの成功時のみ、認証 ID プロパティ（APICertificationID プロパティ）に新しく生成された認証 ID が設定されます。

APIGetFreeItem メソッド

【機能】
プロダクト ID とシリアル No.を指定して、ActivationKey テーブルより自由入力項目を取得します。

【構文】
<VB.NET>
Public Function **APIGetFreeItem** () As **Boolean**

<C#>
public **bool** **APIGetFreeItem** ()

<VC++>
public : **VARIANT_BOOL NewtoneNRDvcpp::INRD_APIActivation::APIGetFreeItem()**

【引数】
なし

【戻り値】
True: 正常
False: エラー(未設定や範囲を超えているプロパティがある)

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	設定する内容	
UI	APIProductID	(認証登録する)プロダクト ID	
UI	APISerialNo	(認証登録する)シリアル No.	
UI	APIUseProxyServer	プロキシサーバーの使用区分	
UI	APIProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時
UI	APIProxyServerPort	プロキシサーバーのポート	
UI	APIProxyServerUserName	プロキシサーバーのユーザ名	
UI	APIProxyServerPassword	プロキシサーバーのパスワード	
Code	APIWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード(基本認証使用時)	
Code	APIWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名(基本認証使用時)	
Code	APIWebServiceCheckPassword	Web サービス確認パスワード	
Code	APIWebServiceTimeout	Web サービスのタイムアウト	
Code	APIWebServiceURL	Web サービスの URL	
Code	APIWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分	

※設定区分:UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム(コード)で設定されるプロパティ

【解説】
プロダクト ID とシリアル No.を指定して、ActivationKeyTable より自由入力項目を取得します。当メソッド（APIGetFreeItem メソッド）実行後、APIFreeItem1～APIFreeItem5 プロパティに値が設定されます。

APIGetProductIdSerialNoList メソッド

【機能】
外部データベースとのリンク用キー項目を指定し ActivationKey テーブルよりプロダクト ID とシリアル No.のペア文字列の列挙を取得します。

【構文】
<VB.NET>
Public Function **APIGetProductIdSerialNoList** () As **Integer**

<C#>
public **int APIGetProductIdSerialNoList** ()

<VC++>
public : **long NewtoneNRDvcpp::INRD_APIActivation::APIGetProductIdSerialNoList()**

【引数】
なし

【戻り値】
取得したプロダクト ID とシリアル No.のペアリングの数。
エラー時は 0 が設定され、戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

また、APIProductIdSerialNoList プロパティに取得したプロダクト ID とシリアル No.をデリミタのカンマ(,)で区切った文字列が設定されます。該当するデータが存在しない時は、""(空文字列)が設定されます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	設定する内容	
UI	APIExternalLinkKey	APIGetProductIdSerialNoList メソッド 実行時の外部データベースとのリンク用キー項目を設定します。	
UI	APIUseProxyServer	プロキシサーバーの使用区分	
UI	APIProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時
UI	APIProxyServerPort	プロキシサーバーのポート	
UI	APIProxyServerUserName	プロキシサーバーのユーザ名	
UI	APIProxyServerPassword	プロキシサーバーのパスワード	
Code	APIWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード(基本認証使用時)	
Code	APIWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名(基本認証使用時)	
Code	APIWebServiceCheckPassword	Web サービス確認パスワード	

Code	APIWebServiceTimeout	Web サービスのタイムアウト
Code	APIWebServiceURL	Web サービスの URL
Code	APIWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分

※設定区分：UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム（コード）で設定されるプロパティ

【解説】

APIExternalLinkKey プロパティに外部データベースとのリンク用キーを指定し ActivationKey テーブルよりプロダクト ID とシリアル No.のペア文字列の列挙を取得します。

認証レスキュー！とは別の外部データベースとのリンク用キーを APIExternalLinkKey プロパティに設定し、当メソッド（APIGetProductIdSerialNoList メソッド）を実行すると、認証レスキュー！のデータベースの ActivationKey テーブルより該当するプロダクト ID とシリアル No.のペア文字列の列挙をデリミタ付きで APIProductIdSerialNoList プロパティに設定して返します。

プロダクト ID とシリアル No.のペア文字列の列挙は、たとえば次のような文字列です。
デリミタはカンマ（, ）です。
1111-1111-1111,aaaa1111,22222-22222-22222,bbbb2222,33333-33333-33333,cccc3333

この例では次の 3 組のプロダクト ID とシリアル No.が返されました。
プロダクト ID =”11111-11111-11111” シリアル No.=”aaaa1111”
プロダクト ID =”22222-22222-22222” シリアル No.=”bbbb2222”
プロダクト ID =”33333-33333-33333” シリアル No.=”cccc3333”

当メソッド（APIGetProductIdSerialNoList メソッド）の具体的な利用方法の例は次のようなものがあります。

貴社のアプリケーション内でエンドユーザに対し、複数のプロダクト ID とシリアル No.のペアを一括で認証登録させたい場合を考えます。

手順 1
貴社のアプリケーション内の一括認証処理の UI 上でエンドユーザにユニークな「リンク用キー」（例：ユーザ ID）を入力させる。

手順 2
エンドユーザに入力させた「リンク用キー」を APIExternalLinkKey プロパティに設定し、他の必須設定プロパティの設定後、当メソッド（APIGetProductIdSerialNoList メソッド）を実行する。

手順 3
認証レスキュー！の認証用 DLL が、「Activationkey テーブル」上のその「リンク用キー」に該当する全レコード分のプロダクト ID とシリアル No.の列挙をデリミタ付きの文字列として APIProductIdSerialNoList プロパティに設定して返す。

手順 4
一括認証処理の UI 上にそれらの返されたプロダクト ID とシリアル No.を一覧表示し、エンドユーザの確認を求めた後、インターネットによる認証登録（APIActivateRegisterInternet メソッド）で認証登録する。

APIGetProxyDataForExpirationDate メソッド

【機能】
代理有効期限取得データを取得します。

【構文】
<VB.NET>
Public Function **APIGetProxyDataForExpirationDate**() As **Boolean**

<C#>
public **bool** **APIGetProxyDataForExpirationDate** ()

<VC++>
public : **VARIANT_BOOL**
NewtoneNRDvcpp::INRD_APIActivation::APIGetProxyDataForExpirationDate()

【引数】
なし

【戻り値】
True: 正常
False: エラー(未設定や範囲を超えているプロパティがある)

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	機能
UI	APIProxyDataPath	(有効期限を更新する)代理有効期限取得データのパス ファイルの拡張子は「.NRS」です。
Code	APIEncryptionPassword	暗号化時のパスワード
Code	APIEncryptionSaltString	暗号化時の Salt 文字列

※設定区分:UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム(コード)で設定されるプロパティ

【解説】
代理有効期限取得データを取得します。

[1]このメソッドの戻り値は成功か失敗(True/False)です。

[2]代理認証データのパスや内容が正しくない場合は、当メソッドは失敗します。

[3]このメソッドの成功時のみ、次表の各プロパティに値が設定されます。
代理有効期限更新関連のメソッドによって、事前に当メソッド(APIGetProxyDataForExpirationDate メソッド)の実行が必要かどうかや設定されるプロパティが異なります。

◆当メソッド(APIGetProxyDataForExpirationDate メソッド)の利用形態

代理有効期限更新関連のメソッド	事前の当メソッドの実行	プロパティ			
		プロダクト ID プロパティ（APIProductID プロパティ）	シリアル No. プロパティ（APISerialNo プロパティ）	現在の有効期限（取得専用）（APICurrentExpirationDate プロパティ）	新しい有効期限（取得専用）（APINewExpirationDate プロパティ）
代理有効期限更新準備（APIPreparationOfProxyUpdateOfExpirationDate メソッド）	不要	設定不要	設定不要	設定不要	設定不要
代理有効期限取得（APIGetProxyUpdateOfExpirationDate メソッド）	必要	当メソッドで設定	当メソッドで設定	当メソッドで設定	APIGetProxyUpdateOfExpirationDate メソッドより取得
代理有効期限更新確定（APIDeterminationOfProxyUpdateOfExpirationDate メソッド）	必要	当メソッドで設定	当メソッドで設定	当メソッドで設定	当メソッドで設定

[4]当メソッドの実行には上記の【必須設定プロパティ】にある、代理有効期限取得データのパス（APIProxyDataPath プロパティ）に適切な設定が必要です。

APIGetProxyDataForRegister メソッド

【機能】
代理認証登録データを取得します。
（代理認証機能については、「代理認証機能について」を参照）

【構文】
<VB.NET>
Public Function **APIGetProxyDataForRegister** () As **Boolean**

<C#>
public **bool** **APIGetProxyDataForRegister** ()

<VC++>
public : **VARIANT_BOOL**
NewtoneNRDvcpp::INRD_APIActivation::APIGetProxyDataForRegister()

【引数】
なし

【戻り値】
True:　正常
False:　エラー（未設定や範囲を超えているプロパティがある）

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	機能
Code	APIEncryptionPassword	暗号化時のパスワード
Code	APIEncryptionSaltString	暗号化時の Salt 文字列
UI	APIProxyDataPath	（認証登録する）代理認証データのパス。 ファイルの拡張子は「.NRS」です。

※設定区分：UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム（コード）で設定されるプロパティ

【解説】
代理認証登録データを取得します。

[1]このメソッドの戻り値は成功か失敗（True/False）です。

[2]代理認証データのパスや内容が正しくない場合は、当メソッドは失敗します。

[3]このメソッドの成功時のみ、次表の各プロパティに値が設定されます。
代理認証関連のメソッドによって、事前に当メソッド（APIGetProxyDataForRegister メソッド）の実行が必要かどうかや設定されるプロパティが異なります。

◆当メソッド（APIGetProxyDataForRegister メソッド）の利用形態

代理認証 関連のメソッド	事前の当メソッドの実行	プロパティ		
		認証 ID プロパティ（APICertificationID プロパティ）	プロダクト ID プロパティ（APIProductID プロパティ）	シリアル No.プロパティ（APISerialNo プロパティ）
代理認証登録準備（APIProxyActivateRegisterPrepare メソッド）	不要	設定不要	設定不要	設定不要
代理認証登録実行（APIProxyActivateRegisterExecute メソッド）	必要	当メソッドで設定	UI 画面上などから設定	UI 画面上などから設定
代理認証登録確定（APIProxyActivateRegisterFix メソッド）	必要	当メソッドで設定	当メソッドで設定	当メソッドで設定

[4]当メソッドの実行には上記の【必須設定プロパティ】にある、代理認証データパスプロパティ（APIProxyDataPath プロパティ）に適切な設定が必要です。

APIGetProxyDataForRemove メソッド

【機能】
代理認証解除データを取得します。
（代理認証機能については、「代理認証機能について」を参照）

【構文】
<VB.NET>
Public Function **APIGetProxyDataForRemove** () As **Boolean**

<C#>
public **bool** **APIGetProxyDataForRemove** ()

<VC++>
public : **VARIANT_BOOL**
NewtoneNRDvcpp::INRD_APIActivation::APIGetProxyDataForRemove()

【引数】
なし

【戻り値】
True:　正常
False:　エラー（未設定や範囲を超えているプロパティがある）

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	機能
Code	APIEncryptionPassword	暗号化時のパスワード
Code	APIEncryptionSaltString	暗号化時の Salt 文字列
UI	APIProxyDataPath	（認証登録する）代理認証データのパス。 ファイルの拡張子は「.NRS」です。

※設定区分：UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム（コード）で設定されるプロパティ

【解説】
代理認証解除データを取得します。

[1]このメソッドの戻り値は成功か失敗（True/False）です。

[2]代理認証データのパスや内容が正しくない場合は、当メソッドは失敗します。

[3]このメソッドの成功時のみ、次表の各プロパティに値が設定されます。
代理認証関連のメソッドによって、事前に当メソッド（APIGetProxyDataForRemove メソッド）の実行が必要かどうかや設定されるプロパティが異なります。

◆当メソッド（APIGetProxyDataForRemove メソッド）の利用形態

代理認証 関連のメソッド	事前の当メソッドの実行	プロパティ		
		認証 ID プロパティ（APICertificationID プロパティ）	プロダクト ID プロパティ（APIProductID プロパティ）	シリアル No.プロパティ（APISerialNo プロパティ）
代理認証解除準備（APIProxyActivateRemovePrepare メソッド）	不要	設定不要	設定不要	設定不要
代理認証解除実行（APIProxyActivateRemoveExecute メソッド）	必要	当メソッドで設定	当メソッドで設定	当メソッドで設定

[4]当メソッドの実行には上記の【必須設定プロパティ】にある、代理認証データパスプロパティ（APIProxyDataPath プロパティ）に適切な設定が必要です。

APIGetProxyUpdateOfExpirationDate メソッド

【機能】
「代理有効期限/取得」を実行します。(代理 PC で利用)

【構文】
<VB.NET>
Public Function **APIGetProxyUpdateOfExpirationDate**() As **Boolean**

<C#>
public **bool** **APIGetProxyUpdateOfExpirationDate**()

<VC++>
public : **VARIANT_BOOL**
NewtoneNRDvcpp::INRD_APIActivation::APIGetProxyUpdateOfExpirationDate()

【引数】
なし

【戻り値】
True: 正常
False: エラー(未設定や範囲を超えているプロパティがある)

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

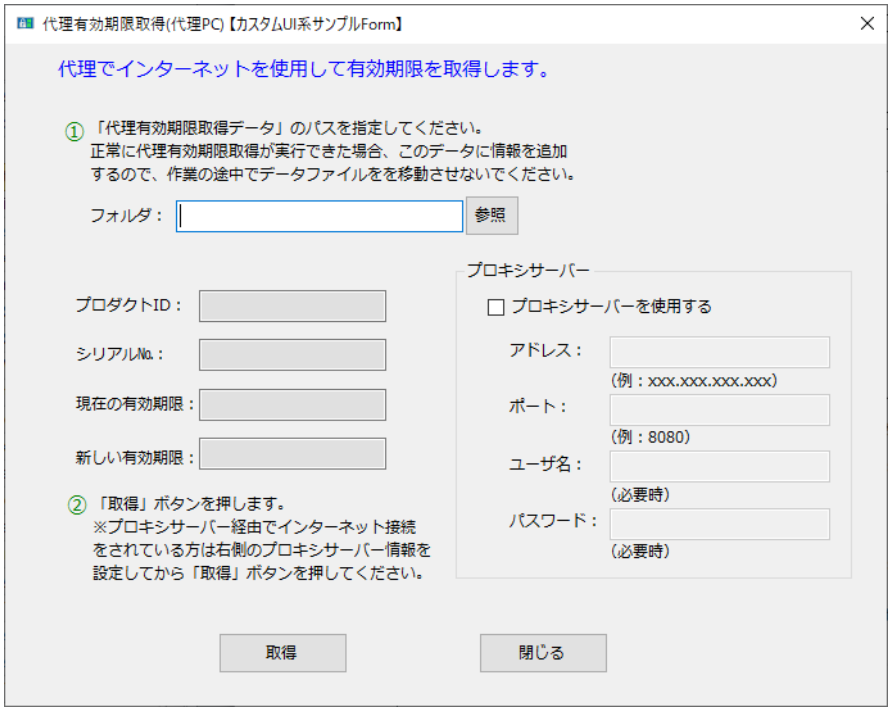
設定区分※	プロパティ	設定する内容	
UI	APIProxyDataPath	(有効期限を更新する)代理有効期限取得データのパス ファイルの拡張子は「.NRS」です。	
UI	APIProductID	(代理有効期限を取得する)プロダクト ID 代理有効期限取得データの取得メソッド(APIGetProxyDataForExpirationDate メソッド)の実行により、代理有効期限取得データから取得したプロダクト ID が当プロパティに設定されます。	
UI	APISerialNo	(代理有効期限を取得する)シリアル No. 代理有効期限取得データの取得メソッド(APIGetProxyDataForExpirationDate メソッド)の実行により、代理有効期限取得データから取得したシリアル No.が当プロパティに設定されます。	
UI	APIUseProxyServer	プロキシサーバーの使用区分	
UI	APIProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー
UI	APIProxyServerPort	プロキシサーバーのポート	

UI	APIProxyServerUserName	プロキシサーバーのユーザ名	使用時
UI	APIProxyServerPassword	プロキシサーバーのパスワード	
Code	APIEncryptionPassword	暗号化時のパスワード	
Code	APIEncryptionSaltString	暗号化時の Salt 文字列	
Code	APIWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード（基本認証使用時）	
Code	APIWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名（基本認証使用時）	
Code	APIWebServiceCheckPassword	Web サービス確認パスワード	
Code	APIWebServiceTimeout	Web サービスのタイムアウト	
Code	APIWebServiceURL	Web サービスの URL	
Code	APIWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分	

※設定区分：UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム（コード）で設定されるプロパティ

【解説】
「代理有効期限/取得」を実行します。（代理 PC で利用）

◆想定 UI 画面
当メソッドの利用時に、想定する貴社作成の UI（ユーザインターフェース）画面は次の通りです。



この画面上の「プロダクト ID」、「シリアル No.」、「現在の有効期限」は、代理有効期限取得データの取得メソッド（APIGetProxyDataForExpirationDate メソッド）の実行で設定されるプロダクト ID プロパティ（APIProductID プロパティ）、シリアル No.プロパティ（APISerialNo プロパティ）、現在の有効期限プロパティ（APICurrentExpirationDate プロパティ）で取得できます。

◆処理手順
一連の処理の手順は次の通りです。

[1] 代理有効期限取得データパスをエンドユーザに画面より入力させ、代理有効期限取得データパ

スプロパティ(APIProxyDataPath プロパティ)に設定します。この際、ファイルの拡張子は「.NRS」とします。

[2] 代理有効期限取得データの取得メソッド(APIGetProxyDataForExpirationDate メソッド)を実行します。
これにより、プロダクト ID プロパティ(APIProductID プロパティ)、シリアル No.プロパティ(APISerialNo プロパティ)、現在の有効期限プロパティ(APICurrentExpirationDate プロパティ)に代理有効期限取得データから取得したプロダクト ID、プロパティ、現在の有効期限が設定されます。その内容を画面に表示します。この内容表示そのものは必須ではありません。

[3] 代理有効期限取得データから取得したプロダクト ID とシリアル No.をプロダクト ID プロパティ(APIProductID プロパティ)とシリアル No. プロパティ(APISerialNo プロパティ)にそれぞれ設定します。あるいは、別の方法で結果としてプロダクト ID プロパティ(APIProductID プロパティ)とシリアル No. プロパティ(APISerialNo プロパティ)にそれぞれ認証登録するプロダクト ID とシリアル No.を設定します。

[4]上記の【必須設定プロパティ】一覧にある「設定区分」が「Code」のプロパティに適切な値を設定します。
★これらのプロパティは、一度設定した内容で変更がなければ再設定する必要はありません。

[5] 上記の「想定 UI 画面」の「取得」ボタンに連動させるなどして、当メソッド(APIGetProxyUpdateOfExpirationDate メソッド)を実行します。

APIGetRegisteredInfoFromRegistry メソッド

【機能】
レジストリから認証登録済みの情報を取得します。

【構文】
<VB.NET>
Public Function **APIGetRegisteredInfoFromRegistry**() As **Boolean**

<C#>
public **bool** **APIGetRegisteredInfoFromRegistry**()

<VC++>
public：**VARIANT_BOOL**
NewtoneNRDvcpp::INRD_APIActivation::APIGetRegisteredInfoFromRegistry()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	設定する内容
Code	APIDisabledNICIgnore	処理を高速化するため、無効になっている NIC（Network Interface Card）を無視します。
Code	APIEncryptionPassword	暗号化時のパスワード
Code	APIEncryptionSaltString	暗号化時の Salt 文字列
Code	APITrialPeriod	猶予（試用）日数（0：猶予期間なし）
Code	APIUseCpuInfo	CPU 情報の使用区分
Code	APIUseMacAddress	MAC アドレスの使用区分
Code	APIWebServiceCheckPassword	Web サービス確認パスワード
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス

※設定区分：UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム（コード）で設定されるプロパティ

【解説】
レジストリから認証登録済みの情報を取得します。

[1]このメソッドの戻り値は成功か失敗（True/False）。

[2]認証されていない場合は、当メソッドは失敗します。

[3]このメソッドの成功時のみ、次のプロパティに認証登録済みの各情報が設定されます。
・プロダクト ID（APIProductID プロパティ）
・シリアル No.（APISerialNo プロパティ）
・認証 ID（APICertificationID プロパティ）
・ライセンスキー（APILicenseKey プロパティ）
・現在の有効期限（APICurrentExpirationDate）
・フローティングライセンス状況（APIFloatingLicenseState プロパティ）
（0：未登録　1：登録済）

[4]これらのプロパティは、正常に読み込みが成功しない限り、初期値（空文字列または 0）が設定されます。

APIGetRegisteredInfoFromRegistry2 メソッド

【機能】
レジストリから認証登録済みの情報を取得します。
Windows の管理者でなくても実行可能です。

【構文】
<VB.NET>
Public Function **APIGetRegisteredInfoFromRegistry2**() As **Boolean**

<C#>
public **bool** **APIGetRegisteredInfoFromRegistry2**()

<VC++>
public : **VARIANT_BOOL**
NewtoneNRDvcpp::INRD_APIActivation::APIGetRegisteredInfoFromRegistry2()

【引数】
なし

【戻り値】
True: 正常
False: エラー(未設定や範囲を超えているプロパティがある)

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	設定する内容
Code	APIDisabledNICIgnore	処理を高速化するため、無効になっている NIC(Network Interface Card)を無視します。
Code	APIEncryptionPassword	暗号化時のパスワード
Code	APIEncryptionSaltString	暗号化時の Salt 文字列
Code	APISelectRunAppDatePathFlag	「アプリ起動日」の読み込み/書き込み場所の切り替えフラグ
Code	APITrialPeriod	猶予(試用)日数(0:猶予期間なし)
Code	APIUseCpuInfo	CPU 情報の使用区分
Code	APIUseMacAddress	MAC アドレスの使用区分
Code	APIWebServiceCheckPassword	Web サービス確認パスワード
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス

※設定区分:UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム(コード)で設定されるプロパティ

【解説】
以下の機能以外は従来の APIGetRegisteredInfoFromRegistry メソッドと同等です。

APISelectRunAppDatePathFlag プロパティ値による場所+従来の APIVendorsProductStartRegistryKeyPath プロパティ値の場所に「アプリ起動日」の読み込み/書き込みを行います。

このメソッドは、ユーザ様からのご要望で追加いたしました。
Windows の管理者でなくても APIActivateStatusCheck2 メソッドを使用して認証状態を確認後、認証情報を取得するために、このメソッドが必要となり追加されました。

APIGetRegisteredInfoFromWeb メソッド

【機能】
インターネットを使用して ActivationKeyTable から「ライセンス数」「有効期限の利用」「有効期限」を、ActivationDataTable から「認証 ID」「認証日時（作成日）」を取得します。

【構文】
<VB.NET>
Public Function **APIGetRegisteredInfoFromWeb**() As **String**

<C#>
public **string** **APIGetRegisteredInfoFromWeb**()

<VC++>
public : **_bstr_t**
NewtoneNRDvcpp::INRD_APIActivation::APIGetRegisteredInfoFromWeb()

【引数】
なし

【戻り値】
①設定ライセンス数
②認証済ライセンス数
③有効期限有無
④有効期限
②分の「認証 ID」と「認証日時（作成日）」
として文字列で返します。
該当する認証キーが存在しない場合は、空を返します。
また、各項目の値を デリミタの カンマ（, ）で区切った文字列が設定されます。

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	設定する内容	
UI	APIProductID	（認証登録する）プロダクト ID	
UI	APISerialNo	（認証登録する）シリアル No.	
UI	APIUseProxyServer	プロキシサーバーの使用区分	
UI	APIProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時
UI	APIProxyServerPort	プロキシサーバーのポート	
UI	APIProxyServerUserName	プロキシサーバーのユーザ名	
UI	APIProxyServerPassword	プロキシサーバーのパスワード	
Code	APIWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード（基本認証使用時）	

Code	APIWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名（基本認証使用時）
Code	APIWebServiceCheckPassword	Web サービス確認パスワード
Code	APIWebServiceTimeout	Web サービスのタイムアウト
Code	APIWebServiceURL	Web サービスの URL
Code	APIWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分

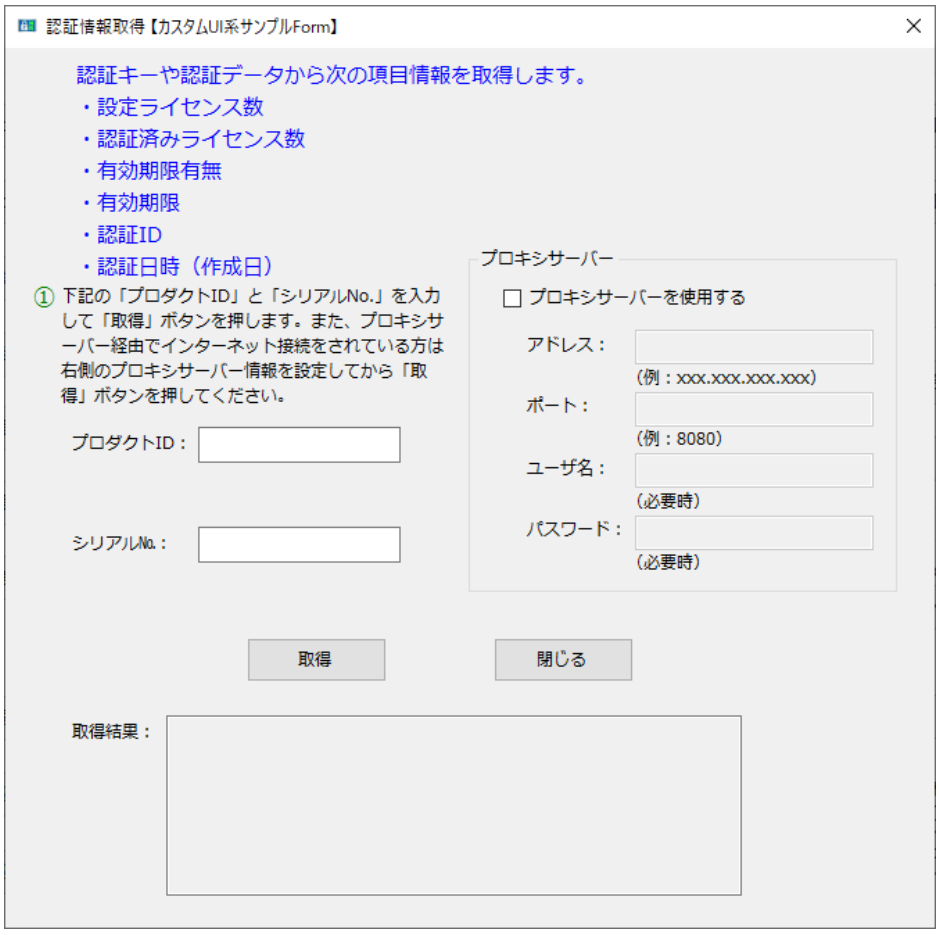
※設定区分：UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム（コード）で設定されるプロパティ

【解説】

インターネットを使用して ActivationKeyTable から「ライセンス数」「有効期限の利用」「有効期限」を、ActivationDataTable から「認証 ID」「認証日時（作成日）」を取得します。

◆想定UI画面

当メソッドの利用時に、想定する貴社作成の UI（ユーザインターフェース）画面は次の通りです。



◆処理手順

一連の処理の手順は次の通りです。

[1]プロダクト ID とシリアル No. をエンドユーザに画面より入力させ、プロダクト ID プロパティ（ APIProductID プロパティ）とシリアル No. プロパティ（ APISerialNo プロパティ）にそれぞれ設定します。あるいは、別の方法で結果としてプロダクト ID プロパティ（ APIProductID プロパティ）とシリアル No プロパティ（ APISerialNo プロパティ）にそれぞれ認証登録するプロダクト ID とシリアル No.を設定します。

[2]上記の【必須設定プロパティ】一覧にある「設定区分」が「Code」のプロパティに適切な値を設定し

ます。
★これらのプロパティは、一度設定した内容で変更がなければ再設定する必要はありません。

[3]上記の「想定 UI 画面」の「取得」ボタンに連動させるなどして、当メソッド（APIGetRegisteredInfoFromWeb メソッド）を実行します。

取得した文字列の列挙は、たとえば次のような文字列です。
デリミタはカンマ(,)です。
1,1,0,,50533-21818,2014/03/24 15:42:00

この例では、設定ライセンス 1 の認証済みの結果が返されました。
設定ライセンス数=”1”
認証済ライセンス数=”1”
有効期限有無=”0”
有効期限=””
認証 ID=”50533-21818”
認証日時(作成日)=”2014/03/24 15:42:00”

APIPreparationOfProxyUpdateOfExpirationDate メソッド

【機能】
「代理有効期限取得/準備」を実行します。(有効期限を更新したい PC で利用)

【構文】
<VB.NET>
Public Function **APIPreparationOfProxyUpdateOfExpirationDate**() As **Boolean**

<C#>
public **bool** **APIPreparationOfProxyUpdateOfExpirationDate**()

<VC++>
public : **VARIANT_BOOL**
NewtoneNRDvcpp::INRD_APIActivation::APIPreparationOfProxyUpdateOfExpiration Date()

【引数】
なし

【戻り値】
True: 正常
False: エラー(未設定や範囲を超えているプロパティがある)

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

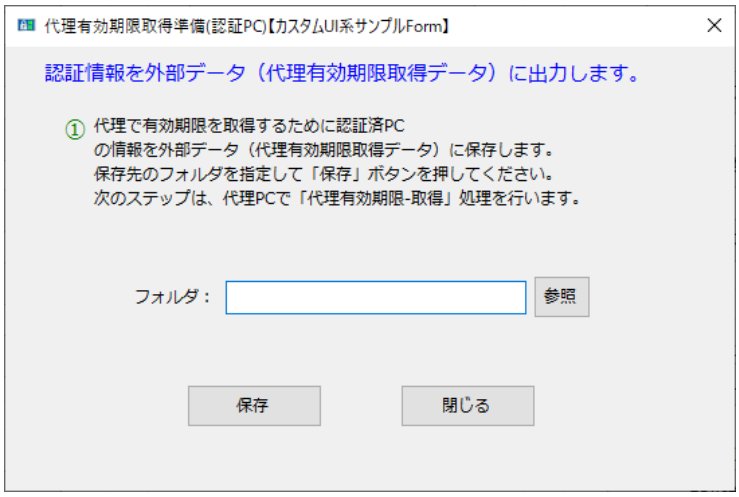
設定区分※	プロパティ	設定する内容
UI	APIProxyDataPath	(有効期限を更新する)代理有効期限取得データのパス ファイルの拡張子は「.NRS」です。
Code	APIDisabledNICIgnore	処理を高速化するため、無効になっている NIC(Network Interface Card)を無視します。
Code	APIEncryptionPassword	暗号化時のパスワード
Code	APIEncryptionSaltString	暗号化時の Salt 文字列
Code	APITrialPeriod	猶予(試用)日数
Code	APIUseCpuInfo	CPU 情報の使用区分
Code	APIUseMacAddress	MAC アドレスの使用区分
Code	APIWebServiceCheckPassword	Web サービス確認パスワード
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス

※設定区分:UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム(コード)で設定されるプロパティ

【解説】
「代理有効期限取得/準備」を実行します。(有効期限を更新したい PC で利用)

◆想定 UI 画面

当メソッドの利用時に、想定する貴社作成の UI（ユーザインターフェース）画面は次の通りです。



◆処理手順

一連の処理の手順は次の通りです。

[1] 代理有効期限取得データパスをエンドユーザに画面より入力させ、代理有効期限取得データパスプロパティ（APIProxyDataPath プロパティ）に設定します。この際、ファイルの拡張子は「.NRS」とします。

[2]上記の【必須設定プロパティ】一覧にある「設定区分」が「Code」のプロパティに適切な値を設定します。

★これらのプロパティは、一度設定した内容で変更がなければ再設定する必要はありません。

[3] 上記の「想定 UI 画面」の「保存」ボタンに連動させるなどして、当メソッド（APIPreparationOfProxyUpdateOfExpirationDate メソッド）を実行します。

APIProxyActivateRegisterExecute メソッド

【機能】
「代理認証登録/実行」を実行します。(代理 PC で利用)
(代理認証機能については、「代理認証機能について」を参照)

【構文】
<VB.NET>
Public Function **APIProxyActivateRegisterExecute**() As **Boolean**

<C#>
public **bool** **APIProxyActivateRegisterExecute**()

<VC++>
public : **VARIANT_BOOL**
NewtoneNRDvcpp::INRD_APIActivation::APIProxyActivateRegisterExecute()

【引数】
なし

【戻り値】
True: 正常
False: エラー(未設定や範囲を超えているプロパティがある)

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

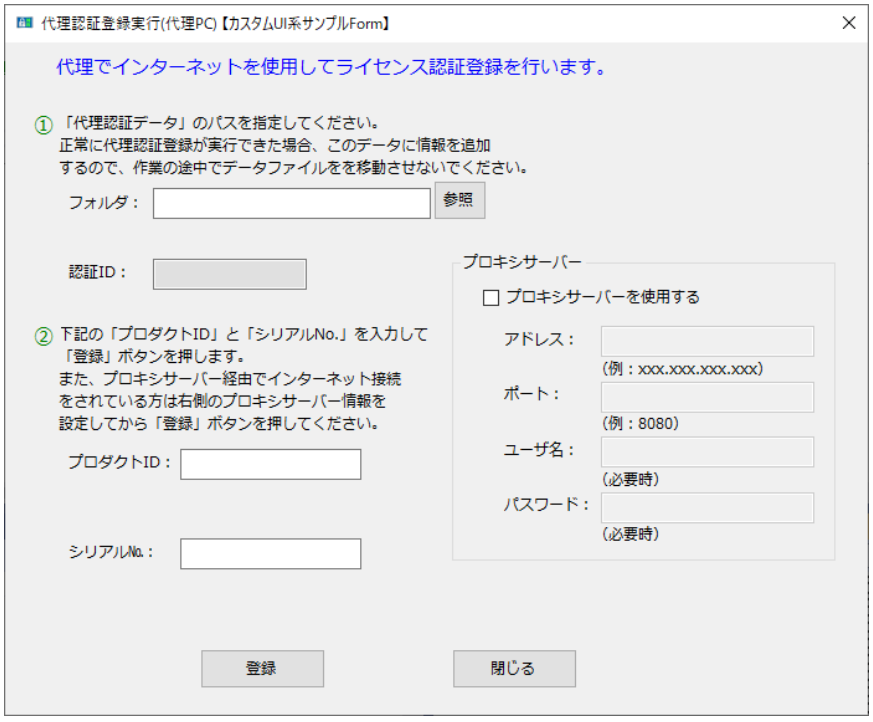
設定区分※	プロパティ	設定する内容	
UI	APIProxyDataPath	(認証登録する)代理認証データのパス。 ファイルの拡張子は「.NRS」です。	
UI	APICertificationID	(認証登録する)認証 ID(取得専用)。 代理認証登録データの取得メソッド(APIGetProxyDataForRegister メソッド)の実行により、代理認証データから取得した認証 ID が当プロパティに設定されます。	
UI	APIProductID	(認証登録する)プロダクト ID	
UI	APISerialNo	(認証登録する)シリアル No.	
UI	APIUseProxyServer	プロキシサーバーの使用区分	
UI	APIProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時有効
UI	APIProxyServerPort	プロキシサーバーのポート	
UI	APIProxyServerUserName	プロキシサーバーのユーザ名	
UI	APIProxyServerPassword	プロキシサーバーのパスワード	

Code	APIEncryptionPassword	暗号化時のパスワード
Code	APIEncryptionSaltString	暗号化時の Salt 文字列
Code	APITrialPeriod	猶予(試用)日数(0:猶予期間なし)
Code	APIUseCpuInfo	CPU 情報の使用区分
Code	APIUseMacAddress	MAC アドレスの使用区分
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス
Code	APIWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード
Code	APIWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名
Code	APIWebServiceCheckPassword	Web サービス確認パスワード
Code	APIWebServiceTimeout	Web サービスのタイムアウト
Code	APIWebServiceURL	Web サービスの URL
Code	APIWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分

※設定区分:UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム(コード)で設定されるプロパティ

【解説】
「代理認証登録/実行」を実行します。(代理 PC で利用)

◆想定 UI 画面
当メソッドの利用時に、想定する貴社作成の UI(ユーザインターフェース)画面は次の通りです。



この画面上の「認証 ID」は、代理認証登録データの取得メソッド(APIGetProxyDataForRegister メソッド)の実行で設定される認証 ID プロパティ(APICertificationID プロパティ)で取得できます。

◆処理手順
一連の処理の手順は次の通りです。

[1]代理認証データパスをエンドユーザに画面より入力させ、代理認証データパスプロパティ(APIProxyDataPath プロパティ)に設定します。この際、ファイルの拡張子は「.NRS」とします。

[2]代理認証登録データの取得メソッド(APIGetProxyDataForRegister メソッド)を実行します。

これにより、認証 ID プロパティ（APICertificationID プロパティ）に代理認証登録データから取得した認証 ID が設定されます。
その内容を画面に表示します。この内容表示そのものは必須ではありません。

[3]プロダクト ID とシリアル No.をエンドユーザに画面より入力させ、プロダクト ID プロパティ（APIProductID プロパティ）とシリアル No. プロパティ（APISerialNo プロパティ）にそれぞれ設定します。あるいは、別の方法で結果としてプロダクト ID プロパティ（APIProductID プロパティ）とシリアル No. プロパティ（APISerialNo プロパティ）にそれぞれ認証登録するプロダクト ID とシリアル No.を設定します。

[4]上記の【必須設定プロパティ】一覧にある「設定区分」が「Code」のプロパティに適切な値を設定します。
★これらのプロパティは、一度設定した内容で変更がなければ再設定する必要はありません。

[5]上記の「想定 UI 画面」の「登録」ボタンに連動させるなどして、当メソッド（APIProxyActivateRegisterExecute メソッド）を実行します。

APIProxyActivateRegisterFix メソッド

【機能】
「代理認証登録/確定」を実行します。(認証登録したい PC で利用)
(代理認証機能については、「代理認証機能について」を参照)

【構文】
<VB.NET>
Public Function **APIProxyActivateRegisterFix**() As **Boolean**

<C#>
public **bool** **APIProxyActivateRegisterFix**()

<VC++>
public : **VARIANT_BOOL**
NewtoneNRDvcpp::INRD_APIActivation::APIProxyActivateRegisterFix()

【引数】
なし

【戻り値】
True: 正常
False: エラー(未設定や範囲を超えているプロパティがある)

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

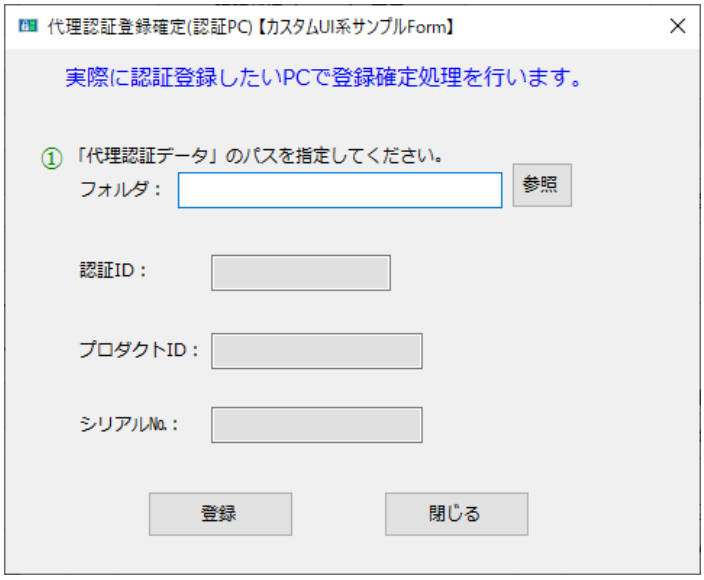
設定区分※	プロパティ	設定する内容
UI	APIProxyDataPath	(認証登録する)代理認証データのパス ファイルの拡張子は「.NRS」です。
UI	APICertificationID	(認証登録する)認証 ID(取得専用)。 代理認証登録データの取得メソッド(APIGetProxyDataForRegister メソッド)の実行により、代理認証データから取得した認証 ID が当プロパティに設定されます。
UI	APIProductID	(認証登録する)プロダクト ID 代理認証登録データの取得メソッド(APIGetProxyDataForRegister メソッド)の実行により、代理認証データから取得したプロダクト ID が当プロパティに設定されます。
UI	APISerialNo	(認証登録する)シリアル No. 代理認証登録データの取得メソッド(APIGetProxyDataForRegister メソッド)の実行により、代理認証データから取得したシリアル No.が当プロパティに設定されます。

Code	APIDisabledNICIgnore	処理を高速化するため、無効になっているNIC(Network Interface Card)を無視します。
Code	APIEncryptionPassword	暗号化時のパスワード
Code	APIEncryptionSaltString	暗号化時の Salt 文字列
Code	APITrialPeriod	猶予(試用)日数(0:猶予期間なし)
Code	APIUseCpuInfo	CPU 情報の使用区分
Code	APIUseMacAddress	MAC アドレスの使用区分
Code	APIWebServiceCheckPassword	Web サービス確認パスワード
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス

※設定区分:UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム(コード)で設定されるプロパティ

【解説】
「代理認証登録/確定」を実行します。(認証登録したい PC で利用)

◆想定 UI 画面
当メソッドの利用時に、想定する貴社作成の UI(ユーザインターフェース)画面は次の通りです。



この画面上の「認証 ID」、「プロダクト ID」、「シリアル No.」は、代理認証登録データの取得メソッド(APIGetProxyDataForRegister メソッド)の実行で設定される認証 ID プロパティ(APICertificationID プロパティ)、プロダクト ID プロパティ(APIProductID プロパティ)、シリアル No.プロパティ(APISerialNo プロパティ)でそれぞれ取得できます。

◆処理手順
一連の処理の手順は次の通りです。

[1]代理認証データパスをエンドユーザに画面より入力させ、代理認証データパスプロパティ(APIProxyDataPath プロパティ)に設定します。この際、ファイルの拡張子は「.NRS」とします。

[2]代理認証登録データの取得メソッド(APIGetProxyDataForRegister メソッド)を実行します。
これにより、認証 ID プロパティ(APICertificationID プロパティ)、プロダクト ID プロパティ(APIProductID プロパティ)、シリアル No.プロパティ(APISerialNo プロパティ)に代理認証データから取得した認証 ID、プロダクト ID、シリアル No.それぞれが設定されます。
それらの内容を画面に表示します。この内容表示そのものは必須ではありません。

[3]上記の【必須設定プロパティ】一覧にある「設定区分」が「Code」のプロパティに適切な値を設定します。
★これらのプロパティは、一度設定した内容で変更がなければ再設定する必要はありません。

[5]上記の「想定 UI 画面」の「登録」ボタンに連動させるなどして、当メソッド（APIProxyActivateRegisterFix メソッド）を実行します。

APIProxyActivateRegisterPrepare メソッド

【機能】
「代理認証登録/準備」を実行します。(認証登録したい PC で利用)
(代理認証機能については、「代理認証機能について」を参照)

【構文】
<VB.NET>
Public Function **APIProxyActivateRegisterPrepare**() As **Boolean**

<C#>
public **bool** **APIProxyActivateRegisterPrepare**()

<VC++>
public : **VARIANT_BOOL**
NewtoneNRDvcpp::INRD_APIActivation::APIProxyActivateRegisterPrepare()

【引数】
なし

【戻り値】
True: 正常
False: エラー(未設定や範囲を超えているプロパティがある)

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

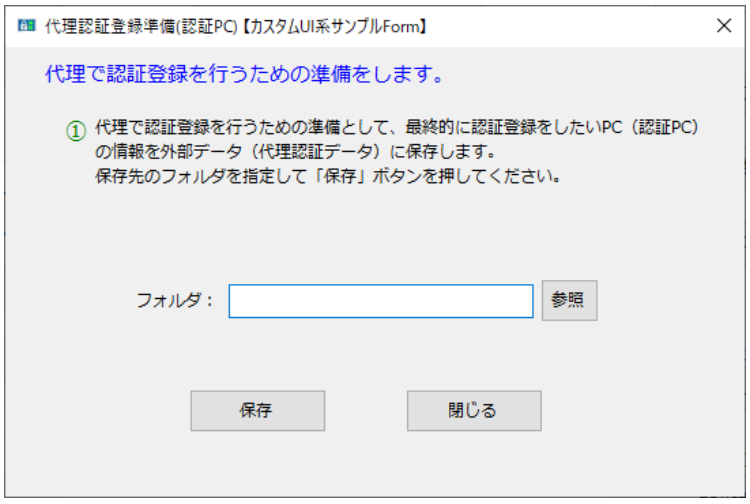
設定区分※	プロパティ	設定する内容
UI	APIProxyDataPath	(認証登録する)代理認証データのパス ファイルの拡張子は「.NRS」です。
Code	APIDisabledNICIgnore	処理を高速化するため、無効になっている NIC(Network Interface Card)を無視します。
Code	APIEncryptionPassword	暗号化時のパスワード
Code	APIEncryptionSaltString	暗号化時の Salt 文字列
Code	APITrialPeriod	猶予(試用)日数
Code	APIUseCpuInfo	CPU 情報の使用区分
Code	APIUseMacAddress	MAC アドレスの使用区分
Code	APIWebServiceCheckPassword	Web サービス確認パスワード
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス

※設定区分:UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム(コード)で設定されるプロパティ

【解説】
「代理認証登録/準備」を実行します。(認証登録したい PC で利用)

◆想定 UI 画面

当メソッドの利用時に、想定する貴社作成の UI（ユーザインターフェース）画面は次の通りです。



◆処理手順

一連の処理の手順は次の通りです。

[1]代理認証データパスをエンドユーザに画面より入力させ、代理認証データパスプロパティ（APIProxyDataPath プロパティ）に設定します。この際、ファイルの拡張子は「.NRS」とします。

[2]上記の【必須設定プロパティ】一覧にある「設定区分」が「Code」のプロパティに適切な値を設定します。

★これらのプロパティは、一度設定した内容で変更がなければ再設定する必要はありません。

[3]上記の「想定 UI 画面」の「保存」ボタンに連動させるなどして、当メソッド（APIProxyActivateRegisterPrepare メソッド）を実行します。

APIProxyActivateRemoveExecute メソッド

【機能】
「代理認証解除/実行」を実行します。(代理 PC で利用)
(代理認証機能については、「代理認証機能について」を参照)

【構文】
<VB.NET>
Public Function **APIProxyActivateRemoveExecute**() As **Boolean**

<C#>
public **bool** **APIProxyActivateRemoveExecute**()

<VC++>
public : **VARIANT_BOOL**
NewtoneNRDvcpp::INRD_APIActivation::APIProxyActivateRemoveExecute()

【引数】
なし

【戻り値】
True: 正常
False: エラー(未設定や範囲を超えているプロパティがある)

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	設定する内容
UI	APIProxyDataPath	(認証登録する)代理認証データのパス ファイルの拡張子は「.NRS」です。
UI	APICertificationID	(認証登録する)認証 ID(取得専用)。 代理認証解除データの取得メソッド(APIGetProxyDataForRemove メソッド)の実行により、代理認証データから取得した認証 ID が当プロパティに設定されます。
UI	APIProductID	(認証登録する)プロダクト ID 代理認証解除データの取得メソッド(APIGetProxyDataForRemove メソッド)の実行により、代理認証データから取得したプロダクト ID が当プロパティに設定されます。
UI	APISerialNo	(認証登録する)シリアル No. 代理認証解除データの取得メソッド(APIGetProxyDataForRemove メソッド)の実行により、代理認証データから取得した

		シリアル No.が当プロパティに設定されます。	
UI	APIUseProxyServer	プロキシサーバーの使用区分	
UI	APIProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時有効
UI	APIProxyServerPort	プロキシサーバーのポート	
UI	APIProxyServerUserName	プロキシサーバーのユーザ名	
UI	APIProxyServerPassword	プロキシサーバーのパスワード	
Code	APIEncryptionPassword	暗号化時のパスワード	
Code	APIEncryptionSaltString	暗号化時の Salt 文字列	
Code	APITrialPeriod	猶予(試用)日数	
Code	APIUseCpuInfo	CPU 情報の使用区分	
Code	APIUseMacAddress	MAC アドレスの使用区分	
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス	
Code	APIWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード	
Code	APIWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名	
Code	APIWebServiceCheckPassword	Web サービス確認パスワード	
Code	APIWebServiceTimeout	Web サービスのタイムアウト	
Code	APIWebServiceURL	Web サービスの URL	
Code	APIWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分	

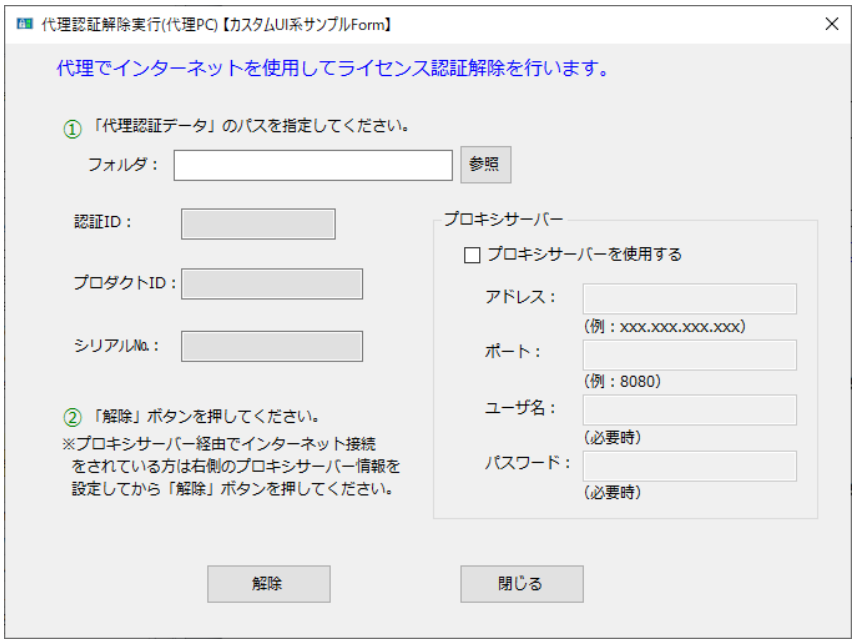
※設定区分:UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム(コード)で設定されるプロパティ

【解説】

「代理認証解除/実行」を実行します。(代理 PC で利用)

◆想定 UI 画面

当メソッドの利用時に、想定する貴社作成の UI(ユーザインターフェース)画面は次の通りです。



この画面上の「認証 ID」、「プロダクト ID」、「シリアル No.」は、代理認証解除データの取得メソッド(APIGetProxyDataForRemove メソッド)の実行で設定される認証 ID プロパティ(APICertificationID

プロパティ)、プロダクト ID プロパティ(APIProductID プロパティ)、シリアル No.プロパティ(APISerialNo プロパティ)でそれぞれ取得できます。

◆処理手順

一連の処理の手順は次の通りです。

[1]代理認証データパスをエンドユーザに画面より入力させ、代理認証データパスプロパティ(APIProxyDataPath プロパティ)に設定します。この際、ファイルの拡張子は「.NRS」とします。

[2]代理認証解除データの取得メソッド(APIGetProxyDataForRemove メソッド)を実行します。
これにより、認証 ID プロパティ(APICertificationID プロパティ)、プロダクト ID プロパティ(APIProductID プロパティ)、シリアル No.プロパティ(APISerialNo プロパティ)に代理認証データから取得した認証 ID、プロダクト ID、シリアル No.それぞれが設定されます。
それらの内容を画面に表示します。この内容表示そのものは必須ではありません。

[3]上記の【必須設定プロパティ】一覧にある「設定区分」が「Code」のプロパティに適切な値を設定します。
★これらのプロパティは、一度設定した内容で変更がなければ再設定する必要はありません。

[4]上記の「想定 UI 画面」の「解除」ボタンに連動させるなどして、当メソッド(APIProxyActivateRemoveExecute メソッド)を実行します。

APIProxyActivateRemovePrepare メソッド

【機能】
「代理認証解除/準備」を実行します。(認証登録したい PC で利用)
(代理認証機能については、「代理認証機能について」を参照)

【構文】
<VB.NET>
Public Function **APIProxyActivateRemovePrepare**() As **Boolean**

<C#>
public **bool** **APIProxyActivateRemovePrepare**()

<VC++>
public : **VARIANT_BOOL**
NewtoneNRDvcpp::INRD_APIActivation::APIProxyActivateRemovePrepare()

【引数】
なし

【戻り値】
True: 正常
False: エラー(未設定や範囲を超えているプロパティがある)

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

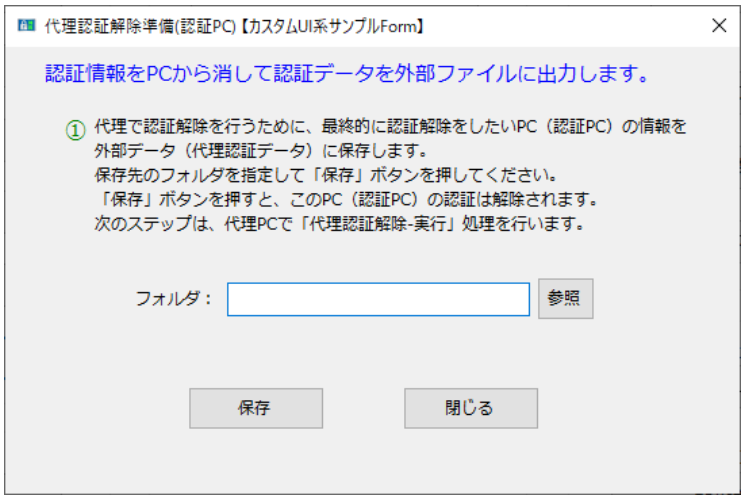
設定区分※	プロパティ	設定する内容
UI	APIProxyDataPath	(認証登録する)代理認証データのパス ファイルの拡張子は「.NRS」です。
Code	APIDisabledNICIgnore	処理を高速化するため、無効になっている NIC(Network Interface Card)を無視します。
Code	APIEncryptionPassword	暗号化時のパスワード
Code	APIEncryptionSaltString	暗号化時の Salt 文字列
Code	APITrialPeriod	猶予(試用)日数
Code	APIUseCpuInfo	CPU 情報の使用区分
Code	APIUseMacAddress	MAC アドレスの使用区分
Code	APIWebServiceCheckPassword	Web サービス確認パスワード
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス

※設定区分:UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム(コード)で設定されるプロパティ

【解説】
「代理認証解除/準備」を実行します。(認証登録したい PC で利用)

◆想定 UI 画面

当メソッドの利用時に、想定する貴社作成の UI（ユーザインターフェース）画面は次の通りです。



◆処理手順

一連の処理の手順は次の通りです。

[1]代理認証データパスをエンドユーザに画面より入力させ、代理認証データパスプロパティ（APIProxyDataPath プロパティ）に設定します。この際、ファイルの拡張子は「.NRS」とします。

[2]上記の【必須設定プロパティ】一覧にある「設定区分」が「Code」のプロパティに適切な値を設定します。

★これらのプロパティは、一度設定した内容で変更がなければ再設定する必要はありません。

[3]上記の「想定 UI 画面」の「保存」ボタンに連動させるなどして、当メソッド（APIProxyActivateRemovePrepare メソッド）を実行します。

APIReadProxyServerInfoFromRegistry メソッド

【機能】
レジストリからプロキシサーバーの情報を取得します。

【構文】
<VB.NET>
Public Function **APIReadProxyServerInfoFromRegistry** () As **Boolean**

<C#>
public **bool** **APIReadProxyServerInfoFromRegistry** ()

<VC++>
public : **VARIANT_BOOL**
NewtoneNRDvcpp::INRD_APIActivation::APIReadProxyServerInfoFromRegistry()

【引数】
なし

【戻り値】
True: 正常
False: エラー(未設定や範囲を超えているプロパティがある)

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	設定する内容
Code	APIEncryptionPassword	暗号化時のパスワード
Code	APIEncryptionSaltString	暗号化時の Salt 文字列
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス

※設定区分:UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム(コード)で設定されるプロパティ

【解説】
レジストリからプロキシサーバーの情報を取得します。

[1]このメソッドの戻り値は成功か失敗(True/False)。

[2] プロキシサーバーのユーザ名とパスワードを暗号化されおり復号化が失敗すると、当メソッドは失敗します。また、レジストリの読み込みで何らかの原因で失敗すると、当メソッドは失敗します。

[3]このメソッドの成功時のみ、次のプロパティにプロキシサーバーの各情報が設定されます。
・プロキシサーバーのアドレス(APIProxyServerAddress)
・プロキシサーバーのパスワード(APIProxyServerPassword)

・プロキシサーバーのポート（APIProxyServerPort）
・プロキシサーバーのユーザ名（APIProxyServerUserName）
・プロキシサーバーの使用区分（APIUseProxyServer）

[4]これらのプロパティは、正常に読み込みが成功しない限り、初期値（空文字列）が設定されます。

APIRestoreCancelStatus メソッド

【機能】
「認証解除状態回復」を実行します。

【構文】
<VB.NET>
Public Function **APIRestoreCancelStatus**() As **Boolean**

<C#>
public **bool** **APIRestoreCancelStatus**()

<VC++>
public：**VARIANT_BOOL**

NewtoneNRDvcpp::INRD_APIActivation::APIRestoreCancelStatus()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	機能
UI	APICertificationID	（登録解除する）認証 ID（取得専用）。（レジストリからの）認証登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行により、登録済みの認証 ID が当プロパティに設定されます。
UI	APIProductID	（認証解除する）プロダクト ID（レジストリからの）認証登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行により、登録済みのプロダクト ID が当プロパティに設定されます。
UI	APISerialNo	（認証解除する）シリアル No.（レジストリからの）認証登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行により、登録済みのシリアル No.が当プロパティに設定されます。

UI	APIUseProxyServer	プロキシサーバーの使用区分	
UI	APIProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時有効
UI	APIProxyServerPort	プロキシサーバーのポート	
UI	APIProxyServerUserName	プロキシサーバーのユーザ名	
UI	APIProxyServerPassword	プロキシサーバーのパスワード	
Code	APIDisabledNICIgnore	処理を高速化するため、無効になっている NIC（Network Interface Card）を無視します。	
Code	APIEncryptionPassword	暗号化時のパスワード	
Code	APIEncryptionSaltString	暗号化時の Salt 文字列	
Code	APITrialPeriod	猶予（試用）日数	
Code	APIUseCpuInfo	CPU 情報の使用区分	
Code	APIUseMacAddress	MAC アドレスの使用区分	
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス	
Code	APIWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード	
Code	APIWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名	
Code	APIWebServiceCheckPassword	Web サービス確認パスワード	
Code	APIWebServiceTimeout	Web サービスのタイムアウト	
Code	APIWebServiceURL	Web サービスの URL	
Code	APIWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分	

※設定区分：UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム（コード）で設定されるプロパティ

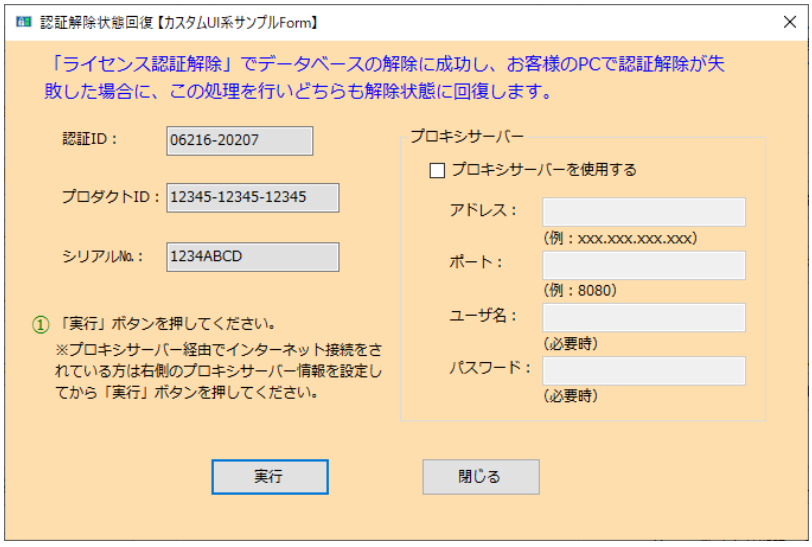
【解説】

「認証解除状態回復」を実行します。

「ライセンス認証解除」でデータベースの解除に成功したが、エンドユーザ PC での認証解除が失敗した状態になった場合に、このメソッドによる処理でエンドユーザの操作でその状態を回復しエンドユーザ PC を認証解除状態とします。

◆想定 UI 画面

当メソッドの利用時に、想定する貴社作成の UI（ユーザインターフェース）画面は次の通りです。



この画面上の「認証 ID」、「プロダクト ID」、「シリアル No.」は、（レジストリからの）認証登録済み情報

の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行で設定される認証 ID プロパティ（APICertificationID プロパティ）、プロダクト ID プロパティ（APIProductID プロパティ）、シリアル No.プロパティ（APISerialNo プロパティ）でそれぞれ取得できます。

◆処理手順

一連の処理の手順は次の通りです。

[1]（レジストリからの）認証登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）を実行します。これにより、認証 ID プロパティ、プロダクト ID プロパティ、シリアル No.プロパティに各値が設定されます。

[2]認証 ID プロパティ、プロダクト ID プロパティ、シリアル No.プロパティの各値を画面に表示します。

[3]上記の【必須設定プロパティ】一覧にある「設定区分」が「Code」のプロパティに適切な値を設定します。

★これらのプロパティは、一度設定した内容で変更がなければ再設定する必要はありません。

[4] 上記の「想定 UI 画面」の「実行」ボタンに連動させるなどして、当メソッド（APIRestoreCancelStatus メソッド）を実行します。

APIRestoreRegisterStatus メソッド

【機能】
「認証登録状態回復」を実行します。

【構文】
<VB.NET>
Public Function **APIRestoreRegisterStatus**() As **Boolean**

<C#>
public **bool** **APIRestoreRegisterStatus**()

<VC++>
public：**VARIANT_BOOL**

NewtoneNRDvcpp::INRD_APIActivation::APIRestoreRegisterStatus()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	設定する内容	
UI	APIProductID	（認証登録する）プロダクト ID	
UI	APISerialNo	（認証登録する）シリアル No.	
UI	APIUseProxyServer	プロキシサーバーの使用区分	
UI	APIProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時有効
UI	APIProxyServerPort	プロキシサーバーのポート	
UI	APIProxyServerUserName	プロキシサーバーのユーザ名	
UI	APIProxyServerPassword	プロキシサーバーのパスワード	
Code	APIDisabledNICIgnore	処理を高速化するため、無効になっている NIC（Network Interface Card）を無視します。	
Code	APIEncryptionPassword	暗号化時のパスワード	
Code	APIEncryptionSaltString	暗号化時の Salt 文字列	
Code	APITrialPeriod	猶予（試用）日数	
Code	APIUseCpuInfo	CPU 情報の使用区分	
Code	APIUseMacAddress	MAC アドレスの使用区分	
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス	

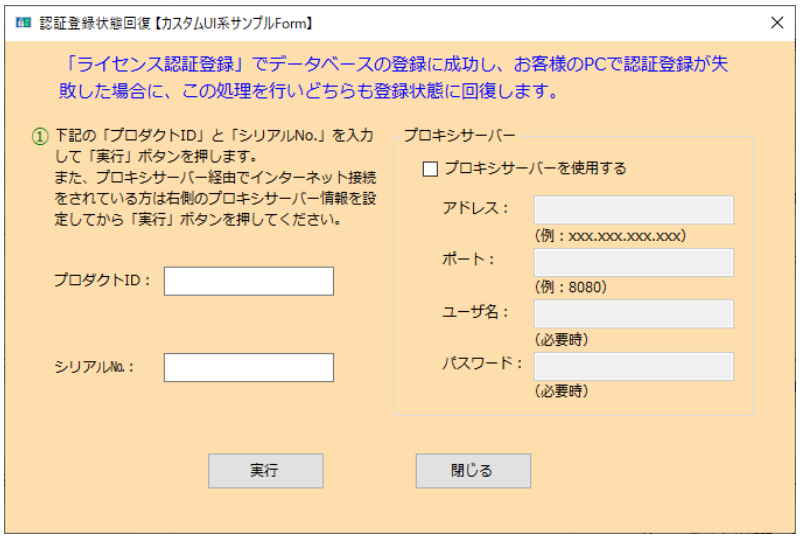
Code	APIWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード
Code	APIWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名
Code	APIWebServiceCheckPassword	Web サービス確認パスワード
Code	APIWebServiceTimeout	Web サービスのタイムアウト
Code	APIWebServiceURL	Web サービスの URL
Code	APIWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分

※設定区分：UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム（コード）で設定されるプロパティ

【解説】
「認証登録状態回復」を実行します。

「ライセンス認証登録」でデータベースの登録に成功したが、エンドユーザ PC での認証登録が失敗した状態になった場合に、このメソッドによる処理でエンドユーザの操作でその状態を回復しエンドユーザ PC を認証登録状態とします。

◆想定 UI 画面
当メソッドの利用時に、想定する貴社作成の UI（ユーザインターフェース）画面は次の通りです。



◆処理手順
一連の処理の手順は次の通りです。

[1]プロダクト ID とシリアル No.をエンドユーザに画面より入力させ、プロダクト ID プロパティ（APIProductID プロパティ）とシリアル No. プロパティ（APISerialNo プロパティ）にそれぞれ設定します。あるいは、別の方法で結果としてプロダクト ID プロパティ（APIProductID プロパティ）とシリアル No. プロパティ（APISerialNo プロパティ）にそれぞれ認証登録するプロダクト ID とシリアル No.を設定します。

[2]上記の【必須設定プロパティ】一覧にある「設定区分」が「Code」のプロパティに適切な値を設定します。
★これらのプロパティは、一度設定した内容で変更がなければ再設定する必要はありません。

[3]上記の「想定 UI 画面」の「登録」ボタンに連動させるなどして、当メソッド（APIRestoreRegisterStatus メソッド）を実行します。

APIRunNR2AppDateRemove メソッド

【機能】
「アプリ起動日」を削除します。

【構文】
<VB.NET>
Public Function **APIRunNR2AppDateRemove**() As **Boolean**

<C#>
public **bool** **APIRunNR2AppDateRemove**()

<VC++>
public：**VARIANT_BOOL**

NewtoneNRDvcpp::INRD_APIActivation::APIRunNR2AppDateRemove()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	設定する内容
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス

※設定区分：UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム（コード）で設定されるプロパティ

【解説】
「アプリ起動日」を削除します。
認証レスキュー！を含む貴社のアプリを起動した際に、レジストリに記録されている前回の「アプリ起動日」を確認します。
（もし、「アプリ起動日」が存在しなかった場合は、起動した際にレジストリに記録します。）
その「アプリ起動日」より PC の日付が古い場合、故意に PC の日付が変更されたということで「PC の日付が変更されました。（APIErrorStatus=174）」というメッセージが表示されます。

例：
たとえば、本日を 6 月 1 日とします。PC の日付を 6 月 2 日に設定しアプリを起動します。
次に、PC の日付を本日（つまり 6 月 1 日）に戻しても、「アプリ起動日」は 6 月 2 日で記録されており、PC の日付が古いので「PC の日付が変更されました。」というメッセージが表示されます。
この状態になった場合、PC の日付を 6 月 2 日に設定しない限り、認証登録等が実行できません。
このメソッドを実行後は、再度 PC の日付を本日（つまり 6 月 1 日）に設定して、アプリをご利用いた

だける状態にします。

APIRunNR2AppDateRemove2 メソッド

【機能】
「アプリ起動日」を削除します。

【構文】
<VB.NET>
Public Function **APIRunNR2AppDateRemove2**() As **Boolean**

<C#>
public **bool** **APIRunNR2AppDateRemove2**()

<VC++>
public：**VARIANT_BOOL**

NewtoneNRDvcpp::INRD_APIActivation::APIRunNR2AppDateRemove2()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	設定する内容
Code	APISelectRunAppDatePathFlag	「アプリ起動日」の読み込み/書き込み場所の切り替えフラグ
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス

※設定区分：UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム（コード）で設定されるプロパティ

【解説】
以下の機能以外は従来の APIRunNR2AppDateRemove メソッドと同等です。
APISelectRunAppDatePathFlag プロパティ値による場所+従来の
APIVendorsProductStartRegistryKeyPath プロパティ値の場所の「アプリ起動日」を削除します。

このメソッドは、ユーザ様からのご要望で追加いたしました。
APIActivateStatusCheck2 メソッドで書き込まれた「アプリ起動日」を削除します。

APITrialStartDateRemove メソッド

【機能】
「猶予日数」の「開始日」を削除します。

【構文】
<VB.NET>
Public Function **APITrialStartDateRemove**() As **Boolean**

<C#>
public **bool** **APITrialStartDateRemove**()

<VC++>
public：**VARIANT_BOOL**

NewtoneNRDvcpp::INRD_APIActivation::APITrialStartDateRemove()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	設定する内容
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス

※設定区分：UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム（コード）で設定されるプロパティ

【解説】
「猶予日数」の「開始日」を削除します。
このメソッドを実行後は、エンドユーザは再度「猶予日数」分の利用が可能になります。

APIUpdateOfExpirationDate メソッド

【機能】
「有効期限の更新」を実行します。

【構文】
<VB.NET>
Public Function **APIUpdateOfExpirationDate**() As **Boolean**

<C#>
public **bool** **APIUpdateOfExpirationDate**()

<VC++>
public：**VARIANT_BOOL**

NewtoneNRDvcpp::INRD_APIActivation::APIUpdateOfExpirationDate()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	機能
UI	APIProductID	（有効期限を更新する）プロダクト ID（レジストリからの）認証登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行により、登録済みのプロダクト ID が当プロパティに設定されます。
UI	APISerialNo	（有効期限を更新する）シリアル No.（レジストリからの）認証登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行により、登録済みのシリアル No.が当プロパティに設定されます。
UI	APICurrentExpirationDate	（有効期限を更新する）現在の有効期限（レジストリからの）認証登録済み情報の取得メソッド

		(APIGetRegisteredInfoFromRegistryメソッド)の実行により、登録済みの現在の有効期限が当プロパティに設定されます。	
UI	APIUseProxyServer	プロキシサーバーの使用区分	
UI	APIProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時有効
UI	APIProxyServerPort	プロキシサーバーのポート	
UI	APIProxyServerUserName	プロキシサーバーのユーザ名	
UI	APIProxyServerPassword	プロキシサーバーのパスワード	
Code	APIDisabledNICIgnore	処理を高速化するため、無効になっているNIC(Network Interface Card)を無視します。	
Code	APIEncryptionPassword	暗号化時のパスワード	
Code	APIEncryptionSaltString	暗号化時の Salt 文字列	
Code	APIOverwriteModeOfExpirationDateUpdate	新しい有効期限が現在と同じか古い場合の対応を設定します。 True:新しい有効期限が現在と同じか古い場合でも更新します。 False:新しい有効期限が現在と同じか古い場合は更新しません。	
Code	APITrialPeriod	猶予(試用)日数	
Code	APIUseCpuInfo	CPU 情報の使用区分	
Code	APIUseMacAddress	MAC アドレスの使用区分	
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス	
Code	APIWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード	
Code	APIWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名	
Code	APIWebServiceCheckPassword	Web サービス確認パスワード	
Code	APIWebServiceTimeout	Web サービスのタイムアウト	
Code	APIWebServiceURL	Web サービスの URL	
Code	APIWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分	

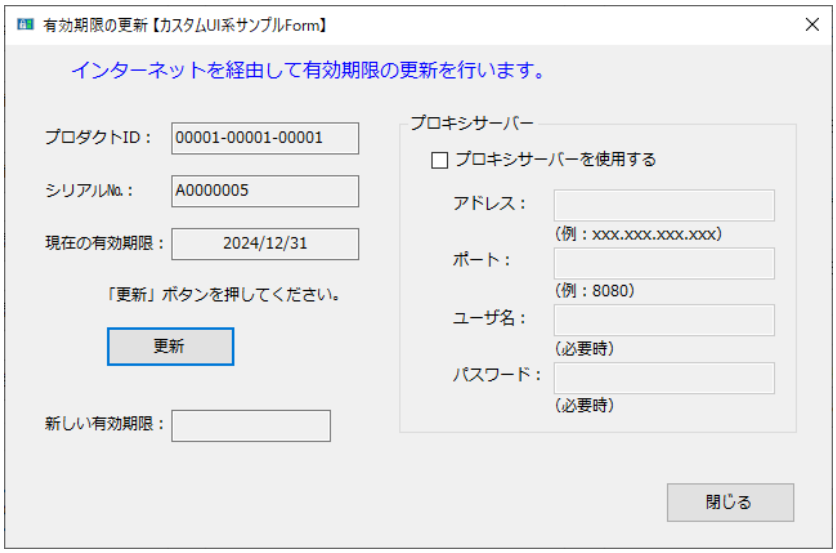
※設定区分:UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム(コード)で設定されるプロパティ

【解説】
「有効期限の更新」を実行します。

プロダクト ID とシリアル No.が有効期限によるライセンスの場合、当メソッドによる処理を利用してエンドユーザに有効期限を更新させることができます。
この処理をエンドユーザに実行してもらう前に、貴社で新しい有効期限の設定をする必要があります。
有効期限の設定は認証管理システムの「認証キー編集(表形式)」処理を利用します。

◆想定 UI 画面

当メソッドの利用時に、想定する貴社作成の UI（ユーザインターフェース）画面は次の通りです。



この画面上の「認証 ID」、「プロダクト ID」、「シリアル No.」は、（レジストリからの）認証登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行で設定される認証 ID プロパティ（APICertificationID プロパティ）、プロダクト ID プロパティ（APIProductID プロパティ）、シリアル No. プロパティ（APISerialNo プロパティ）でそれぞれ取得できます。
この画面上の「現在の有効期限」は、（レジストリからの）認証登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）の実行で設定される現在の有効期限プロパティ（APICurrentExpirationDate プロパティ）で取得できます。

◆処理手順
一連の処理の手順は次の通りです。

[1]（レジストリからの）認証登録済み情報の取得メソッド（APIGetRegisteredInfoFromRegistry メソッド）を実行します。
これにより、プロダクト ID プロパティ、シリアル No.プロパティ、現在の有効期限プロパティに各値が設定されます。

[2]プロダクト ID プロパティ、シリアル No.プロパティ、現在の有効期限プロパティの各値を画面に表示します。

[3]上記の【必須設定プロパティ】一覧にある「設定区分」が「Code」のプロパティに適切な値を設定します。
★これらのプロパティは、一度設定した内容で変更がなければ再設定する必要はありません。

[4] 上記の「想定 UI 画面」の「更新」ボタンに連動させるなどして、当メソッド（APIUpdateOfExpirationDate メソッド）を実行します。

[5]上記の「想定 UI 画面」の「新しい有効期限」に更新後の有効期限を表示します。更新後の有効期限は、処理手順の[1]を再度行うことで取得できます。

なお、エンドユーザに有効期限を更新させる方法として当処理を実行させる他に、エンドユーザに一度認証の解除後、再度認証の登録をしてもらうことでも更新が完了します。
エンドユーザが代理認証を利用している場合で、有効期限によるライセンスを更新する場合はその方法を使います。

APIWriteProxyServerInfoToRegistry メソッド

【機能】
レジストリへプロキシサーバーの情報を書き込みます。

【構文】
<VB.NET>
Public Function **APIWriteProxyServerInfoToRegistry**() As **Boolean**

<C#>
public **bool** **APIWriteProxyServerInfoToRegistry**()

<VC++>
public：**VARIANT_BOOL**
NewtoneNRDvcpp::INRD_APIActivation::APIWriteProxyServerInfoToRegistry()

【引数】
なし

【戻り値】
True：　正常
False：　エラー（未設定や範囲を超えているプロパティがある）

戻り値とは別に、APIErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

設定区分※	プロパティ	設定する内容	
UI	APIUseProxyServer	プロキシサーバーの使用区分	
UI	APIProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時
UI	APIProxyServerPort	プロキシサーバーのポート	
UI	APIProxyServerUserName	プロキシサーバーのユーザ名	
UI	APIProxyServerPassword	プロキシサーバーのパスワード	
Code	APIEncryptionPassword	暗号化時のパスワード	
Code	APIEncryptionSaltString	暗号化時の Salt 文字列	
Code	APIVendorsProductStartRegistryKeyPath	ベンダアプリケーション開始レジストリキーパス	

※設定区分：UI⇒主に「想定 UI 画面」から設定されるプロパティ/Code⇒主にプログラム（コード）で設定されるプロパティ

【解説】
レジストリへプロキシサーバーの情報を書き込みます。

[1]このメソッドの戻り値は成功か失敗（True/False）。

[2]プロキシサーバーのユーザ名とパスワードを暗号化しており暗号化が失敗すると、当メソッドは失敗します。また、レジストリの書き込みで何らかの原因で失敗すると、当メソッドは失敗します。

<ASP.NET 系プロパティ>

プロパティー覧

プロパティ	機能
APIxError 列挙体	エラー内容を表します。
APIxDisabledNICIgnore	処理を高速化するため、無効になっている NIC (Network Interface Card)を無視します。
APIxErrorStatus	ASP.NET 系のメソッドを使用した際のエラーの内容を返す。(取得専用)
APIxExpirationDate	有効期限を設定します。
APIxExternalLinkKey	外部データベースとの「リンク用キー」を設定します。
APIxFreeItem1～5	自由入力項目 1～5 を設定します。
APIxKindOfRandom	シリアル No.をランダムに自動作成する場合の文字種別を設定します。
APIxLicenseCount	ライセンス数を設定します。
APIxNumberingCount	シリアル No.を作成する数を設定します。
APIxProductID	プロダクト ID を設定します。
APIxProxyServerAddress	プロキシサーバーのアドレスを設定します。
APIxProxyServerPassword	プロキシサーバーのパスワードを設定します。
APIxProxyServerPort	プロキシサーバーのポートを設定します。
APIxProxyServerUserName	プロキシサーバーのユーザ名を設定します。
APIxSerialNo	シリアル No.を設定・取得します。
APIxSerialNoString	認証キー情報作成直後のシリアル No.を取得します。(取得専用)
APIxStartNo	シリアル No.を自動的にナンバリングして作成する場合の開始番号を設定します。
APIxStartFixedString	作成するシリアル No.の上位固定文字列を設定します。
APIxStepNo	シリアル No.を自動的にナンバリングして作成する場合の間隔(ステップ)数を設定します。
APIxUseFloatingLicense	フローティングライセンスの使用区分(デフォルト:False)を設定します。
APIxUseProxyServer	プロキシサーバーの使用区分(デフォルト:False)を設定します。
APIxUseExpirationDate	有効期限利用の有無(False:利用しない、True:利用する)を設定します。
APIxWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワードを設定します。
APIxWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名を設定します。
APIxWebServiceCheckPassword	Web サービス確認パスワード(8 文字以上)を設定します。
APIxWebServiceTimeout	Web サービスのタイムアウト(デフォルト:60 秒)を設定します。
APIxWebServiceURL	Web サービスの URL を設定します。
APIxWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用(デフォルト:False)を設定します。

APIxError 列挙体

エラー内容を表します。
public enum APIxError
パブリックメンバ

メンバ名	値	内容	関連するメソッド
None	0	エラーなし	
OutOfMemoryException	11	プログラムの実行を継続するためのメモリが不足している場合にスローされる例外。	
StackOverflowException	12	入れ子になったメソッド呼び出しが多くなりすぎ、実行スタックがオーバーフローした場合にスローされる例外。 このクラスは継承できません。	
UnauthorizedAccessException	13	オペレーティング システムが I/O エラーまたは特定の種類のセキュリティ エラーのためにアクセスを拒否する場合、スローされる例外。	
IoDirectoryNotFoundException	14	ファイルまたはディレクトリの一部が見つからない場合にスローされる例外。	
IoDriveNotFoundException	15	使用できないドライブまたは共有にアクセスしようとするとスローされる例外。	
IoEndOfStreamException	16	ストリームの末尾を越えて読み取ろうとしたときにスローされる例外。	
IoFileLoadException	17	マネージ アセンブリが見つかったが、読み込むことができない場合にスローされる例外。	
IoFileNotFoundException	18	ディスク上に存在しないファイルにアクセスしようとして失敗したときにスローされる例外。	
IoIOException	19	I/O エラーが発生したときにスローされる例外。	
IoPathTooLongException	20	パス名またはファイル名がシステム定義の最大長を超えている場合にスローされる例外。	
BadStrInProductID	101	プロダクト ID に不正な文字が含まれています。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey APIxDeleteActivationKey APIxEditOfExpirationDate
BadStrInSerialNo	102	シリアル No.に不正な文字が含まれています。	APIxCreateOneActivationKey APIxDeleteActiva

			tionKey
FailedEnableNIC	104	NIC の設定に失敗しました。(有効化)	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey APIxDeleteActivationKey APIxEditOfExpirationDate
FailedDisableNIC	105	NIC の設定に失敗しました。(無効化)	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey APIxDeleteActivationKey APIxEditOfExpirationDate
NoMACAdress	106	MAC アドレスが 1 つも取得できませんでした。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey APIxDeleteActivationKey APIxEditOfExpirationDate
AlreadyRegistered	111	既に登録済みのプロダクト ID とシリアル№の組み合わせがあったため登録を中止しました。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey
BadCheckPassword	112	確認パスワードが不正です。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey APIxDeleteActivationKey

			APIxDeleteActivationKey APIxEditOfExpirationDate
BadWaiFile	113	WebServEnv.wai ファイルに問題があります。 認証 Web サービス実行 PC 上で「認証レスキュー！.NET Web 環境設定」（WebAdmin.exe）が行われていない可能性があります。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey APIxDeleteActivationKey APIxEditOfExpirationDate
ExpirationDateIsOutOfRange	116	有効期限が不正です。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey APIxCreateRandomActivationKey APIxEditOfExpirationDate
ExpirationDateIsOldDate	117	入力された「有効期限」が無効（前日以前）です。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey APIxCreateRandomActivationKey APIxEditOfExpirationDate
ProductIDIsEmpty	156	プロダクトIDが入力されていません。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey APIxEditOfExpirationDate
SerialNoIsEmpty	157	シリアル№が入力されていません。	APIxCreateOneActivationKey APIxEditOfExpirationDate

CanNotUseTheString	164	使用できない文字列が含まれています。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey APIxDeleteActivationKey APIxEditOfExpirationDate
PropertyValueNotFound	165	「必須設定プロパティ」に値が設定されていません。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey APIxDeleteActivationKey APIxEditOfExpirationDate
DataBaseNotFound	170	データベースが存在しません。 データベースは「Webインストーラ」を起動し「データベースのインストール」を実行すると作成されます。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey APIxDeleteActivationKey APIxEditOfExpirationDate
CanNotConnectToTheServer	171	何らかの理由でサーバーに接続できませんでした。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey APIxDeleteActivationKey APIxEditOfExpirationDate
DataTableNotFound	172	データテーブルが存在しません。 Web 環境設定で「データテーブル新規作成」の処理を行ってから、再度、この処理を行ってください。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRando

			mActivationKey APIxDeleteActivationKey APIxEditOfExpirationDate
BadStrInFixedSerialNo	173	シリアルNo.の上位固定文字列に不正な文字が含まれています。	APIxCreateNumberingActivationKey APIxCreateRandomActivationKey
InvalidStartNo	174	開始番号は0～99999の範囲で設定してください。	APIxCreateNumberingActivationKey
InvalidStepNo	175	ステップ数は1～99999の範囲で設定してください。	APIxCreateNumberingActivationKey
InvalidNumberingNo	176	ナンバリング数は1～1000の範囲で設定してください。	APIxCreateNumberingActivationKey APIxCreateRandomActivationKey
InvalidLicenseNo	177	ライセンス数は1～100の範囲で設定してください。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey
欠番	178	旧レンタル機能関連	
SerialNoDigitsTooMany	179	上位固定文字列と番号の桁数を合わせると指定したシリアル No.の桁数より多すぎます。	APIxCreateNumberingActivationKey APIxCreateRandomActivationKey
CanNotEnterBlanksInTheProductID	182	プロダクトIDには空白は入力できません。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey APIxEditOfExpirationDate
ProductIDDigitsNotEnough	183	入力されたプロダクト ID の桁数が足りません。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey

			APIxEditOfExpirationDate
AnErrorOccurred	184	何らかのエラーが発生しました。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey APIxDeleteActivationKey APIxDeleteActivationKey APIxEditOfExpirationDate
ProductIDAndSerialNoNotFound	185	そのプロダクトIDとシリアルNo.の組み合わせは登録されていません。	APIxDeleteActivationKey APIxEditOfExpirationDate
SerialNoDigitsNotEnough	186	入力されたシリアル No.の桁数が足りません。	APIxCreateOneActivationKey APIxEditOfExpirationDate
CanNotEnterBlanksInTheSerialNo	187	シリアル No.には空白は入力できません。	APIxCreateOneActivationKey APIxEditOfExpirationDate
BadStrInExternalLinkKey	188	リンク用キーに不正な文字が含まれています。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey
BadStrInFreeItem1	189	自由入力項目 1 に不正な文字が含まれています。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey
BadStrInFreeItem2	190	自由入力項目 2 に不正な文字が含まれています。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey
BadStrInFreeItem3	191	自由入力項目 3 に不正な文字が含まれています。	APIxCreateNumberingActivationKe

			y APIxCreateOneActivationKey APIxCreateRandomActivationKey
BadStrInFreeItem4	192	自由入力項目 4 に不正な文字が含まれています。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey
BadStrInFreeItem5	193	自由入力項目 5 に不正な文字が含まれています。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey
BadKindOfRandomOfSerialNo	194	シリアルNo.をランダムに自動作成するための文字種別が不正です。	APIxCreateRandomActivationKey
BadExpirationDate	195	有効期限が不正です。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey APIxEditOfExpirationDate
ProductIDIsTooLong	196	入力されたプロダクト ID の桁数が多いです。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey APIxEditOfExpirationDate
SerialNoIsTooLong	197	入力されたシリアル No.の桁数が多いです。	APIxCreateOneActivationKey APIxEditOfExpirationDate
BadFormatExpirationDate	198	有効期限の書式が不正です。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey

			APIxEditOfExpirationDate
ExternalLinkKeyIsTooLong	199	入力されたリンク用キーの桁数が多いです。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey
APIxFreeItem1IsTooLong	200	入力された自由入力項目 1 の桁数が多いです。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey
APIxFreeItem2IsTooLong	201	入力された自由入力項目 2 の桁数が多いです。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey
APIxFreeItem3IsTooLong	202	入力された自由入力項目 3 の桁数が多いです。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey
APIxFreeItem4IsTooLong	203	入力された自由入力項目 4 の桁数が多いです。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey
APIxFreeItem5IsTooLong	204	入力された自由入力項目 5 の桁数が多いです。	APIxCreateNumberingActivationKey APIxCreateOneActivationKey APIxCreateRandomActivationKey

APIxDisabledNICIgnore プロパティ

【機能】
処理を高速化するため、無効になっている NIC（Network Interface Card）を無視します。
デフォルト：True。

【構文】
<VB.NET>

```
Public Property APIxDisabledNICIgnore As Boolean
```

<C#>

```
public bool APIxDisabledNICIgnore { set; get; }
```

【解説】
認証レスキュー！の認証 UI ライブラリでは、各処理時に PC の MAC アドレスを最大で 5 個記録します。MAC アドレスは LAN ポートなどのネットワーク機器に固有な物理アドレスです。
このプロパティが False の場合、無効になっている NIC を一度有効にし、情報を取得後に再度 NIC を無効にします。この際に少し時間が掛かります。
このプロパティを True に設定すると、この無効の NIC は無視し、有効の NIC のみ情報を取得して記録します。これにより、情報の取得が高速化します。

APIxErrorStatus プロパティ

【機能】
ASP.NET 系のメソッドを使用した際のエラーの内容を返します。

【構文】
<VB.NET>

```
Public ReadOnly Property APIxErrorStatus As Integer
```

<C#>

```
public int APIxErrorStatus { get; }
```

【解説】
ASP.NET 系のメソッドは戻り値として正常（True）かエラー（False）を返しますが、この APIxErrorStatus プロパティはそのエラーの内容を APIxError 列挙体の参照で返します。この APIxErrorStatus プロパティは取得専用です。

APIxExpirationDate プロパティ

【機能】
有効期限を設定します。

【構文】
<VB.NET>

```
Public Property APIxExpirationDate As String
```

<C#>

```
public string APIxExpirationDate { set; get; }
```

【解説】
有効期限を設定します。
有効期限は、"yyyy/mm/dd"形式で設定してください。

(例)西暦 2020 年 10 月 1 日を設定する場合
<VB> Class1.myAPIActivate.APIxExpirationDate = "2020/10/01"
<C#> Class1.myAPIActivate.APIxExpirationDate = "2020/10/01";

次のメソッドの実行時に当プロパティ(APIxExpirationDate プロパティ)に有効期限を設定します。
・APIxCreateOneActivationKey メソッド
・APIxCreateRandomActivationKey メソッド
・APIxCreateNumberingActivationKey メソッド
・APIxEditOfExpirationDate メソッド

APIxExternalLinkKey プロパティ

【機能】
外部データベースとの「リンク用キー」を設定します。

【構文】
<VB.NET>

```
Public Property APIxExternalLinkKey As String
```

<C#>

```
public string APIxExternalLinkKey { set; get; }
```

【解説】
外部データベースとの「リンク用キー」を設定します。
「リンク用キー」の設定は任意で、認証レスキュー！の運用時の必須設定項目ではありません。
この項目は最大 64 バイトです。

次の認証キー情報を作成するメソッドの実行時に当プロパティ（APIxExternalLinkKey プロパティ）に設定された値が、認証レスキュー！のデータベースの認証キーテーブルの「リンク用キー」に格納されます。

・APIxCreateOneActivationKey メソッド
・APIxCreateRandomActivationKey メソッド
・APIxCreateNumberingActivationKey メソッド

この内、APIxCreateRandomActivationKey メソッドとAPIxCreateNumberingActivationKey メソッドは複数の認証キー（レコード）を作成できますが、すべての認証キー（レコード）に当プロパティ（APIxExternalLinkKey プロパティ）に設定された値が、作成するすべての認証キー（レコード）の「リンク用キー」に格納されます。

APIxFreeItem1～5 プロパティ

【機能】
自由入力項目 1～5 を設定します。

【構文】
<VB.NET>

```
Public Property APIxFreeItem1 As String
Public Property APIxFreeItem2 As String
Public Property APIxFreeItem3 As String
Public Property APIxFreeItem4 As String
Public Property APIxFreeItem5 As String
```

<C#>

```
public string APIxFreeItem1 { set; get; }
public string APIxFreeItem2 { set; get; }
public string APIxFreeItem3 { set; get; }
public string APIxFreeItem4 { set; get; }
public string APIxFreeItem5 { set; get; }
```

【解説】
自由入力項目 1～5 を設定します。
「自由入力項目」の設定は任意で、認証レスキュー！の運用時の必須設定項目ではありません。
この項目は最大 128 バイトです。

次の認証キー情報を作成するメソッドの実行時に当プロパティ（APIxFreeItem1～5 プロパティ）に設定された値が、認証レスキュー！のデータベースの認証キーテーブルの「自由入力項目 1～5」に格納されます。

・APIxCreateOneActivationKey メソッド
・APIxCreateRandomActivationKey メソッド
・APIxCreateNumberingActivationKey メソッド

この内、APIxCreateRandomActivationKey メソッドと APIxCreateNumberingActivationKey メソッドは認証キーテーブルの認証キー（レコード）を同時に複数作成できますが、当プロパティ（APIxFreeItem1～5 プロパティ）に設定された値が、作成するすべての認証キー（レコード）の「自由入力項目 1～5」に格納されます。

APIxKindOfRandom プロパティ

【機能】
シリアル No.をランダムに自動作成する場合の文字種別を設定します。

【構文】
<VB.NET>

```
Public Property APIxKindOfRandom As Integer
```

<C#>

```
public int APIxKindOfRandom { set; get; }
```

【解説】
シリアル No.をランダムに自動作成する場合の文字種別を設定します。
当プロパティ(APIxKindOfRandom プロパティ)に設定する値は次の通りです。
0:数字のみ
1:数字と英字(大文字)
2:数字と英字(小文字)
3:数字と英字(大小文字)

次のメソッドの実行時に当プロパティ(APIxKindOfRandom プロパティ)を設定します。
・APIxCreateRandomActivationKey メソッド

APIxLicenseCount プロパティ

【機能】
ライセンス数を設定します。

【構文】
<VB.NET>

```
Public Property APIxLicenseCount As Integer
```

<C#>

```
public int APIxLicenseCount { set; get; }
```

【解説】
ライセンス数を設定します。

次のメソッドの実行時に当プロパティ(APIxLicenseCount プロパティ)に()内のライセンス数の各値を設定します。
・APIxCreateOneActivationKey メソッド(作成するライセンス数)
・APIxCreateRandomActivationKey メソッド(作成するライセンス数)
・APIxCreateNumberingActivationKey メソッド(作成するライセンス数)

APIxNumberingCount プロパティ

【機能】
シリアル No.を作成する数を設定します。

【構文】
<VB.NET>

```
Public Property APIxNumberingCount As Integer
```

<C#>

```
public int APIxNumberingCount { set; get; }
```

【解説】
シリアル No.を作成する数を設定します。

次のメソッドの実行時に当プロパティ(APIxNumberingCount プロパティ)を設定します。
・APIxCreateRandomActivationKey メソッド
・APIxCreateNumberingActivationKey メソッド

APIxProductID プロパティ

【機能】
プロダクト ID を設定します。

【構文】
<VB.NET>
Public Property **APIxProductID** As **String**

<C#>
public **string** **APIxProductID** { set; get; }

【解説】
プロダクト ID を設定します。

次のメソッドの実行時に当プロパティ(APIxProductID プロパティ)に()内のプロダクト ID の各値を設定します。
・APIxCreateOneActivationKey メソッド(作成するプロダクト ID)
・APIxCreateRandomActivationKey メソッド(作成するプロダクト ID)
・APIxCreateNumberingActivationKey メソッド(作成するプロダクト ID)
・APIxDeleteActivationKey メソッド(削除するプロダクト ID)
・APIxEditOfExpirationDate メソッド(有効期限を変更したいプロダクト ID)

APIxProxyServerAddress プロパティ

【機能】
プロキシサーバーのアドレスを設定します。

【構文】
<VB.NET>
Public Property **APIxProxyServerAddress** As **String**

<C#>
public **string** **APIxProxyServerAddress** { set; get; }

【解説】
プロキシサーバーのアドレスを設定します。
インターネットに接続する際にプロキシサーバーを使用する場合のみ設定が必要です。
プロキシサーバーを使用する場合は、当プロパティを含む次のプロパティに適切な設定が必要です。
・APIxProxyServerAddress プロパティ(プロキシサーバーのアドレス)→**当プロパティ**
・APIxProxyServerPort プロパティ(プロキシサーバーのポート)
・APIxProxyServerUserName プロパティ(プロキシサーバーのユーザ名)
・APIxProxyServerPassword プロパティ(プロキシサーバーのパスワード)

プロキシサーバーを使用する場合、当プロパティの値を利用するメソッドは次の通りです。
・APIxCreateOneActivationKey メソッド(製品 1 本分の認証キーの新規作成)
・APIxCreateRandomActivationKey メソッド(認証キーのシリアル No.をランダムに自動作成)
・APIxCreateNumberingActivationKey メソッド(認証キーのシリアル No.を自動ナンバリング作成)
・APIxDeleteActivationKey メソッド(作成済みの認証キーの削除)
・APIxEditOfExpirationDate メソッド(既存の有効期限を変更))

APIxProxyServerPassword プロパティ

【機能】
プロキシサーバーのパスワードを設定します。

【構文】
<VB.NET>
Public Property **APIxProxyServerPassword** As **String**

<C#>
public **string** **APIxProxyServerPassword** { set; get; }

【解説】
プロキシサーバーのパスワードを設定します。
インターネットに接続する際にプロキシサーバーを使用する場合のみ設定が必要です。
プロキシサーバーを使用する場合は、当プロパティを含む次のプロパティに適切な設定が必要です。
・APIxProxyServerAddress プロパティ(プロキシサーバーのアドレス)
・APIxProxyServerPort プロパティ(プロキシサーバーのポート)
・APIxProxyServerUserName プロパティ(プロキシサーバーのユーザ名)
・APIxProxyServerPassword プロパティ(プロキシサーバーのパスワード)→**当プロパティ**

プロキシサーバーを使用する場合、当プロパティの値を利用するメソッドは次の通りです。
・APIxCreateOneActivationKey メソッド(製品 1 本分の認証キーの新規作成)
・APIxCreateRandomActivationKey メソッド(認証キーのシリアル No.をランダムに自動作成)
・APIxCreateNumberingActivationKey メソッド(認証キーのシリアル No.を自動ナンバリング作成)
・APIxDeleteActivationKey メソッド(作成済みの認証キーの削除)
・APIxEditOfExpirationDate メソッド(既存の有効期限を変更))

APIxProxyServerPort プロパティ

【機能】
プロキシサーバーのポートを設定します。

【構文】
<VB.NET>
Public Property **APIxProxyServerPort** As **String**

<C#>
public **string** **APIxProxyServerPort** { set; get; }

【解説】
プロキシサーバーのポートを設定します。
インターネットに接続する際にプロキシサーバーを使用する場合のみ設定が必要です。
プロキシサーバーを使用する場合は、当プロパティを含む次のプロパティに適切な設定が必要です。
・APIxProxyServerAddress プロパティ(プロキシサーバーのアドレス)
・APIxProxyServerPort プロパティ(プロキシサーバーのポート)→**当プロパティ**
・APIxProxyServerUserName プロパティ(プロキシサーバーのユーザ名)
・APIxProxyServerPassword プロパティ(プロキシサーバーのパスワード)

プロキシサーバーを使用する場合、当プロパティの値を利用するメソッドは次の通りです。
・APIxCreateOneActivationKey メソッド(製品 1 本分の認証キーの新規作成)
・APIxCreateRandomActivationKey メソッド(認証キーのシリアル No.をランダムに自動作成)
・APIxCreateNumberingActivationKey メソッド(認証キーのシリアル No.を自動ナンバリング作成)
・APIxDeleteActivationKey メソッド(作成済みの認証キーの削除)
・APIxEditOfExpirationDate メソッド(既存の有効期限を変更))

APIxProxyServerUserName プロパティ

【機能】
プロキシサーバーのユーザ名を設定します。

【構文】
<VB.NET>
Public Property **APIxProxyServerUserName** As **String**

<C#>
public **string** **APIxProxyServerUserName** { set; get; }

【解説】
プロキシサーバーのユーザ名を設定します。
インターネットに接続する際にプロキシサーバーを使用する場合のみ設定が必要です。
プロキシサーバーを使用する場合は、当プロパティを含む次のプロパティに適切な設定が必要です。
・APIxProxyServerAddress プロパティ(プロキシサーバーのアドレス)
・APIxProxyServerPort プロパティ(プロキシサーバーのポート)
・APIxProxyServerUserName プロパティ(プロキシサーバーのユーザ名)→**当プロパティ**
・APIxProxyServerPassword プロパティ(プロキシサーバーのパスワード)

プロキシサーバーを使用する場合、当プロパティの値を利用するメソッドは次の通りです。
・APIxCreateOneActivationKey メソッド(製品 1 本分の認証キーの新規作成)
・APIxCreateRandomActivationKey メソッド(認証キーのシリアル No.をランダムに自動作成)
・APIxCreateNumberingActivationKey メソッド(認証キーのシリアル No.を自動ナンバリング作成)
・APIxDeleteActivationKey メソッド(作成済みの認証キーの削除)
・APIxEditOfExpirationDate メソッド(既存の有効期限を変更))

APIxSerialNo プロパティ

【機能】
シリアル No.を設定します。

【構文】
<VB.NET>
Public Property **APIxSerialNo** As **String**

<C#>
public **string** **APIxSerialNo** { set; get; }

【解説】
シリアル No.を設定します。

次のメソッドの実行時に当プロパティ(APIxProductID プロパティ)に()内のシリアル No.の各値を設定します。
・APIxCreateOneActivationKey メソッド(作成するシリアル No.)
・APIxDeleteActivationKey メソッド(削除するシリアル No.)
・APIxEditOfExpirationDate メソッド(有効期限を変更したいシリアル No.)

APIxSerialNoString プロパティ

【機能】
認証キー情報作成直後のシリアル No.を取得します。(取得専用)

【構文】
<VB.NET>
Public Property **APIxSerialNoString** As **String**

<C#>
public **string** **APIxSerialNoString** { get; }

【解説】
認証キー情報作成直後のシリアル No.を取得します。(取得専用)
認証キー情報とは、プロダクト ID、シリアル No.、ライセンス数、有効期限利用の有無、有効期限などの総称です。

次のメソッドを実行すると当プロパティ(APIxSerialNoString プロパティ)に新しく生成されたシリアル No の文字列が設定されます。

・APIxCreateOneActivationKey メソッド
・APIxCreateNumberingActivationKey メソッド
・APIxCreateRandomActivationKey メソッド

複数のシリアル No.が生成された場合は、デリミタのカンマ(,)で区切られた文字列が設定されます。
(例)"00000001,00000002,00000003"

上記メソッド実行時のエラーによるシリアル No.の未生成時は、当プロパティ(APIxSerialNo プロパティ)には、""(空文字列)が設定されます。

APIxStartNo プロパティ

【機能】
シリアル No.を自動的にナンバリングして作成する場合の開始番号を設定します。

【構文】
<VB.NET>

```
Public Property APIxStartNo As Integer
```

<C#>

```
public int APIxStartNo { set; get; }
```

【解説】
シリアル No.を自動的にナンバリングして作成する場合の開始番号を設定します。

次のメソッドの実行時に当プロパティ(APIxStartNo プロパティ)を設定します。
・APIxCreateNumberingActivationKey メソッド

APIxStartFixedString プロパティ

【機能】
作成するシリアル No.の上位固定文字列を設定します。

【構文】
<VB.NET>

```
Public Property APIxStartFixedString As String
```

<C#>

```
public string APIxStartFixedString { set; get; }
```

【解説】
作成するシリアル No.の上位固定文字列を設定します。

(例)作成するシリアル No.の上位 4 桁を"0001"で固定する場合
<VB> Class1.myAPIActivate.APIxStartFixedString = "0001"
<C#> Class1.myAPIActivate.APIxStartFixedString = "0001";

次のメソッドの実行時に当プロパティ(APIxStartFixedString プロパティ)に上位固定文字列を設定します。
・APIxCreateRandomActivationKey メソッド
・APIxCreateNumberingActivationKey メソッド

APIxStepNo プロパティ

【機能】
シリアル No.を自動的にナンバリングして作成する場合の間隔（ステップ）数を設定します。

【構文】
<VB.NET>

```
Public Property APIxStepNo As Integer
```

<C#>

```
public int APIxStepNo { set; get; }
```

【解説】
シリアル No.を自動的にナンバリングして作成する場合の間隔（ステップ）数を設定します。

次のメソッドの実行時に当プロパティ（APIxStepNo プロパティ）を設定します。
・APIxCreateNumberingActivationKey メソッド

APIxUseFloatingLicense プロパティ

【機能】
フローティングライセンス使用区分（デフォルト：False）を設定します。

【構文】
<VB.NET>

```
Public Property APIxUseFloatingLicense As Boolean
```

<C#>

```
public bool APIxUseFloatingLicense { set; get; }
```

【解説】
フローティングライセンス使用区分（デフォルト：False）を設定します。

APIxUseProxyServer プロパティ

【機能】
プロキシサーバーの使用区分（デフォルト：False）を設定します。

【構文】
<VB.NET>

```
Public Property APIxUseProxyServer As Boolean
```

<C#>

```
public bool APIxUseProxyServer { set; get; }
```

【解説】
プロキシサーバーの使用区分（デフォルト：False）を設定します。
インターネットに接続する際にプロキシサーバーを使用するかどうかを設定します。
プロキシサーバーを使用する場合は当プロパティに True を、使用しない場合は False をそれぞれ設定します。
プロキシサーバーを使用する場合は、次のプロパティに適切な設定が必要です。
・APIxProxyServerAddress プロパティ（プロキシサーバーのアドレス）
・APIxProxyServerPort プロパティ（プロキシサーバーのポート）
・APIxProxyServerUserName プロパティ（プロキシサーバーのユーザ名）
・APIxProxyServerPassword プロパティ（プロキシサーバーのパスワード）

プロキシサーバーを使用する場合、当プロパティの値を利用するメソッドは次の通りです。
・APIxCreateOneActivationKey メソッド（製品 1 本分の認証キーの新規作成）
・APIxCreateRandomActivationKey メソッド（認証キーのシリアル No.をランダムに自動作成）
・APIxCreateNumberingActivationKey メソッド（認証キーのシリアル No.を自動ナンバリング作成）
・APIxDeleteActivationKey メソッド（作成済みの認証キーの削除）
・APIxEditOfExpirationDate メソッド（既存の有効期限を変更））

APIxUseExpirationDate プロパティ

【機能】
有効期限利用の有無（False：利用しない、True：利用する）を設定します。

【構文】
<VB.NET>
Public Property **APIxUseExpirationDate** As **Boolean**

<C#>
public **bool** **APIxUseExpirationDate** { set; get; }

【解説】
有効期限利用の有無（False：利用しない、True：利用する）を設定します。

次のメソッドの実行時に当プロパティ（APIxUseExpirationDate プロパティ）に有効期限利用の有無を設定します。
・APIxCreateOneActivationKey メソッド
・APIxCreateRandomActivationKey メソッド
・APIxCreateNumberingActivationKey メソッド
・APIxEditOfExpirationDate メソッド

APIxWebServiceBasicAuthenticationPassword プロパティ

【機能】
Web サービス時の基本認証パスワードを設定します。

【構文】
<VB.NET>
Public Property **APIxWebServiceBasicAuthenticationPassword** As **String**

<C#>
public **string** **APIxWebServiceBasicAuthenticationPassword** { set; get; }

【解説】
Web サーバーで基本認証を使用する場合に、そのパスワードを指定します。
基本認証に関しては、APIxWebServiceUseBasicAuthentication プロパティを参照してください。

APIxWebServiceBasicAuthenticationUserName プロパティ

【機能】
Web サービス時の基本認証ユーザ名を設定します。

【構文】
<VB.NET>
Public Property **APIxWebServiceBasicAuthenticationUserName** As **String**

<C#>
public **string** **APIxWebServiceBasicAuthenticationUserName** { set; get; }

【解説】
Web サーバーで基本認証を使用する場合に、そのユーザ名を指定します。
基本認証に関しては、APIxWebServiceUseBasicAuthentication プロパティを参照してください。

APIxWebServiceCheckPassword プロパティ

【機能】
Web サービス確認パスワード(8 文字以上)を設定します。

【構文】
<VB.NET>
Public Property **APIxWebServiceCheckPassword** As **String**

<C#>
public **string** **APIxWebServiceCheckPassword** { set; get; }

【解説】
Web サービスを利用する場合の確認用のパスワードを設定します。ここでは、Web サーバー側設定アプリケーション「認証 Web サービス」の環境設定処理の Web サービスの確認パスワードと同じものを設定します。
確認パスワードは必須項目です、省略はできません。8 文字以上で半角の次の文字が使用できます。
・大文字の英字（A～Z）
・小文字の英字（a～z）
・数字（0～9）
・記号（+-*/!#$%&()=¥@<>?）

APIxWebServiceTimeout プロパティ

【機能】
Web サービスのタイムアウト（デフォルト：60 秒）を設定します。

【構文】
<VB.NET>

```
Public Property APIxWebServiceTimeout As Integer
```

<C#>

```
public int APIxWebServiceTimeout { set; get; }
```

【解説】
Web サービスに接続して応答を待つ最大時間を秒単位で指定します。初期値は 60 秒です。

APIxWebServiceURL プロパティ

【機能】
Web サービスの URL を設定します。

【構文】
<VB.NET>
Public Property **APIxWebServiceURL** As **String**

<C#>
public **string** **APIxWebServiceURL** { set; get; }

【解説】
認証に関するシステムを Web サービスとして提供する Web サーバーの URL を指定します。
以下に例を示します。

自 PC のローカルホストの Web サーバー(IIS)にアクセスする例:
"http://localhost/NRDWebService/Service.asmx"
(注)この例は実際の運用ではありえません。貴社のアプリケーションで認証 UI ライブラリ(DLL)を使用した開発時に、テスト用に使用される URL です。

自社 Web サーバー(IIS)にアクセスさせる例:
"http://www.newtone.co.jp/NRDWebService/Service.asmx"

クラウドサービス Microsoft Azure の Web アプリ(App Service)に配置した Web サービスを利用する例:
"http://newtonecojp.azurewebsites.net/Service.asmx"

APIxWebServiceUseBasicAuthentication プロパティ

【機能】
Web サービス時の基本認証の使用（デフォルト：False）を設定します。

【構文】
<VB.NET>
Public Property **APIxWebServiceUseBasicAuthentication** As **Boolean**

<C#>
public **bool** **APIxWebServiceUseBasicAuthentication** { set; get; }

【解説】
Web サーバーで基本認証を使用する場合は、当プロパティを True にします。
初期値は、False（基本認証を使用しない）です。

当プロパティは、Web サーバー（IIS）側で特定のアカウント（ユーザ名とパスワード）でアクセスできるフォルダにこの Web サービスが配置してある場合に使用できるセキュリティ設定です。

Web サーバーで基本認証を使用する一般的な手順は次の通りです。
1.サーバーPC 上でユーザを作成。
　この際のユーザ名とパスワードがそのまま基本認証に使われます。
2.基本認証フォルダのセキュリティ設定
　フォルダのプロパティを開き、セキュリティタブで上記 1 のユーザ名を　追加し、「読み取り」権限を付与します。
3.IIS でのセキュリティ設定
　IIS　で該当フォルダに対し「認証」の設定で「匿名認証」を無効にして　「基本認証」を有効にします。

なお、Web サーバーでの基本認証の詳細につきましてはマイクロソフト社の関連ドキュメントをご覧ください。

<ASP.NET 系メソッド>

メソッド一覧

メソッド	機能
APIxCreateNumberingActivationKey	製品の認証キー情報を指定した数だけシリアル No.を自動的にナンバリングして作成します。
APIxCreateOneActivationKey	製品 1 本分の認証キー情報を新規に作成します。
APIxCreateRandomActivationKey	製品の認証キー情報を指定した数だけ、シリアル No.をランダムに自動作成します。
APIxDeleteActivationKey	作成済みの認証キー情報を削除します。
APIxEditOfExpirationDate	既存の認証キー情報内の有効期限を変更します。

APIxCreateNumberingActivationKey メソッド

【機能】
製品の認証キー情報を指定した数だけシリアル No.を自動的にナンバリングして作成します。

【構文】
<VB.NET>
Public Function **APIxCreateNumberingActivationKey** () As **Integer**

<C#>
public **int** **APIxCreateNumberingActivationKey** ()

【引数】
なし

【戻り値】
生成されたシリアル No.の数。
エラー時は 0 が設定され、戻り値とは別に、APIxErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

また、APIxSerialNoString プロパティに新しく生成されたシリアル No.をデリミタのカンマ(,)で区切った文字列が設定されます。シリアル No.の未生成時は、""(空文字列)が設定されます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

プロパティ	機能	
APIxUseProxyServer	プロキシサーバーの使用区分	
APIxProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時
APIxProxyServerPort	プロキシサーバーのポート	
APIxProxyServerUserName	プロキシサーバーのユーザ名	
APIxProxyServerPassword	プロキシサーバーのパスワード	
APIxWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード(基本認証使用時)	
APIxWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名(基本認証使用時)	
APIxWebServiceCheckPassword	Web サービス確認パスワード	
APIxWebServiceTimeout	Web サービスのタイムアウト	
APIxWebServiceURL	Web サービスの URL	
APIxWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分	
APIxProductID	プロダクト ID	
APIxLicenseCount	ライセンス数	
APIxUseExpirationDate	有効期限利用の有無(False:利用しない、True:利用する)	
APIxExpirationDate	有効期限	
APIxStartFixedString	作成するシリアル No.の上位固定文字列	

APIxStartNo	作成するシリアル No.開始番号
APIxStepNo	作成するシリアル No.間隔（ステップ）数
APIxNumberingCount	作成するシリアル No.の数
APIxExternalLinkKey	外部データベースとのリンク用キー項目を設定
APIxFreeItem1～5	自由入力項目 1～5 を設定

【解説】

このメソッドは、認証キー情報を指定した数だけシリアル No.を自動的にナンバリングして作成します。

認証キー情報とは、プロダクト ID、シリアル No.、ライセンス数、有効期限利用の有無、有効期限などの総称です。

認証管理システム（InsideSystem.exe）の「認証キー作成（自動ナンバリング）」処理と同等の機能を提供します。

◆「認証キー作成（自動ナンバリング）」処理の画面例（参考）

◆処理手順

一連の処理の手順は次の通りです。

[1]上記の【必須設定プロパティ】一覧にある各プロパティに適切な値を設定します。

[2]当メソッド（APIxCreateNumberingActivationKey メソッド）を実行します。

APIxCreateOneActivationKey メソッド

【機能】
製品 1 本分の認証キー情報を新規に作成します。

【構文】
<VB.NET>
Public Function **APIxCreateOneActivationKey** () As **Integer**

<C#>
public **int** **APIxCreateOneActivationKey**()

【引数】
なし

【戻り値】
生成されたシリアル No.の数(当メソッドの場合は 1)。
エラー時は 0 が設定され、戻り値とは別に、APIxErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。
また、APIxSerialNoString プロパティには新しく生成されたシリアル No.(当メソッドの場合は、APIxSerialNo プロパティに指定した値)が設定されます。エラーによるシリアル No.の未生成時は、この APIxSerialNoString プロパティには、""(空文字列)が設定されます。

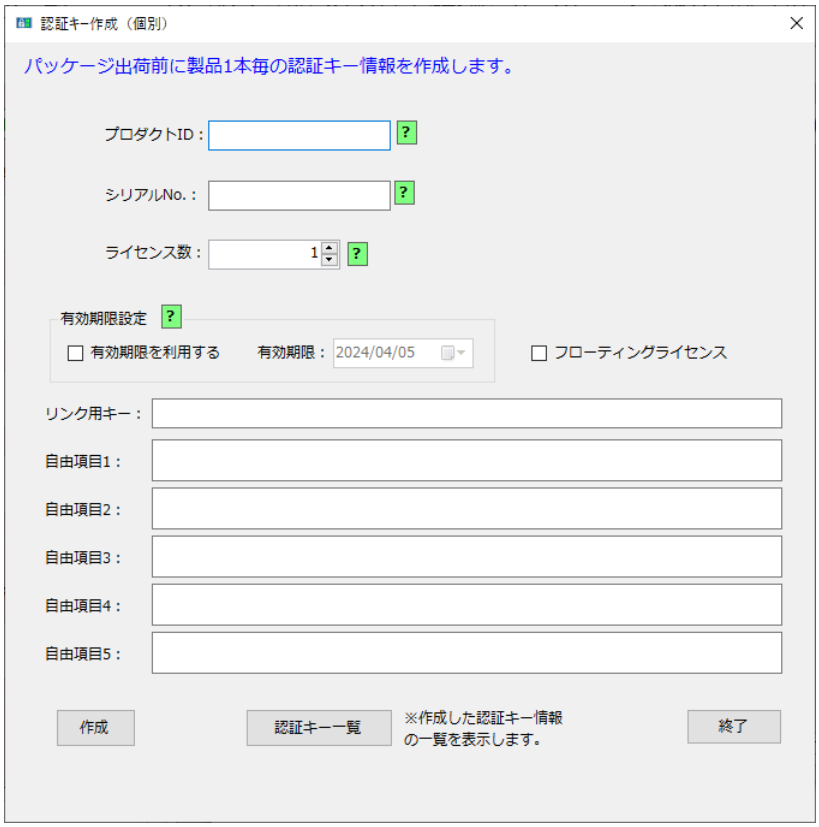
【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

プロパティ	機能	
APIxUseProxyServer	プロキシサーバーの使用区分	
APIxProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時
APIxProxyServerPort	プロキシサーバーのポート	
APIxProxyServerUserName	プロキシサーバーのユーザ名	
APIxProxyServerPassword	プロキシサーバーのパスワード	
APIxWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード(基本認証使用時)	
APIxWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名(基本認証使用時)	
APIxWebServiceCheckPassword	Web サービス確認パスワード	
APIxWebServiceTimeout	Web サービスのタイムアウト	
APIxWebServiceURL	Web サービスの URL	
APIxWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分	
APIxProductID	(作成する)プロダクト ID	
APIxSerialNo	(作成する)シリアル No.	
APIxLicenseCount	(作成する)ライセンス数	
APIxUseExpirationDate	有効期限利用の有無(False:利用しない、True:利用する)	
APIxExpirationDate	有効期限	

APIxExternalLinkKey	外部データベースとのリンク用キー項目を設定
APIxFreeItem1～5	自由入力項目 1～5 を設定

【解説】
このメソッドは、製品 1 本分の認証キー情報を新規に作成します。
認証キー情報とは、プロダクト ID、シリアル No.、ライセンス数、有効期限利用の有無、有効期限などの総称です。
認証管理システム（InsideSystem.exe）の「認証キー作成（個別）」処理と同等の機能を提供します。

◆「認証キー作成（個別）」処理の画面例（参考）



◆処理手順
一連の処理の手順は次の通りです。

[1]上記の【必須設定プロパティ】一覧にある各プロパティに適切な値を設定します。

[2]当メソッド（APIxCreateOneActivationKey メソッド）を実行します。

APIxCreateRandomActivationKey メソッド

【機能】
製品の認証キー情報を指定した数だけ、シリアル No.をランダムに自動作成します。

【構文】
<VB.NET>
Public Function **APIxCreateRandomActivationKey** () As **Integer**

<C#>
public **int** **APIxCreateRandomActivationKey** ()

【引数】
なし

【戻り値】
生成されたシリアル No.の数。
エラー時は 0 が設定され、戻り値とは別に、APIxErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。
また、APIxSerialNoString プロパティに新しく生成されたシリアル No.をデリミタのカンマ(,)で区切った文字列が設定されます。シリアル No.の未生成時は、""(空文字列)が設定されます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

プロパティ	機能	
APIxUseProxyServer	プロキシサーバーの使用区分	
APIxProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時
APIxProxyServerPort	プロキシサーバーのポート	
APIxProxyServerUserName	プロキシサーバーのユーザ名	
APIxProxyServerPassword	プロキシサーバーのパスワード	
APIxWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード(基本認証使用時)	
APIxWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名(基本認証使用時)	
APIxWebServiceCheckPassword	Web サービス確認パスワード	
APIxWebServiceTimeout	Web サービスのタイムアウト	
APIxWebServiceURL	Web サービスの URL	
APIxWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分	
APIxProductID	プロダクト ID	
APIxLicenseCount	ライセンス数	
APIxUseExpirationDate	有効期限利用の有無(False:利用しない、True:利用する)	
APIxExpirationDate	有効期限	
APIxStartFixedString	シリアル No.の上位固定文字列	
APIxKindOfRandom	シリアル No.のランダム文字種の指定	

	0:数字のみ、1:数字と英字(大文字)、2:数字と英字(小文字)、3:数字と英字(大小文字)
APIxNumberingCount	作成するシリアル No.の数
APIxExternalLinkKey	外部データベースとのリンク用キー項目を設定
APIxFreeItem1～5	自由入力項目 1～5 を設定

【解説】
このメソッドは、認証キー情報を指定した数だけ、シリアル No.をランダムに自動作成します。
認証キー情報とは、プロダクト ID、シリアル No.、ライセンス数、有効期限利用の有無、有効期限などの総称です。
認証管理システム(InsideSystem.exe)の「認証キー作成(ランダム生成)」処理と同等の機能を提供します。

◆「認証キー作成(ランダム生成)」処理の画面例(参考)

◆処理手順
一連の処理の手順は次の通りです。

[1]上記の【必須設定プロパティ】一覧にある各プロパティに適切な値を設定します。

[2]当メソッド(APIxCreateRandomActivationKey メソッド)を実行します。

APIxDeleteActivationKey メソッド

【機能】
作成済みの認証キー情報を削除します。

【構文】
<VB.NET>
Public Function **APIxDeleteActivationKey**() As **Integer**

<C#>
public **int** **APIxDeleteActivationKey**()

【引数】
なし

【戻り値】
削除に成功した場合は 1 が設定されます。
エラー時は 0 が設定され、戻り値とは別に、APIxErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

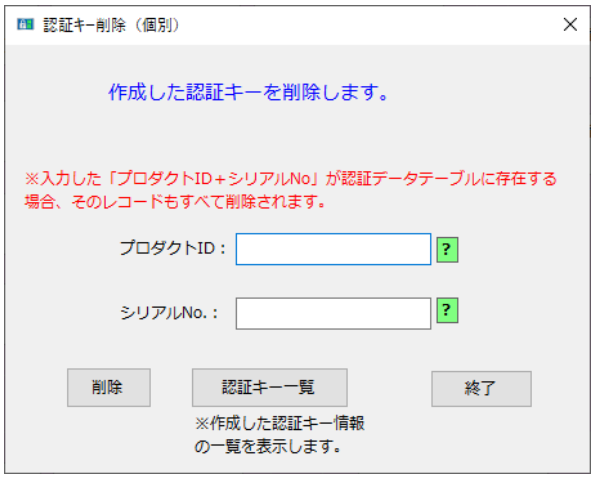
【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

プロパティ	機能	
APIxUseProxyServer	プロキシサーバーの使用区分	
APIxProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時
APIxProxyServerPort	プロキシサーバーのポート	
APIxProxyServerUserName	プロキシサーバーのユーザ名	
APIxProxyServerPassword	プロキシサーバーのパスワード	
APIxWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード（基本認証使用時）	
APIxWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名（基本認証使用時）	
APIxWebServiceCheckPassword	Web サービス確認パスワード	
APIxWebServiceTimeout	Web サービスのタイムアウト	
APIxWebServiceURL	Web サービスの URL	
APIxWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分	
APIxProductID	（削除する）プロダクト ID	
APIxSerialNo	（削除する）シリアル No.	

【解説】
このメソッドは、作成済みの認証キー情報を削除します。
認証キー情報とは、プロダクト ID、シリアル No.、ライセンス数、有効期限利用の有無、有効期限などの総称です。
認証管理システム（InsideSystem.exe）の「認証キー削除（個別）」処理と同等の機能を提供します。

指定した認証キーテーブルの「プロダクト ID」+「シリアル No.」が認証データテーブルに存在する場合は、そのレコードもすべて削除されます。

◆「認証キー作成（個別）」処理の画面例（参考）



◆処理手順
一連の処理の手順は次の通りです。

[1]上記の【必須設定プロパティ】一覧にある各プロパティに適切な値を設定します。

[2]当メソッド（APIxDeleteActivationKey メソッド）を実行します。

APIxEditOfExpirationDate メソッド

【機能】
既存の認証キー情報内の有効期限を変更します。

【構文】
<VB.NET>
Public Function **APIxEditOfExpirationDate**() As **Integer**

<C#>
public **int** **APIxEditOfExpirationDate**()

【引数】
なし

【戻り値】
有効期限の変更に成功した場合は 1 が設定されます。
エラー時は 0 が設定され、戻り値とは別に、APIxErrorStatus プロパティに当メソッド実行後のエラー内容がセットされます。

【必須設定プロパティ】
このメソッドの利用には、あらかじめ次のプロパティに適切な値が設定されている必要があります。

プロパティ	機能	
APIxUseProxyServer	プロキシサーバーの使用区分	
APIxProxyServerAddress	プロキシサーバーのアドレス	プロキシサーバー使用時
APIxProxyServerPort	プロキシサーバーのポート	
APIxProxyServerUserName	プロキシサーバーのユーザ名	
APIxProxyServerPassword	プロキシサーバーのパスワード	
APIxWebServiceBasicAuthenticationPassword	Web サービス時の基本認証パスワード（基本認証使用時）	
APIxWebServiceBasicAuthenticationUserName	Web サービス時の基本認証ユーザ名（基本認証使用時）	
APIxWebServiceCheckPassword	Web サービス確認パスワード	
APIxWebServiceTimeout	Web サービスのタイムアウト	
APIxWebServiceURL	Web サービスの URL	
APIxWebServiceUseBasicAuthentication	Web サービス時の基本認証の使用区分	
APIxProductID	（有効期限を変更する）プロダクト ID	
APIxSerialNo	（有効期限を変更する）シリアル No.	
APIxUseExpirationDate	有効期限利用の有無（False：利用しない、True：利用する）	
APIxExpirationDate	（新しい）有効期限	

【解説】
このメソッドは、既存の認証キー情報の内の有効期限を変更します。
認証キー情報とは、プロダクト ID、シリアル No.、ライセンス数、有効期限利用の有無、有効期限な

どの総称です。
認証管理システム（InsideSystem.exe）の「認証キー編集（表形式）」処理の一部と同等の機能を提供します。

◆「認証キー編集（表形式）」処理の画面例（参考）

◆処理手順
一連の処理の手順は次の通りです。

[1]上記の【必須設定プロパティ】一覧にある各プロパティに適切な値を設定します。

[2]当メソッド（APIxEditOfExpirationDate メソッド）を実行します。

代理認証機能について

次の 8 つのメソッド

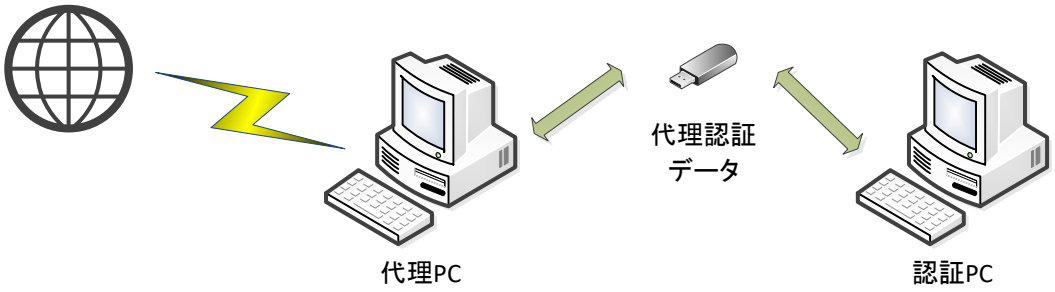
1.「代理認証登録/準備」処理の呼び出し/ProxyActivateRegisterPrepare
2.「代理認証登録/確定」処理の呼び出し/ProxyActivateRegisterFix
3.「代理認証解除/準備」処理の呼び出し/ProxyActivateRemovePrepare
4.「代理有効期限更新/準備」処理の呼び出し/PreparationOfProxyUpdateOfExpirationDate
5.「代理有効期限更新/確定」処理の呼び出し/DeterminationOfProxyUpdateOfExpirationDate
6.「代理認証登録/実行」処理の呼び出し(代理 PC で利用)/ProxyActivateRegisterExecute
7.「代理認証解除/実行」処理の呼び出し(代理 PC で利用)/ProxyActivateRemoveExecute
8.「代理有効期限/取得」処理の呼び出し(代理 PC で利用)/GetProxyUpdateOfExpirationDate

は代理で認証する機能です。

これらの機能は、実際にライセンスの認証登録をして貴社のアプリケーションを動作させたい**エンドユーザの PC がインターネット経由での認証ができない状態で、かつ貴社も電話による認証を受け付ける体制をもたない場合**の手段として利用できます。
これは、エンドユーザにとっては電話で認証する場合と違い貴社の休業日でも認証でき、貴社にとっては電話受付による人材や体制のためのコストが不要になる、という双方にとってのメリットがあります。

代理認証は、エンドユーザがこの機能を使ってインターネット接続できる他の PC で代わりに認証をして、その認証情報を貴社のアプリケーションを動作させたい PC に保存する、というものです。

次図のような流れになります。



この代理認証機能を使用しないのであれば以降の記載は必要ありませんので読み飛ばしてください。

◆エンドユーザが行う処理

＜代理認証登録＞

・最終的に認証登録したい PC で、「代理認証**登録**/準備」処理を行い、代理認証準備データを USB メモリなどに出力します。

・代理に使う PC で、代理認証準備データを読込み「代理認証**登録**/実行」処理をインターネット経由で行い、代理認証実行データを USB メモリなどに出力します。

・最終的に認証登録したい PC で、代理認証実行データを読込み「代理認証**登録**/確定」処理を行い認証**登録**が確定します。

＜代理認証解除＞

・最終的に認証解除したい PC で、「代理認証**解除**/準備」処理を行い、代理認証準備データを USB メモリなどに出力します。この時点で認証 PC の認証は解除されます。

・代理に使う PC で、代理認証準備データを読込み「代理認証**解除**/実行」処理をインターネット経由で行い、認証**解除**が確定します。

＜代理有効期限更新＞

・最終的に認証登録済の有効期限を更新したい PC で、「代理**有効期限更新**/準備」処理を行い、代理有効期限更新準備データを USB メモリなどに出力します。

・代理に使う PC で、代理有効期限更新準備データを読込み「代理**有効期限**/取得」処理をインターネット経由で行い、代理有効期限取得データを USB メモリなどに出力します。

・最終的に認証登録済の有効期限を更新したい PC で、代理有効期限取得データを読込み「代理**有効期限更新**/確定」処理を行い有効期限の**更新**が確定します。

◆貴社が行う作業

上記の「エンドユーザが行う処理」をエンドユーザに操作させるために認証 UI ライブラリ（DLL）を利用してアプリケーションに代理機能を組み込む必要があります。

エンドユーザに対し、最終的に認証したい PC で作業させる処理の例は付属の開発言語別のサンプルプロジェクトの **PackageApp** 内に、代理 PC で作業させる処理の例は付属の開発言語別のサンプルプロジェクトの **ProxyApp** にそれぞれあります。

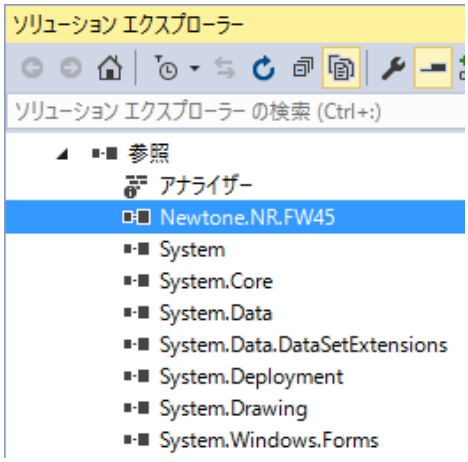
したがって、貴社は最終的に認証したい PC 用のアプリケーションの他に代理 PC で代理処理をするための（**ProxyApp** のような）アプリケーションを作成し配布する必要があります。

■認証 UI ライブラリの参照

Visual Studio 2015（Visual Baisc 2015/C# 2015）の場合

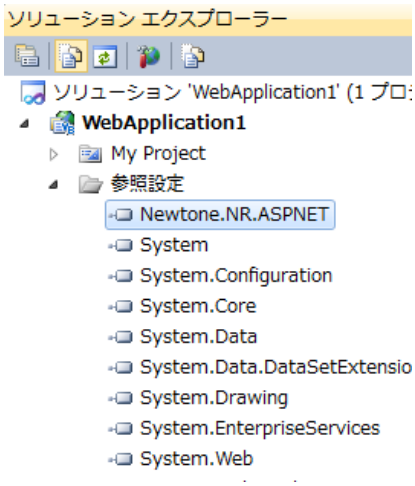
認証レスキュー！の認証 UI ライブラリ（DLL）を使用するプロジェクトの「参照の追加」で Newtone.NR.FW45.dll を参照します。

<Newtone.NR.FW45.dll を参照した後の参照設定の様子（Visual Basic 2015）>



また、ASP.NET 系の DLL を使用する場合は、プロジェクトの「参照の追加」で Newtone.NR.ASPNET.dll を参照します。

<Newtone.NR.ASPNET.dll を参照した後の参照設定の様子（Visual Basic 2010）>



Newtone.NR.FW45.dll、Newtone.NR.NET6.dlll および Newtone.NR.ASPNET.dll は、インストール先がデフォルトの場合、次のフォルダ内にあります。

<32bitOS の場合> C:¥Program Files¥Newtone¥NRD¥NRDDLL¥NRDLL
<64bitOS の場合> C:¥Program Files (x86)¥Newtone¥NRD¥NRDDLL¥NRDLL

Visual C++の場合

認証 UI ライブラリ(DLL)を Visual C++(以降 VC++)から利用する方法は次の通りです。

次の 3 つのファイルが必要です。
Newtone.NR.FW45.dll(.NET Framework 4.5～4.8 アプリケーション用)
NewtoneNRDvcpp.dll(Visual C++で作成したアプリケーションの配布時)
NewtoneNRDvcpp.tlb(Visual C++で作成したアプリケーションの配布時)

これらは、認証レスキュー！.NET の「認証 UI ライブラリ」をインストールしてインストール先がデフォルトの場合、次のフォルダ内にあります。

<32bitOS の場合> C:¥Program Files¥Newtone¥NRD¥NRDDLL¥NRDLL
<64bitOS の場合> C:¥Program Files (x86)¥Newtone¥NRD¥NRDDLL¥NRDLL

手順 1:DLL の登録

次の通りコマンドプロンプトなどで DLL の登録(解除)を行います。
<DllPath>は、上記のフォルダパスを示します。

```
--------------
DLL の登録
--------------

C:¥WINDOWS¥Microsoft.NET¥Framework(または Framework64)¥v4.0.30319¥regasm.exe "<DllPath>NewtoneNRDvcpp.dll" /tlb:"<DllPath>NewtoneNRDvcpp.tlb" /codebase /nologo

(例:)<32bitOS の場合>
C:¥WINDOWS¥Microsoft.NET¥Framework¥v4.0.30319¥regasm.exe "C:¥Program Files¥Newtone¥NRD¥NRDDLL¥NRDLL¥NewtoneNRDvcpp.dll" /tlb:"C:¥Program Files¥Newtone¥NRD¥NRDDLL¥NRDLL¥NewtoneNRDvcpp.tlb" /codebase /nologo

(例:)<64bitOS 上で 32bit(Win32,x86)EXE を実行する場合>
C:¥WINDOWS¥Microsoft.NET¥Framework¥v4.0.30319¥regasm.exe "C:¥Program Files (x86)¥Newtone¥NRD¥NRDDLL¥NRDLL¥NewtoneNRDvcpp.dll" /tlb:"C:¥Program Files (x86)¥Newtone¥NRD¥NRDDLL¥NRDLL¥NewtoneNRDvcpp.tlb" /codebase /nologo

(例:)<64bitOS 上で 64bit(Win64,x64)EXE を実行する場合>
C:¥WINDOWS¥Microsoft.NET¥Framework64¥v4.0.30319¥regasm.exe "C:¥Program Files (x86)¥Newtone¥NRD¥NRDDLL¥NRDLL¥NewtoneNRDvcpp.dll" /tlb:"C:¥Program Files (x86)¥Newtone¥NRD¥NRDDLL¥NRDLL¥NewtoneNRDvcpp.tlb" /codebase /nologo
```

また、登録した DLL を解除する場合は次の通りです。

```
--------------
DLL の解除
--------------

C:¥WINDOWS¥Microsoft.NET¥Framework(または Framework64)¥v4.0.30319¥regasm.exe "<DllPath>NewtoneNRDvcpp.dll" /tlb:"<DllPath>NewtoneNRDvcpp.tlb" /u /nologo

(例:)<32bitOS の場合>
```

C:¥WINDOWS¥Microsoft.NET¥Framework¥v4.0.30319¥regasm.exe ″C:¥Program Files¥Newtone¥NRD¥NRDDLL¥NRDLL¥NewtoneNRDvcpp.dll″ /tlb:″C:¥Program Files¥Newtone¥NRD¥NRDDLL¥NRDLL¥NewtoneNRDvcpp.tlb″ /u /nologo

（例：）<64bitOS 上で 32bit(Win32,x86)EXE の実行を解除するの場合>
C:¥WINDOWS¥Microsoft.NET¥Framework¥v4.0.30319¥regasm.exe ″C:¥Program Files (x86)¥Newtone¥NRD¥NRDDLL¥NRDLL¥NewtoneNRDvcpp.dll″ /tlb:″C:¥Program Files (x86)¥Newtone¥NRD¥NRDDLL¥NRDLL¥NewtoneNRDvcpp.tlb″ /u /nologo

（例：）<64bitOS 上で 64bit(Win64,x64)EXE の実行を解除するの場合>
C:¥WINDOWS¥Microsoft.NET¥Framework64¥v4.0.30319¥regasm.exe ″C:¥Program Files (x86)¥Newtone¥NRD¥NRDDLL¥NRDLL¥NewtoneNRDvcpp.dll″ /tlb:″C:¥Program Files (x86)¥Newtone¥NRD¥NRDDLL¥NRDLL¥NewtoneNRDvcpp.tlb″ /u /nologo

手順 2：VC++用のタイプライブラリの Path をアプリケーション内で指定する

VC++アプリケーションから認証レスキュー！.NET の VC++用の DLL を利用するためには、そのアプリケーション内で VC++用のタイプライブラリ（NewtoneNRDvcpp.tlb）の Path を指定する必要があります。
認証レスキュー！.NET の VC++用のサンプルプロジェクトではタイプライブラリ（NewtoneNRDvcpp.tlb）へのパスは次のファイル内に記述してありますのでご確認ください。

既定 UI 系サンプルプロジェクト PackageApp¥SamplePackageApp.FW45 → SamplePackageApp.FW45.h
既定 UI 系サンプルプロジェクト ProxyApp¥SampleProxyApp.FW45 → SampleProxyApp.FW45.h

カスタム UI 系サンプル PackageApp¥SamplePackageAppAPI.FW45 → PackageApp.h
カスタム UI 系サンプル ProxyApp¥SampleProxyAppAPI.FW45 → ProxyApp.h

（コード例）
```
// 認証レスキュー！.NET の VC++用の tlb をインポート
#import ″..¥¥..¥¥..¥¥..¥¥NRDLL¥¥NewtoneNRDvcpp.tlb″ //raw_interfaces_only
//※当サンプルでは、raw_interfaces_only 属性は付与しない
```

■認証 UI ライブラリ（DLL）を利用したコーディング <既定 UI 系の場合>

Visual Baisc 2015 の場合

付属のサンプルプロジェクト(NRDDLL¥SampleProject¥VisualBasic¥PackageApp¥SamplePackageApp.FW45)の例で説明します。

最初に、Newtone.NR.Activationクラスのインスタンスを作成します。
次の例では、2つのFormで同一のインスタンスで使用するためクラスClass1内でPublic Sharedで宣言しています。

```
Public Class Class1

    Public Shared myActivate As Newtone.NR.FW45.NRD_Activation = New Newtone.NR.FW45.NRD_Activation()

End Class
```

次に DLL のプロパティを設定します。
これらのプロパティは通常、最初に 1 回だけ固定的に設定するコードを記述します。

```
Public Class Form1

    Private Sub Form1_Load(sender As System.Object, e As System.EventArgs) Handles MyBase.Load

        '------------------------------------------------------------------
        'Newtone.NR.FW45.dllのプロパティの設定
        '【重要】DLLの以下のプロパティは必ず適切なものを設定してください。
        '------------------------------------------------------------------

        'ベンダ製品スタート開始レジストリキーパス（※最終的には必ず貴社のものに変更してください）
        Class1.myActivate.VendorsProductStartRegistryKeyPath =
"Software¥Newtone¥NinshoRescue¥NRD¥SampleProject"
        '電話で認証時の電話番号（※最終的には必ず貴社のものに変更してください）
        Class1.myActivate.TelephoneNumber = "012-345-6789"
        '暗号化時のパスワード（※最終的には必ず貴社のものに変更してください）
        Class1.myActivate.EncryptionPassword = "12345678ABCDEFGH"
        '暗号化時のSalt文字列(8バイト以上)（※最終的には必ず貴社のものに変更してください）
        Class1.myActivate.EncryptionSaltString = "認証レスキュー！"
        '猶予日数（デフォルト：0日、設定可能範囲：1～365日）（※最終的には必ず貴社のものに変更してください）
        Class1.myActivate.TrialPeriod = 0
        '猶予期間の名称（デフォルト："猶予（試用）期間"）
        Class1.myActivate.TrialPeriodName = "猶予（試用）"
        'MACアドレスの使用（デフォルト：True）「代理認証」利用時はMACアドレス必須
        Class1.myActivate.UseMacAddress = True
        'CPU情報の使用（デフォルト：True）
        Class1.myActivate.UseCpuInfo = True
        'WebサービスのURL（※最終的には必ず貴社のものに変更してください）
        Class1.myActivate.WebServiceURL = "http://localhost/NRDWebService/Service.asmx"
        'Webサービス時の基本認証の使用（デフォルト：False）
        Class1.myActivate.WebServiceUseBasicAuthentication = False
        'Webサービス時の基本認証ユーザ名
        Class1.myActivate.WebServiceBasicAuthenticationUserName = ""
        'Webサービス時の基本認証パスワード
        Class1.myActivate.WebServiceBasicAuthenticationPassword = ""
```

```
        'Webサービス確認パスワード（※最終的には必ず貴社のものに変更してください）
        Class1.myActivate.WebServiceCheckPassword = "ABcd1234"
        'プロダクトIDの桁数（デフォルト：17）（※最終的には必ず貴社のものに変更してください）
        Class1.myActivate.ProductIdNumberOfDigits = 17
        'シリアルNo.の桁数（デフォルト：8）（※最終的には必ず貴社のものに変更してください）
        Class1.myActivate.SerialNoNumberOfDigits = 8
        'Webサービスのタイムアウト（デフォルト：60秒）
        Class1.myActivate.WebServiceTimeout = 60
        '認証登録時の設定プロダクトID（デフォルト：空文字列）
        Class1.myActivate.SetProductID = ""
        '認証登録時の設定シリアルNo.（デフォルト：空文字列）
        Class1.myActivate.SetSerialNo = ""
        '無効になっているNICを無視する（高速化を図る）（デフォルト：True）
        Class1.myActivate.DisabledNICIgnore = True

        CertificationStatus() '認証状況の確認

    End Sub

・・・

End Class
```

後は、必要に応じてDLLのメソッドを呼び出します。

次の例では、ActivateStatusCheck()メソッドで「認証状態確認」を行います。

```
Public Class Form1

    Private Sub CertificationStatus()

        '----------------------------------------
        '認証状況の確認
        '----------------------------------------

        'Button1、Button2、Button3 を含む GroupBox1 の Enabled プロパティを False にして無効とする。
        GroupBox1.Enabled = False

        Dim ret As Integer
        Dim stat As String '表示メッセージ

        ret = Class1.myActivate.ActivateStatusCheck()

        '認証状況確認
        Select Case ret
            Case 0 '期限切れ（猶予有効時）
                stat =
                "********************************" + vbCr +
                "これは、サンプルメッセージです。" + vbCr +
                "********************************" + vbCr +
                "[ID:" + ret.ToString + "] " + Class1.myActivate.TrialPeriodName + "期限が切れました。" + vbCr + "メニューは実行できません。" + vbCr + "「ライセンス管理」からライセンスの認証登録を行ってください。"
                MessageBox.Show(stat, "認証確認", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)

                Label2.Text =
                    "現在、「猶予（試用）期限切れ」のため「処理メニュー」は" + vbCr +
```

```
                    "グレーアウトになっています。"
                Case 1 To 365 '猶予日数有
                    stat =
                    "**********" + vbCr +
                    "これは、サンプルメッセージです。" + vbCr +
                    "**********" + vbCr +
                    "[ID:" + ret.ToString + "] " + Class1.myActivate.TrialPeriodName + "期間残日数は
" + ret.ToString + "日です。" + vbCr + "続行します。"
                    MessageBox.Show(stat,    "  認  証  確  認  ",    MessageBoxButtons.OK,
MessageBoxIcon.Information)
                    '猶予（試用）残日数があるので、Button1、Button2、Button3 を含む
                    'GroupBox1 の Enabled プロパティを True にして有効とする。
                    GroupBox1.Enabled = True

                    Label2.Text =
                        "現在、「猶予（試用）期間内」のため「処理メニュー」を" + vbCr +
                        "表示しています。"
                Case 400 '未認証（猶予無効時）
                    stat =
                    "**********" + vbCr +
                    "これは、サンプルメッセージです。" + vbCr +
                    "**********" + vbCr +
                    "[ID:" + ret.ToString + "] " + "ライセンスが認証されていません。" + vbCr + "メニ
ューは実行できません。" + vbCr + "「ライセンス管理」からライセンスの認証登録を行ってください。"

                    MessageBox.Show(stat,    "  認  証  確  認  ",    MessageBoxButtons.OK,
MessageBoxIcon.Exclamation)

                    Label2.Text =
                        "現在、「未承認」のため「処理メニュー」はグレーアウトに" + vbCr +
                        "なっています。"
                Case 500 '認証済み
                    '認証済みなので、Button1、Button2、Button3 を含む
                    'GroupBox1 の Enabled プロパティを True にして有効とする。
                    GroupBox1.Enabled = True

                    Label2.Text =
                        "現在、「認証済」のため「処理メニュー」を表示しています。"
                Case -999 '認証済ハードウェア情報不一致
                    stat =
                    "**********" + vbCr +
                    "これは、サンプルメッセージです。" + vbCr +
                    "**********" + vbCr +
                    "[ID:" + ret.ToString + "] " + "認証済ですが認証時のハードウェア情報と一致しない
情報があります。"
                    MessageBox.Show(stat,    "  認  証  確  認  ",    MessageBoxButtons.OK,
MessageBoxIcon.Exclamation)

                    Label2.Text =
                        "現在、「認証済ハードウェア情報不一致」のため「処理メニュー」は" + vbCr +
                        "グレーアウトになっています。"
                Case -1 'その他エラー
                    stat =
                    "**********" + vbCr +
                    "これは、サンプルメッセージです。" + vbCr +
                    "**********" + vbCr +
                    "[ID:" + ret.ToString + "] " + "認証状況確認中に何らかのエラーが発生しました。"
                    MessageBox.Show(stat,    "  認  証  確  認  ",    MessageBoxButtons.OK,
MessageBoxIcon.Exclamation)
```

```
                Label2.Text =
                    "現在、「その他エラー」のため「処理メニュー」はグレーアウト" + vbCr +
                    "になっています。"
            Case 20000101 To 21001231 '「有効期限」以内
                stat =
                "**********************************" + vbCr +
                "これは、サンプルメッセージです。" + vbCr +
                "**********************************" + vbCr +
                "有効期限は、" + Mid(ret, 1, 4) + "/" + Mid(ret, 5, 2) + "/" + Mid(ret, 7, 2) + "
までです。 "
                MessageBox.Show(stat,     "   認   証   確   認   ",     MessageBoxButtons.OK,
MessageBoxIcon.Information)
                GroupBox1.Enabled = True

                Label2.Text =
                    "現在、「認証登録済/有効期間内」のため「処理メニュー」を" + vbCr +
                    "表示しています。"
            Case -21001231 To -20000101 '「有効期限」以外
                ret = ret * -1
                stat =
                "**********************************" + vbCr +
                "これは、サンプルメッセージです。" + vbCr +
                "**********************************" + vbCr +
                "有効期限は、" + Mid(ret, 1, 4) + "/" + Mid(ret, 5, 2) + "/" + Mid(ret, 7, 2) + "
までです。 "
                stat = stat + vbCr + "有効期限が切れました。"
                MessageBox.Show(stat,     "   認   証   確   認   ",     MessageBoxButtons.OK,
MessageBoxIcon.Exclamation)

                Label2.Text =
                    "現在、「認証登録済/有効期限切れ」のため「処理メニュー」は" + vbCr +
                    "グレーアウトになっています。"
            Case 366 '日付データの取得失敗（猶予）
                stat =
                "**********************************" + vbCr +
                "これは、サンプルメッセージです。" + vbCr +
                "**********************************" + vbCr +
                "日付データの取得に失敗しました。（猶予）"
                MessageBox.Show(stat,     "   認   証   確   認   ",     MessageBoxButtons.OK,
MessageBoxIcon.Exclamation)

                Label2.Text =
                    "現在、「日付データの取得失敗（猶予）」のため「処理メニュー」は" + vbCr +
                    "グレーアウトになっています。"
            Case -21001232 '日付データの取得失敗（有効期限）
                stat =
                "**********************************" + vbCr +
                "これは、サンプルメッセージです。" + vbCr +
                "**********************************" + vbCr +
                "日付データの取得に失敗しました。（有効期限）"
                MessageBox.Show(stat,     "   認   証   確   認   ",     MessageBoxButtons.OK,
MessageBoxIcon.Exclamation)

                Label2.Text =
                    "現在、「日付データの取得失敗（有効期限）」のため「処理メニュー」は" + vbCr +
                    "グレーアウトになっています。"
            Case -3 'PC の日付が意図的に変更された
                stat =
```

```
                "**********************************" + vbCr +
                "これは、サンプルメッセージです。" + vbCr +
                "**********************************" + vbCr +
                "PC の日付が変更されました。"
                MessageBox.Show(stat,    "   認   証   確   認   ",    MessageBoxButtons.OK,
MessageBoxIcon.Exclamation)

                Label2.Text =
                    "現在、「PC の日付が意図的に変更された」ため「処理メニュー」は" + vbCr +
                    "グレーアウトになっています。"
            Case 600 'フローティングライセンス PC として登録済
                stat =
                "**********************************" + vbCr +
                "これは、サンプルメッセージです。" + vbCr +
                "**********************************" + vbCr +
                "フローティングライセンス登録済です。"
                MessageBox.Show(stat,    "   認   証   確   認   ",    MessageBoxButtons.OK,
MessageBoxIcon.Information)
                GroupBox1.Enabled = True


                Label2.Text =
                    "現在、「フローティングライセンス登録済」のため「処理メニュー」を" + vbCr +
                    "表示しています。"
            Case 200001010 To 210012310 '「有効期限」以内で フローティングライセンスが登録済
                ret = ret / 10
                stat =
                "**********************************" + vbCr +
                "これは、サンプルメッセージです。" + vbCr +
                "**********************************" + vbCr +
                "有効期限は、" + Mid(ret, 1, 4) + "/" + Mid(ret, 5, 2) + "/" + Mid(ret, 7, 2) + "
までです。 "
                MessageBox.Show(stat,    "   認   証   確   認   ",    MessageBoxButtons.OK,
MessageBoxIcon.Information)
                GroupBox1.Enabled = True

                Label2.Text =
                    "現在、「フローティングライセンス登録済/有効期間内」のため" + vbCr +
                    "「処理メニュー」を表示しています。"
            Case -210012310 To -200001010 '「有効期限」以外でフローティングライセンスが登録済
                ret = ret / -10
                stat =
                "**********************************" + vbCr +
                "これは、サンプルメッセージです。" + vbCr +
                "**********************************" + vbCr +
                "有効期限は、" + Mid(ret, 1, 4) + "/" + Mid(ret, 5, 2) + "/" + Mid(ret, 7, 2) + "
までです。 "
                stat = stat + vbCr + "有効期限が切れました。"
                MessageBox.Show(stat,    "   認   証   確   認   ",    MessageBoxButtons.OK,
MessageBoxIcon.Exclamation)

                Label2.Text =
                    "現在、「フローティングライセンス登録済/有効期限切れ」のため" + vbCr +
                    "「処理メニュー」はグレーアウトになっています。"
            Case Else '想定外
                stat =
                "**********************************" + vbCr +
                "これは、サンプルメッセージです。" + vbCr +
                "**********************************" + vbCr +
```

```
                "[ID:" + ret.ToString + "] " + "認証状況確認中に想定外のエラーが発生しました。"
                MessageBox.Show(stat)

                Label2.Text =
                    "現在、「想定外のエラー」のため「処理メニュー」は" + vbCr +
                    "グレーアウトになっています。"
            End Select

        End Sub

    ・・・

End Class
```

次の例では、認証状況表示ダイアログを呼び出しています。

```
Public Class Form2

    Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles Button1.Click
        ' 認証状況表示
        If Class1.myActivate.ActivateStatusDisp() = False Then
            MessageBox.Show("エラー")
        End If
    End Sub

    ・・・

End Class
```

以下の例では、認証の登録や解除に関する各処理を呼び出しています。

```
Public Class Form2

    Private Sub Button2_Click(sender As System.Object, e As System.EventArgs) Handles Button2.Click

        '認証登録/インターネット
        If Class1.myActivate.ActivateRegisterInternet() = False Then
            MessageBox.Show("エラー")
        End If

    End Sub


    Private Sub Button3_Click(sender As System.Object, e As System.EventArgs) Handles Button3.Click

        '認証登録/電話
        If Class1.myActivate.ActivateRegisterTelephone() = False Then
            MessageBox.Show("エラー")
        End If

    End Sub

    Private Sub Button4_Click(sender As System.Object, e As System.EventArgs) Handles Button4.Click

        '認証解除/インターネット
        If Class1.myActivate.ActivateRemoveInternet() = False Then
            MessageBox.Show("エラー")
        End If
```

```
    End Sub

    Private Sub Button5_Click(sender As System.Object, e As System.EventArgs) Handles Button5.Click

        '認証解除/電話
        If Class1.myActivate.ActivateRemoveTelephone() = False Then
            MessageBox.Show("エラー")
        End If

    End Sub

    Private Sub Button6_Click(sender As System.Object, e As System.EventArgs) Handles Button6.Click

        '代理認証登録/準備
        If Class1.myActivate.ProxyActivateRegisterPrepare() = False Then
            MessageBox.Show("エラー")
        End If

    End Sub

    Private Sub Button7_Click(sender As System.Object, e As System.EventArgs) Handles Button7.Click

        '代理認証登録/確定
        If Class1.myActivate.ProxyActivateRegisterFix() = False Then
            MessageBox.Show("エラー")
        End If

    End Sub

    Private Sub Button8_Click(sender As System.Object, e As System.EventArgs) Handles Button8.Click

        '代理認証解除/準備
        If Class1.myActivate.ProxyActivateRemovePrepare() = False Then
            MessageBox.Show("エラー")
        End If

    End Sub

・・・

End Class
```

C# 2015 の場合

付属のサンプルプロジェクト(NRDDLL¥SampleProject¥CSharp¥PackageApp¥SamplePackageApp.FW45)の例で説明します。

最初に、Newtone.NR.Activationクラスのインスタンスを作成します。
次の例では、2つのFormで同一のインスタンスで使用するためクラスClass1内でpublic staticで宣言しています。

```
class Class1
{
    public static Newtone.NR.FW45.NRD_Activation myActivate = new
Newtone.NR.FW45.NRD_Activation();
}
```

次に DLL のプロパティを設定します。
これらのプロパティは通常、最初に 1 回だけ固定的に設定するコードを記述します。

```
public partial class Form1 : Form
{
    private void Form1_Load(object sender, EventArgs e)
    {
        //-----------------------------------------------------------------
        //Newtone.NR.FW45.dllのプロパティの設定
        //【重要】DLLの以下のプロパティは必ず適切なものを設定してください。
        //-----------------------------------------------------------------

        //ベンダ製品スタート開始レジストリキーパス(※最終的には必ず貴社のものに変更してください)
        Class1.myActivate.VendorsProductStartRegistryKeyPath =
"Software¥¥Newtone¥¥NinshoRescue¥¥NRD¥¥SampleProject";
        //電話で認証時の電話番号(※最終的には必ず貴社のものに変更してください)
        Class1.myActivate.TelephoneNumber = "012-345-6789";
        //暗号化時のパスワード(※最終的には必ず貴社のものに変更してください)
        Class1.myActivate.EncryptionPassword = "12345678ABCDEFGH";
        //暗号化時のSalt文字列(8バイト以上)(※最終的には必ず貴社のものに変更してください)
        Class1.myActivate.EncryptionSaltString = "認証レスキュー！";
        //猶予日数(デフォルト:0日、設定可能範囲:1～365日)(※最終的には必ず貴社のものに変更してください)
        Class1.myActivate.TrialPeriod = 0;
        //猶予期間の名称(デフォルト:"猶予(試用)")
        Class1.myActivate.TrialPeriodName = "猶予 (試用) ";
        //MACアドレスの使用(デフォルト:True)「代理認証」利用時はMACアドレス必須
        Class1.myActivate.UseMacAddress = true;
        //CPU情報の使用(デフォルト:True)
        Class1.myActivate.UseCpuInfo = true;
        //WebサービスのURL(※最終的には必ず貴社のものに変更してください)
        Class1.myActivate.WebServiceURL = "http://localhost/NRDWebService/Service.asmx";
        //Webサービス時の基本認証の使用(デフォルト:False)
        Class1.myActivate.WebServiceUseBasicAuthentication = false;
        //Webサービス時の基本認証ユーザ名
        Class1.myActivate.WebServiceBasicAuthenticationUserName = "";
        //Webサービス時の基本認証パスワード
        Class1.myActivate.WebServiceBasicAuthenticationPassword = "";
        //Webサービス確認パスワード(※最終的には必ず貴社のものに変更してください)
        Class1.myActivate.WebServiceCheckPassword = "ABcd1234";
        //プロダクトIDの桁数(デフォルト:17)(※最終的には必ず貴社のものに変更してください)
        Class1.myActivate.ProductIdNumberOfDigits = 17;
```

```
        //シリアルNo.の桁数(デフォルト:8)(※最終的には必ず貴社のものに変更してください)
        Class1.myActivate.SerialNoNumberOfDigits = 8;
        //Webサービスのタイムアウト(デフォルト:60秒)
        Class1.myActivate.WebServiceTimeout = 60;
        //認証登録時の設定プロダクトID(デフォルト:空文字列)
        Class1.myActivate.SetProductID = "";
        //認証登録時の設定シリアルNo.(デフォルト:空文字列)
        Class1.myActivate.SetSerialNo = "";
        //無効になっているNICを無視する(高速化を図る)(デフォルト:True)
        Class1.myActivate.DisabledNICIgnore = true;

        CertificationStatus(); //認証状況の確認
    }

・・・

}
```

後は、必要に応じてDLLのメソッドを呼び出します。

次の例では、ActivateStatusCheck()メソッドで「認証状態確認」を行います。

```
public partial class Form1 : Form
{
    private void CertificationStatus()
    {
        //----------------------------------------
        //認証状況の確認
        //----------------------------------------

        //Button1、Button2、Button3を含むGroupBox1のEnabledプロパティをFalseにして無効とする。
        GroupBox1.Enabled = false;

        int ret = 0;
        string stat = null;
        //表示メッセージ

        ret = Class1.myActivate.ActivateStatusCheck();

        //認証状況確認


        if (ret>=1 && ret<=365)
        {
            //猶予日数有

            stat =
                "********************************" + Environment.NewLine +
                "これは、サンプルメッセージです。" + Environment.NewLine +
                "********************************" + Environment.NewLine +
                "[ID:" + ret.ToString() + "] " + Class1.myActivate.TrialPeriodName + "期間残日数
は" + ret.ToString() + "日です。" + Environment.NewLine + "続行します。";
            MessageBox.Show(stat, "認証確認", MessageBoxButtons.OK, MessageBoxIcon.Information);
            //猶予（試用）残日数があるので、Button1、Button2、Button3を含む
            //GroupBox1のEnabledプロパティをTrueにして有効とする。
            GroupBox1.Enabled = true;

            Label2.Text =
```

```
                "現在、「猶予（試用）期間内」のため「処理メニュー」を" + Environment.NewLine +
                "表示しています。";

            return;
        }


        if (ret >= 20000101 && ret <= 21001231)
        {
            //「有効期限」以内

            String retStr = "";
            retStr = ret.ToString();

            stat =
                "********************************" + Environment.NewLine +
                "これは、サンプルメッセージです。" + Environment.NewLine +
                "********************************" + Environment.NewLine +
                "有効期限は、" + retStr.Substring(0, 4) + "/" + retStr.Substring(4, 2) + "/" +
retStr.Substring(6, 2) + "までです。 ";
            MessageBox.Show(stat, "認証確認", MessageBoxButtons.OK, MessageBoxIcon.Information);
            GroupBox1.Enabled = true;

            Label2.Text =
                "現在、「認証登録済/有効期間内」のため「処理メニュー」を"+Environment.NewLine+
                "表示しています。";

            return;
        }

        if (ret >= -21001231 && ret <= -20000101)
        {
            //「有効期限」以外

            String retStr = "";
            ret = ret * -1;
            retStr = ret.ToString();

            stat =
            "********************************" + Environment.NewLine +
            "これは、サンプルメッセージです。" + Environment.NewLine +
            "********************************" + Environment.NewLine +
            "有効期限は、" + retStr.Substring(0, 4) + "/" + retStr.Substring(4, 2) + "/" +
retStr.Substring(6, 2) + "までです。 ";
            stat = stat + Environment.NewLine + "有効期限が切れました。";
            MessageBox.Show(stat, "認証確認", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

            Label2.Text =
                "現在、「認証登録済/有効期限切れ」のため「処理メニュー」は"+Environment.NewLine
+
                "グレーアウトになっています。";

            return;
        }

        if (ret >= 200001010 && ret <= 210012310)
        {
            //「有効期限」以内で フローティングライセンスが登録済
```

```
            ret = ret / 10;
            String retStr = "";
            retStr = ret.ToString();

            stat =
                "********************************" + Environment.NewLine +
                "これは、サンプルメッセージです。" + Environment.NewLine +
                "********************************" + Environment.NewLine +
                "有効期限は、" + retStr.Substring(0, 4) + "/" + retStr.Substring(4, 2) + "/" +
retStr.Substring(6, 2) + "までです。";
            MessageBox.Show(stat, "フローティングライセンス", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            GroupBox1.Enabled = true;

            Label2.Text =
                "現在、「フローティングライセンス登録済/有効期間内」のため"+Environment.NewLine
+
                "「処理メニュー」を表示しています。";

            return;
        }

        if (ret >= -210012310 && ret <= -200001010)
        {
            //「有効期限」以外でフローティングライセンスが登録済

            ret = ret / -10;
            String retStr = "";
            ret = ret * -1;
            retStr = ret.ToString();

            stat =
            "********************************" + Environment.NewLine +
            "これは、サンプルメッセージです。" + Environment.NewLine +
            "********************************" + Environment.NewLine +
            "有効期限は、" + retStr.Substring(0, 4) + "/" + retStr.Substring(4, 2) + "/" +
retStr.Substring(6, 2) + "までです。";
            stat = stat + Environment.NewLine + "有効期限が切れました。";
            MessageBox.Show(stat, "フローティングライセンス", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);

            Label2.Text =
                "現在、「フローティングライセンス登録済/有効期限切れ」のため"+Environment.NewLine
+
                "「処理メニュー」はグレーアウトになっています。";

            return;
        }

        switch (ret)
        {
            case 0:
                //期限切れ（猶予有効時）

                stat =
                    "********************************" + Environment.NewLine +
                    "これは、サンプルメッセージです。" + Environment.NewLine +
                    "********************************" + Environment.NewLine +
                    "[ID:" + ret.ToString() + "] " + Class1.myActivate.TrialPeriodName + "期限が
```

```
切れました。" + Environment.NewLine + "メニューは実行できません。" + Environment.NewLine + "「ライセンス管理」からライセンスの認証登録を行ってください。";
                MessageBox.Show(stat, "認証確認", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);

                Label2.Text =
                    "現在、「猶予（試用）期限切れ」のため「処理メニュー」は"+Environment.NewLine
+
                    "グレーアウトになっています。";

                break;
            case 400:
                //未認証（猶予無効時）
                stat =
                    "********************************" + Environment.NewLine +
                    "これは、サンプルメッセージです。" + Environment.NewLine +
                    "********************************" + Environment.NewLine +
                    "[ID:" + ret.ToString() + "] " + "ライセンスが認証されていません。" +
Environment.NewLine + "メニューは実行できません。" + Environment.NewLine + "「ライセンス管理」からライセンスの認証登録を行ってください。";

                MessageBox.Show(stat, "認証確認", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);

                Label2.Text =
                    "現在、「未承認」のため「処理メニュー」はグレーアウトに"+Environment.NewLine
+
                    "なっています。";

                break;
            case 500:
                //認証済み
                //認証済みなので、Button1、Button2、Button3を含む
                //GroupBox1のEnabledプロパティをTrueにして有効とする。
                GroupBox1.Enabled = true;

                Label2.Text =
                    "現在、「認証済」のため「処理メニュー」を表示しています。";

                break;

            case -999:
                //認証済ハードウェア情報不一致

                stat =
                    "********************************" + Environment.NewLine +
                    "これは、サンプルメッセージです。" + Environment.NewLine +
                    "********************************" + Environment.NewLine +
                    "[ID:" + ret.ToString() + "] " + "認証済ですが認証時のハードウェア情報と一致しない情報があります。";
                MessageBox.Show(stat, "認証確認", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);

                Label2.Text =
                    "現在、「認証済ハードウェア情報不一致」のため「処理メニュー」は" +
Environment.NewLine +
                    "グレーアウトになっています。";

                break;
```

```
            case -1:
                //その他エラー

                stat =
                    "********************************" + Environment.NewLine +
                    "これは、サンプルメッセージです。" + Environment.NewLine +
                    "********************************" + Environment.NewLine +
                    "[ID:" + ret.ToString() + "] " + "認証状況確認中に何らかのエラーが発生しました。";
                MessageBox.Show(stat, "認証確認", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

                Label2.Text =
                    "現在、「その他エラー」のため「処理メニュー」はグレーアウト" + Environment.NewLine +
                    "になっています。";

                break;

            case 366:
                //日付データの取得失敗（猶予）

                stat =
                    "********************************" + Environment.NewLine +
                    "これは、サンプルメッセージです。" + Environment.NewLine +
                    "********************************" + Environment.NewLine +
                    "日付データの取得に失敗しました。（猶予）";
                MessageBox.Show(stat, "認証確認", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

                Label2.Text =
                    "現在、「日付データの取得失敗（猶予）」のため「処理メニュー」は" + Environment.NewLine +
                    "グレーアウトになっています。";

                break;
            case -21001232:
                //日付データの取得失敗（有効期限）

                stat =
                    "********************************" + Environment.NewLine +
                    "これは、サンプルメッセージです。" + Environment.NewLine +
                    "********************************" + Environment.NewLine +
                    "日付データの取得に失敗しました。（有効期限）";
                MessageBox.Show(stat, "認証確認", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

                Label2.Text =
                    "現在、「日付データの取得失敗（有効期限）」のため「処理メニュー」は" + Environment.NewLine +
                    "グレーアウトになっています。";

            break;

            case -3:
                //PCの日付が意図的に変更された

                stat =
                    "********************************" + Environment.NewLine +
```

```
                    "これは、サンプルメッセージです。" + Environment.NewLine +
                    "********************************" + Environment.NewLine +
                    "PCの日付が変更されました。";
                MessageBox.Show(stat, "認証確認", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);

                Label2.Text =
                    "現在、「PCの日付が意図的に変更された」ため「処理メニュー」は" +
Environment.NewLine +
                    "グレーアウトになっています。";

                break;

            case 600:
                //フローティングライセンスPCとして登録済

                stat =
                    "********************************" + Environment.NewLine +
                    "これは、サンプルメッセージです。" + Environment.NewLine +
                    "********************************" + Environment.NewLine +
                    "フローティングライセンス登録済です。";
                MessageBox.Show(stat, "認証確認", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                GroupBox1.Enabled = true;

                Label2.Text =
                    "現在、「フローティングライセンス登録済」のため「処理メニュー」を" +
Environment.NewLine +
                    "表示しています。";

                break;

            default:
                //想定外

                stat =
                    "********************************" + Environment.NewLine +
                    "これは、サンプルメッセージです。" + Environment.NewLine +
                    "********************************" + Environment.NewLine +
                    "[ID:" + ret.ToString() + "] " + "認証状況確認中に想定外のエラーが発生しました";
                MessageBox.Show(stat);

                Label2.Text =
                    "現在、「想定外のエラー」のため「処理メニュー」は" + Environment.NewLine +
                    "グレーアウトになっています。";

                break;
        }
    }

・・・

}
```

次の例では、認証状況表示ダイアログを呼び出しています。

```
public partial class Form2 : Form
```

```
{
    private void Button1_Click(object sender, EventArgs e)
    {
        // 認証状況表示
        if (Class1.myActivate.ActivateStatusDisp() == false)
        {
            MessageBox.Show("エラー");
        }
    }
}
```

以下の例では、認証の登録や解除に関する各処理を呼び出しています。

```
public partial class Form2 : Form
{
    private void Button2_Click(object sender, EventArgs e)
    {
        //認証登録/インターネット
        if (Class1.myActivate.ActivateRegisterInternet() == false)
        {
            MessageBox.Show("エラー");
        }
    }

    private void Button3_Click(object sender, EventArgs e)
    {
        //認証登録/電話
        if (Class1.myActivate.ActivateRegisterTelephone() == false)
        {
            MessageBox.Show("エラー");
        }
    }

    private void Button4_Click(object sender, EventArgs e)
    {
        //認証解除/インターネット
        if (Class1.myActivate.ActivateRemoveInternet() == false)
        {
            MessageBox.Show("エラー");
        }
    }

    private void Button5_Click(object sender, EventArgs e)
    {
        //認証解除/電話
        if (Class1.myActivate.ActivateRemoveTelephone() == false)
        {
            MessageBox.Show("エラー");
        }
    }

    private void Button6_Click(object sender, EventArgs e)
    {
        //代理認証登録/準備
        if (Class1.myActivate.ProxyActivateRegisterPrepare() == false)
        {
            MessageBox.Show("エラー");
        }
```

```
    }

    private void Button7_Click(object sender, EventArgs e)
    {
        //代理認証登録/確定
        if (Class1.myActivate.ProxyActivateRegisterFix() == false)
        {
            MessageBox.Show("エラー");
        }
    }

    private void Button8_Click(object sender, EventArgs e)
    {
        //代理認証解除/準備
        if (Class1.myActivate.ProxyActivateRemovePrepare() == false)
        {
            MessageBox.Show("エラー");
        }
    }

・・・

}
```

Visual C++の場合

付属のサンプルプロジェクト(NRDDLL¥SampleProject¥VC++¥PackageApp)の例で説明します。

最初に、NewtoneNRDvcpp.INRD_Activationクラスを使用するためにタイプライブラリのインポートをします。
次の例では、2つのダイアログで同一のインスタンスで使用するためSamplePackageApp.FW45.h内でタイプライブラリのインポート、クラスの宣言をしています。

認証レスキュー.NETのVC++用タイプライブラリをインポートします。

```
#import "..¥¥..¥¥..¥¥..¥¥NRDLL¥¥NewtoneNRDvcpp.tlb"
```

認証レスキュー.NETのVC++用タイプライブラリの名前空間を宣言します。

```
using namespace NewtoneNRDvcpp;

class CSamplePackageAppFW45App : public CWinApp
{
public:
	CSamplePackageAppFW45App();

// オーバーライド
public:
	virtual BOOL InitInstance();

	//認証レスキューActivationクラスのポインタ
	INRD_ActivationPtr m_pActivation;

// 実装

	DECLARE_MESSAGE_MAP()
};

extern CSamplePackageAppFW45App theApp;
```

次にSamplePackageApp.FW45Dlg.cpp内のOnInitDialogイベント内でvcppActivationクラスのinterfaceポインタを作成してNewtoneNRDvcpp.IActivationクラスのインスタンスを作成します。
次に DLL のプロパティを設定します。これらのプロパティは通常、最初に 1 回だけ固定的に設定するコードを記述するので PackageAppDlg.cpp 内の OnInitDialog イベント内に記述します。

```
BOOL CSamplePackageAppFW45Dlg::OnInitDialog()
{
		・・・


	// COM の初期化
	HRESULT hr = CoInitialize(NULL);

	// vcppActivationクラスのinterfaceポインタを作る
	INRD_ActivationPtr pIAct(__uuidof(vcppNRD_Activation));

	theApp.m_pActivation = pIAct;


	//ベンダ製品スタート開始レジストリキーパス（※最終的には必ず貴社のものに変更してください）
	theApp.m_pActivation->VendorsProductStartRegistryKeyPath =
"Software¥¥Newtone¥¥NinshoRescue¥¥NRD¥¥SampleProject";
```

```
    //電話で認証時の電話番号（※最終的には必ず貴社のものに変更してください）
    theApp.m_pActivation->TelephoneNumber = "012-345-6789";
    //暗号化時のパスワード（※最終的には必ず貴社のものに変更してください）
    theApp.m_pActivation->EncryptionPassword = "12345678ABCDEFGH";
    //暗号化時のSalt文字列(8バイト以上)（※最終的には必ず貴社のものに変更してください）
    theApp.m_pActivation->EncryptionSaltString = "認証レスキュー！";
    //猶予日数（デフォルト：0日、設定可能範囲：1～365日）（※最終的には必ず貴社のものに変更してください）
    theApp.m_pActivation->TrialPeriod = 0;
    //猶予期間の名称（デフォルト："猶予（試用）"）
    theApp.m_pActivation->TrialPeriodName = "猶予（試用）";
    //MACアドレスの使用（デフォルト：True）「代理認証」利用時はMACアドレス必須
    theApp.m_pActivation->UseMacAddress = true;
    //CPU情報の使用（デフォルト：True）
    theApp.m_pActivation->UseCpuInfo = true;
    //WebサービスのURL（※最終的には必ず貴社のものに変更してください）
    theApp.m_pActivation->WebServiceURL = "http://localhost/NRDWebService/Service.asmx";
    //Webサービス時の基本認証の使用（デフォルト：False）
    theApp.m_pActivation->WebServiceUseBasicAuthentication = false;
    //Webサービス時の基本認証ユーザ名
    theApp.m_pActivation->WebServiceBasicAuthenticationUserName = "";
    //Webサービス時の基本認証パスワード
    theApp.m_pActivation->WebServiceBasicAuthenticationPassword = "";
    //Webサービス確認パスワード（※最終的には必ず貴社のものに変更してください）
    theApp.m_pActivation->WebServiceCheckPassword = "ABcd1234";
    //プロダクトIDの桁数（デフォルト：17）（※最終的には必ず貴社のものに変更してください）
    theApp.m_pActivation->ProductIdNumberOfDigits = 17;
    //シリアルNo.の桁数（デフォルト：8）（※最終的には必ず貴社のものに変更してください）
    theApp.m_pActivation->SerialNoNumberOfDigits = 8;
    //Webサービスのタイムアウト（デフォルト：60秒）
    theApp.m_pActivation->WebServiceTimeout = 60;
    //認証登録時の設定プロダクトID（デフォルト：空文字列）
    theApp.m_pActivation->SetProductID = "";
    //認証登録時の設定シリアルNo.（デフォルト：空文字列）
    theApp.m_pActivation->SetSerialNo = "";
    //無効になっているNICを無視する（高速化を図る）（デフォルト：True）
    theApp.m_pActivation->DisabledNICIgnore = true;

    //認証状況の確認
    CertificationStatus();

        ・・・

}
```

アプリケーションの終了時にCOMの終了処理をします。

```
//例：ダイアログの「×」ボタンで終了した時の処理
void CSamplePackageAppFW45Dlg::OnClose()
{
    // TODO: ここにメッセージ ハンドラー コードを追加するか、既定の処理を呼び出します。
    CDialogEx::OnClose();

    // Uninitialize COM.
    CoUninitialize();

}
```

後は、必要に応じてDLLのメソッドを呼び出します。

次の例では、ActivateStatusCheck()メソッドで「認証状態確認」を行います。

```
void CSamplePackageAppFW45Dlg::CertificationStatus()
{
    //Button1、Button2、Button3を無効とする。
    Button1.EnableWindow(false);
    Button2.EnableWindow(false);
    Button3.EnableWindow(false);

    //表示メッセージ
    CString stat;

    long ret = theApp.m_pActivation->ActivateStatusCheck();
    CString strRet;
    strRet.Format(_T("%d"), ret);
    CString tpName = theApp.m_pActivation->TrialPeriodName;

    if (ret >= 1 && ret <= 365)
    {
        //猶予日数有
        stat = _T("**********¥r　これは、サンプルメッセージです。¥r**********¥r[ID:")
            + strRet + _T("] ") + tpName + _T("期間残日数は") + strRet + _T("日です。¥r続行します。");
        MessageBox(stat, _T("認証確認"), MB_OK);

        //猶予（試用）残日数があるので、Button1、Button2、Button3を有効とする。
        Button1.EnableWindow(true);
        Button2.EnableWindow(true);
        Button3.EnableWindow(true);

        txtInfo2 = _T("現在、「猶予（試用）期間内」のため¥r「処理メニュー」を表示しています。");
    }
    else if (ret >= 20000101 && ret <= 21001231)
    {
        //「有効期限」内
        stat = _T("**********¥r　これは、サンプルメッセージです。¥r**********¥r有効期限は")
            + strRet.Mid(0, 4) + _T("/") + strRet.Mid(4, 2) + _T("/") + strRet.Mid(6, 2) + _T("までです。");
        MessageBox(stat, _T("認証確認"), MB_OK);

        Button1.EnableWindow(true);
        Button2.EnableWindow(true);
        Button3.EnableWindow(true);

        txtInfo2 = _T("現在、「認証登録済/有効期間内」のため¥r「処理メニュー」を表示しています。");
    }
    else if (ret >= -21001231 && ret <= -20000101)
    {
        ret *= -1;
        strRet.Format(_T("%d"), ret);

        //「有効期限」外
        stat = _T("**********¥r　これは、サンプルメッセージです。
```

```
¥r**********¥r有効期限は")
            + strRet.Mid(0, 4) + _T("/") + strRet.Mid(4, 2) + _T("/") + strRet.Mid(6, 2) + _T("
までです。")
            + _T("¥r有効期限が切れました。");
        MessageBox(stat, _T("認証確認"), MB_OK);

        txtInfo2 = _T("現在、「認証登録済/有効期間切れ」のため¥r「処理メニュー」は無効になって
います。");
    }
    else if (ret >= 200001010 && ret <= 210012310)
    {
        //「有効期限」内で フローティングライセンスが登録済

        ret /= 10;
        strRet.Format(_T("%d"), ret);

        stat = _T("**********¥r  これは、サンプルメッセージです。
¥r**********¥r有効期限は")
            + strRet.Mid(0, 4) + _T("/") + strRet.Mid(4, 2) + _T("/") + strRet.Mid(6, 2) + _T("
までです。");
        MessageBox(stat, _T("フローティングライセンス"), MB_OK);

        Button1.EnableWindow(true);
        Button2.EnableWindow(true);
        Button3.EnableWindow(true);

        txtInfo2 = _T("現在、「フローティングライセンス登録済/有効期間内」のため¥r「処理メニュ
ー」を表示しています。");
    }
    else if (ret >= -210012310 && ret <= -200001010)
    {
        //「有効期限」外で フローティングライセンスが登録済

        ret /= -10;
        strRet.Format(_T("%d"), ret);

        stat = _T("**********¥r  これは、サンプルメッセージです。
¥r**********¥r有効期限は")
            + strRet.Mid(0, 4) + _T("/") + strRet.Mid(4, 2) + _T("/") + strRet.Mid(6, 2) + _T("
までです。")
            + _T("¥r有効期限が切れました。");
        MessageBox(stat, _T("フローティングライセンス"), MB_OK);

        txtInfo2 = _T("現在、「フローティングライセンス登録済/有効期限切れ」のため¥r「処理メニ
ュー」は無効になっています。");
    }
    else if (ret == 0)
    {
        //期限切れ（猶予有効時）
        stat = _T("**********¥r  これは、サンプルメッセージです。
¥r**********¥r[ID:")
            + strRet + _T("] ") + tpName + _T("期限が切れました。¥rメニューは実行できません。
¥r「ライセンス管理」からライセンスの認証登録を行ってください。");
        MessageBox(stat, _T("認証確認"), MB_OK);

        txtInfo2 = _T("現在、「猶予（試用）期限切れ」のため¥r「処理メニュー」は無効になっていま
す。");
    }
    else if (ret == 400)
```

```
    {
        //未認証（猶予無効時）
        stat = _T("**********¥r　これは、サンプルメッセージです。¥r**********¥r[ID:")
            + strRet + _T("] ライセンスが認証されていません。¥rメニューは実行できません。¥r「ライセンス管理」からライセンスの認証登録を行ってください。");
        MessageBox(stat, _T("認証確認"), MB_OK);

        txtInfo2 = _T("未承認」のため¥r「処理メニュー」は無効になっています。");

    }
    else if (ret == 500)
    {
        //認証済み
        //認証済みなので、Button1、Button2、Button3を有効とする。
        Button1.EnableWindow(true);
        Button2.EnableWindow(true);
        Button3.EnableWindow(true);

        txtInfo2 = _T("現在、「認証済」のため¥r「処理メニュー」を表示しています。");
    }
    else if (ret == -999)
    {
        //認証済ハードウェア情報不一致
        stat = _T("**********¥r　これは、サンプルメッセージです。¥r**********¥r[ID:")
            + strRet + _T("] 認証済ですが認証時のハードウェア情報と一致しない情報があります。");
        MessageBox(stat, _T("認証確認"), MB_OK);

        txtInfo2 = _T("現在、「認証済ハードウェア情報不一致」のため¥r「処理メニュー」は無効になっています。");
    }
    else if (ret == -1)
    {
        //その他エラー
        stat = _T("**********¥r　これは、サンプルメッセージです。¥r**********¥r[ID:")
            + strRet + _T("] 認証状況確認中に何らかのエラーが発生しました。");
        MessageBox(stat, _T("認証確認"), MB_OK);

        txtInfo2 = _T("現在、「その他エラー」のため¥r「処理メニュー」は無効になっています。");
    }
    else if (ret == 366)
    {
        //日付データの取得失敗（猶予）
        stat = _T("**********¥r　これは、サンプルメッセージです。¥r**********¥r日付データの取得に失敗しました。（猶予）");
        MessageBox(stat, _T("認証確認"), MB_OK);

        txtInfo2 = _T("現在、「日付データの取得失敗（猶予）」のため¥r「処理メニュー」は無効になっています。");
    }
    else if (ret == -21001232)
    {
        //日付データの取得失敗（有効期限）
        stat = _T("**********¥r　これは、サンプルメッセージです。¥r**********¥r日付データの取得に失敗しました。（有効期限）");
        MessageBox(stat, _T("認証確認"), MB_OK);
```

```
        txtInfo2 = _T("現在、「日付データの取得失敗（有効期限）」のため¥r「処理メニュー」は無効になっています。");
    }
    else if (ret == -3)
    {
        //PCの日付が意図的に変更された
        stat = _T("**********¥r　これは、サンプルメッセージです。¥r**********¥rPC の日付が変更されました。");
        MessageBox(stat, _T("認証確認"), MB_OK);

        txtInfo2 = _T("現在、「PC の日付が意図的に変更された」のため¥r「処理メニュー」は無効になっています。");
    }
    else if (ret == 600)
    {
        //フローティングライセンスPCとして登録済
        stat = _T("**********¥r　これは、サンプルメッセージです。¥r**********¥rフローティングライセンス登録済です。");
        MessageBox(stat, _T("認証確認"), MB_OK);

        Button1.EnableWindow(true);
        Button2.EnableWindow(true);
        Button3.EnableWindow(true);

        txtInfo2 = _T("現在、「フローティングライセンス登録済」のため¥r「処理メニュー」を表示しています。");
    }
    else
    {
        //想定外
        stat = _T("**********¥r　これは、サンプルメッセージです。¥r**********¥r[ID:")
            + strRet + _T("] 認証状況確認中に想定外のエラーが発生しました");
        MessageBox(stat);

        txtInfo2 = _T("現在、「想定外のエラー」のため¥r「処理メニュー」は無効になっています。");
    }

    UpdateData(FALSE);


}
```

以下では諸々の具体的な使用法の例を紹介します。SamplePackageApp.FW45サンプルでは、Dlg1ダイアログの各ボタンクリックで実行します。

次の例では、認証状況表示ダイアログを呼び出しています。

```
// 認証状況表示
void CDlg1::OnBnClickedButton10()
{
    if (theApp.m_pActivation->ActivateStatusDisp() == VARIANT_FALSE)
        MessageBox(_T("エラー"), _T("ライセンス管理"));
}
```

以下の例では、認証の登録や解除に関する各処理を呼び出しています。

```
//認証登録/インターネット
void CDlg1::OnBnClickedButton1()
{
    if (theApp.m_pActivation->ActivateRegisterInternet() == VARIANT_FALSE)
        MessageBox(_T("エラー"), _T("ライセンス管理"));
}

//認証解除/インターネット
void CDlg1::OnBnClickedButton2()
{
    if (theApp.m_pActivation->ActivateRemoveInternet() == VARIANT_FALSE)
        MessageBox(_T("エラー"), _T("ライセンス管理"));
}

//認証登録/電話
void CDlg1::OnBnClickedButton3()
{
    if (theApp.m_pActivation->ActivateRegisterTelephone() == VARIANT_FALSE)
        MessageBox(_T("エラー"), _T("ライセンス管理"));
}

//認証解除/電話
void CDlg1::OnBnClickedButton4()
{
    if (theApp.m_pActivation->ActivateRemoveTelephone() == VARIANT_FALSE)
        MessageBox(_T("エラー"), _T("ライセンス管理"));
}

//代理認証登録/準備
void CDlg1::OnBnClickedButton5()
{
    if (theApp.m_pActivation->ProxyActivateRegisterPrepare() == VARIANT_FALSE)
        MessageBox(_T("エラー"), _T("ライセンス管理"));
}

//代理認証登録/確定
void CDlg1::OnBnClickedButton6()
{
    if (theApp.m_pActivation->ProxyActivateRegisterFix() == VARIANT_FALSE)
        MessageBox(_T("エラー"), _T("ライセンス管理"));
}

//代理認証解除/準備
void CDlg1::OnBnClickedButton7()
{
    if (theApp.m_pActivation->ProxyActivateRemovePrepare() == VARIANT_FALSE)
        MessageBox(_T("エラー"), _T("ライセンス管理"));
}

//代理有効期限更新/準備
void CDlg1::OnBnClickedButton8()
{
    if (theApp.m_pActivation->PreparationOfProxyUpdateOfExpirationDate() == VARIANT_FALSE)
        MessageBox(_T("エラー"), _T("ライセンス管理"));
}

//代理有効期限更新/確定
void CDlg1::OnBnClickedButton9()
```

```
{
    if (theApp.m_pActivation->DeterminationOfProxyUpdateOfExpirationDate() == VARIANT_FALSE)
        MessageBox(_T("エラー"), _T("ライセンス管理"));
}

//認証状況オンライン表示
void CDlg1::OnBnClickedButton11()
{
    if (theApp.m_pActivation->ActivateStatusOnlineVerify() == VARIANT_FALSE)
        MessageBox(_T("エラー"), _T("ライセンス管理"));
}

//認証オンラインステータス
void CDlg1::OnBnClickedButton12()
{
    long ret = theApp.m_pActivation->ActivateStatusCheckOnline();

    CString strRet;
    strRet.Format(_T("%d"), ret);

    CString msg = _T("戻り値は「") + strRet + _T("」でした。¥r¥r")
        + _T("    0:PCレベルで認証されていない（PCのレジストリには認証登録情報がない）¥r")
        + _T("    1:OK（PCレベルとデータベースで認証登録情報が一致した）¥r")
        + _T("    2:NG（PCレベルと一致する認証登録情報がデータベースにない）¥r")
        + _T("    3:NG（認証済ハードウェア情報不一致）¥r")
        + _T("    4:NG（日付データの取得失敗（猶予））¥r")
        + _T("    6:NG（日付データの取得失敗（有効期限））¥r")
        + _T("  -11:接続できない（認証時は電話）¥r")
        + _T("  -12:接続できない（認証時は代理）¥r")
        + _T(" -999:その他エラー（接続できないなど）¥r")
        + _T("   -1：エラー（未設定や範囲を超えているプロパティがある）¥r")
        + _T("   -3：PCの日付が変更されました。¥r");
        + _T("  600：フローティングライセンス登録済");

    MessageBox(msg, _T("【認証状態オンラインステータスメソッド】"));
}

//有効期限の更新
void CDlg1::OnBnClickedButton13()
{
    if (theApp.m_pActivation->UpdateOfExpirationDate() == VARIANT_FALSE)
        MessageBox(_T("エラー"), _T("ライセンス管理"));
}

//「アプリ起動日」を削除
void CDlg1::OnBnClickedButton14()
{
    if (theApp.m_pActivation->RunNR2AppDateRemove() != VARIANT_FALSE)
        MessageBox(_T("「アプリ起動日」を削除しました。"), _T("ライセンス管理"), MB_OK);
    else
        MessageBox(_T("エラー"), _T("ライセンス管理"));
}

//「猶予日数」の「開始日」を削除
void CDlg1::OnBnClickedButton15()
{
    if (theApp.m_pActivation->TrialStartDateRemove() != VARIANT_FALSE)
        MessageBox(_T("「猶予日数」の「開始日」を削除しました。"), _T("ライセンス管理"), MB_OK);
    else
```

```
        MessageBox(_T("エラー"), _T("ライセンス管理"));
}

//フローティングライセンス使用開始
void CDlg1::OnBnClickedButton16()
{
    if (theApp.m_pActivation->FloatingLicenseStart() == VARIANT_FALSE)
        MessageBox(_T("エラー"), _T("ライセンス管理"));
}

//フローティングライセンス使用終了
void CDlg1::OnBnClickedButton17()
{
    if (theApp.m_pActivation->FloatingLicenseFinish() == VARIANT_FALSE)
        MessageBox(_T("エラー"), _T("ライセンス管理"));
}

//認証登録状態回復メソッド
void CDlg1::OnBnClickedButton18()
{
    if (theApp.m_pActivation->RestoreRegisterStatus() == VARIANT_FALSE)
        MessageBox(_T("エラー"), _T("ライセンス管理"));
}

//認証解除状態回復メソッド
void CDlg1::OnBnClickedButton19()
{
    if (theApp.m_pActivation->RestoreCancelStatus() == VARIANT_FALSE)
        MessageBox(_T("エラー"), _T("ライセンス管理"));
}
```

■認証 UI ライブラリ（DLL）の配布

・配布が必要なファイル

お客様（エンドユーザ）向けに配布するアプリケーションとともに次のファイルを配布する必要があります。この場合のアプリケーションは、認証 UI ライブラリ（DLL）を使用したアプリケーションを指します。

Newtone.NR.FW45.dll（.NET Framework 4.5～4.8 アプリケーション用）
Newtone.NR.NET6.dll（.NET 6/7/8 アプリケーション用）

NewtoneNRDvcpp.dll（Visual C++で作成したアプリケーションの配布時）
NewtoneNRDvcpp.tlb（Visual C++で作成したアプリケーションの配布時）

これらは、認証レスキュー！の「認証 UI ライブラリ」をインストールしてインストール先がデフォルトの場合、次のフォルダ内にあります。

<32bitOS の場合> C:¥Program Files¥Newtone¥NRD¥NRDDLL¥NRDLL
<64bitOS の場合> C:¥Program Files (x86)¥Newtone¥NRD¥NRDDLL¥NRDLL

・お客様（エンドユーザ）PC 上での配置

認証 UI ライブラリ（DLL）をお客様（エンドユーザ）向けに配布するアプリケーションとともに配布する場合、インストーラなどでお客様（エンドユーザ）の PC 上でアプリケーション実行時にパスが通るように配置してください。
配布するアプリケーションと同じフォルダに配置するのが最も簡単な方法です。

＜配布するアプリケーションが Visual C++で作成したアプリケーションの場合＞
インストーラなどでお客様（エンドユーザ）の PC 上で DLL の登録も必要です。
また、Visual C++で作成するアプリケーション内で、VC++用のタイプライブラリ（NewtoneNRDvcpp.tlb）の Path をコードで指定する必要があります。
詳しくは、
SampleProject¥VC++フォルダまたは SampleProject_API¥VC++フォルダの
【認証 UI ライブラリ（DLL）を VC++から利用する方法】.txt
をご覧ください。

●認証レスキュー！で使う主なテーブルの概要

ここでは、「認証レスキュー！」で扱う SQL Server のテーブルを確認します。
テーブルは、以下の 3 種類について説明します。

・＜認証キーテーブル＞
・＜認証データテーブル＞
・＜認証ログテーブル＞

以降でそれぞれを詳しく見て行きましょう。

＜認証キーテーブル＞

このテーブルは、ライセンス認証のキーとなるテーブルで、出荷する（アプリケーション）製品１本毎に 1 レコードが追加されます。
たとえば、出荷する製品の 1 本が 4 ライセンス製品であれば、ライセンス数には 4 と入力します。
実際のレコード追加や削除は、社内システムの「認証キー作成」および「認証キー削除」処理で行います。

＜認証キーテーブルの例＞

製品の定義例		プロダクト ID	シリアル No.	ライセンス数	プラス許可数
製品名	ライセンス				
AAA システム Ver.2.0	1 ライセンス	00001-00002-00001	A01-0001	1	0
〃	〃	〃	A01-0002	1	0
〃	〃	〃	A01-0003	1	0
〃	4 ライセンス	00001-00002-00004	A04-0001	4	0
BBB システム Ver.1	1 ライセンス	00002-00001-00001	B01-0001	1	0
〃	〃	〃	B01-0002	1	0
〃	10 ライセンス	00002-00001-00010	B10-0001	10	0

・プラス許可数
これはライセンス認証登録済みのエンドユーザの PC がクラッシュなどに見舞われ、エンドユーザの PC ではライセンス認証解除をできないため、OS を再セットアップした PC や別の PC に、再度アプリケーションのライセンス認証登録ができない場合に使う特別な項目です。
初期値は 0 です。「認証レスキュー！」では、処理の入力項目ではありません。
認証業務用社内 PC の「認証管理システム」の「電話認証解除の対応」処理で、「クラッシュ」オプションを指定して認証解除を行った場合にこの「プラス許可数」項目が+1 されます。
この項目の必要性は、後で説明します。

＜認証データテーブル＞

このテーブルは、出荷した製品をエンドユーザがライセンス認証登録を行って成功した場合に 1 レコード追加されます。このテーブルのプロダクト ID とシリアル No.が同一なレコード数がそのまま現在のライセンス数になります。

たとえば、このテーブルに同一の「プロダクト ID とシリアル No.」の組み合わせで 3 レコード存在する

場合は、現在 3 ライセンス分の認証が登録済みであることになります。

「認証レスキュー！」のライセンス認証のロジックでは、そのことを使用してライセンス数の上限を、前述の＜認証キーテーブル＞に登録した「ライセンス数」項目（と「プラス許可数」項目を足したもの）と比較しチェックしています。

このテーブルへの実際のレコード追加と削除は、次のタイミングで行われます。
インターネットで認証の場合、エンドユーザ PC で実行されるパッケージに組み込まれたプログラムからの呼び出しによる Web サービス内で行われます。
また、電話によるライセンス認証では社内システムの「電話認証登録の対応」、および「電話認証解除の対応」処理で行われます。

＜認証データテーブルの例＞

製品の定義例		プロダクト ID	シリアル No.	認証 ID	ライセンスキー
製品名	ライセンス				
AAA システム Ver.2.0	1 ライセンス	00001-00002-00001	A01-0001	11111-1111	**********
〃	〃	〃	A01-0002	22222-22222	**********
〃	〃	〃	A01-0003	33333-33333	**********
〃	4 ライセンス	00001-00002-00004	A04-0001	44444-44444	**********
〃	〃	〃	〃	55555-55555	**********
〃	〃	〃	〃	66666-66666	**********
BBB システム Ver.1	1 ライセンス	00002-00001-00001	B01-0001	77777-77777	**********
〃	10 ライセンス	00002-00001-00010	B10-0001	88888-88888	**********
〃	〃	〃	〃	99999-99999	**********

4 ライセンス分の 3 ライセンスが既に認証登録されている

・認証 ID

5 桁 － 5 桁 （数字のみ）
エンドユーザ PC 側の認証登録処理時に自動的に発生します。

・ライセンスキー

15 桁 （数字のみ）
上記の「認証 ID」とプロダクト ID、シリアル No.をもとに生成されます。

このライセンスキーは、インターネットでの認証の場合は、Web サービスで生成後エンドユーザ PC 側の認証登録処理（のプログラム）に返され、プログラムが自動的に処理します。
また、電話での認証登録時はオペレータに電話で伝えられたライセンスキーをエンドユーザが画面で入力して登録します。

ここでは、＜認証キーテーブル＞と＜認証データテーブル＞の両方のテーブルの項目説明が終わったところで、両テーブルに密接に関連する「プラス許可数」項目について説明します。

＜認証キーテーブル＞の「プラス許可数」項目の必要性

通常、ライセンス認証登録済みの PC がクラッシュすると、エンドユーザは製品に添付されているプロダクト ID とシリアル No.しかわかりません。認証解除には認証 ID も必要です。

たとえば、製品が 4PC ライセンスといった複数ライセンスの場合、プロダクト ID+シリアル No.は同一ですからこの情報だけでは、4 台中でクラッシュした個体 PC1 台を特定できません。

この場合、＜認証データテーブル＞には、同一のプロダクト ID+シリアル No.を持つレコードは最大で 4 レコード存在することになりますが、認証 ID やライセンスキーが分からないのでどのレコードがクラッシュした PC 分なのか判定できません。

＜認証キーテーブル＞

製品の定義例		プロダクト ID	シリアル No.	ライセンス数	プラス許可数
製品名	ライセンス				
AAA システム Ver.2.0	1 ライセンス	00001-00002-00001	A01-0001	1	0
〃	4 ライセンス	00001-00002-00004	A04-0001	4	0
BBB システム Ver.1	1 ライセンス	00002-00001-00001	B01-0001	1	0

＜認証データテーブル＞

製品の定義例		プロダクト ID	シリアル No.	認証 ID	ライセンスキー
製品名	ライセンス				
AAA システム Ver.2.0	1 ライセンス	00001-00002-00001	A01-0001	11111-1111	******** *******
〃	4 ライセンス	00001-00002-00004	A04-0001	44444-44444	******** *******
〃	〃	〃	〃	55555-55555	******** *******
〃	〃	〃	〃	66666-66666	******** *******
〃	〃	〃	〃	12345-12345	******** *******
BBB システム Ver.1	1 ライセンス	00002-00001-00001	B01-0001	77777-77777	******** *******

認証 ID が分からないと、これら 4 レコード分のどの PC 分（レコード）がクラッシュしたのか分からない

また消去法で考えたとして、クラッシュしていない残りの 3 台すべての PC の認証情報をエンドユーザから聞き出し煩雑な対応をすることは、エンドユーザはもちろん貴社にとっても得策ではありませんし、仮に 50 ライセンスだったらどうでしょう？ PC49 台分の認証情報をエンドユーザから教えてもらわねばなりません。

そこでこれらの解決方法として、前述の通り社内システムの「電話認証解除の対応」処理で、「クラッシュ」オプションを指定して認証解除を行い「プラス許可数」項目が+1 されると、「ライセンス数」+「プラス許可数」の合計が最大ライセンス数としてみなされ、未登録のライセンス数に 1 ライセンス分の猶予ができるわけです。

＜認証キーテーブル＞

製品の定義例		プロダクト ID	シリアル No.	ライセンス数	プラス許可数
製品名	ライセンス				
AAA システム Ver.2.0	1 ライセンス	00001-00002-00001	A01-0001	1	0
〃	4 ライセンス	00001-00002-00004	A04-0001	4	1
BBB システム Ver.1	1 ライセンス	00002-00001-00001	B01-0001	1	0

合計 5 ライセンス（＜認証 データ テーブル＞上での 5 レコード）まで登録可能となる

＜認証データテーブル＞

製品の定義例		プロダクト ID	シリアル No.	認証 ID	ライセンスキー
製品名	ライセンス				
AAA システム Ver.2.0	1 ライセンス	00001-00002-00001	A01-0001	11111-1111	******** *******
〃	4 ライセンス	00001-00002-00004	A04-0001	44444-44444	******** *******
〃	〃	〃	〃	55555-55555	******** *******
〃	〃	〃	〃	66666-66666	******** *******
〃	〃	〃	〃	12345-12345	******** *******
〃	〃	〃	〃	xxxxx-xxxxx	－
BBB システム Ver.1	1 ライセンス	00002-00001-00001	B01-0001	77777-77777	******** *******

その際、クラッシュした PC 分の＜認証データテーブル＞内のレコードは、以降活用されることのないデッドレコードとなりますが、システム上の問題はありません。

＜認証ログテーブル＞

このテーブルには、ライセンス認証に関わる様々な処理結果がログとして 1 レコードずつ記録されます。
このテーブルの内容は、社内システムの「ログの表示」処理で確認、また必要に応じて削除することができます。

＜認証ログテーブルの例＞

自動No.	日時	処理	ステータス	認証区分	メモ	プロダクトID	シリアルNo.	認証ID	MACアドレス1	CPU情報
1	2014/02/24 15:38:00	1:キー作成	1:正常登録	1:オンライン		00001-00001-00001	A0000001		E840F260C430	Intel(R) Core(TM) i3-212
2	2014/02/24 15:38:00	1:キー作成	1:正常登録	1:オンライン		00001-00001-00001	A0000002		E840F260C430	Intel(R) Core(TM) i3-212
3	2014/02/24 15:38:00	1:キー作成	1:正常登録	1:オンライン		00001-00001-00001	A0000003		E840F260C430	Intel(R) Core(TM) i3-212
4	2014/02/24 15:39:00	1:キー作成	1:正常登録	1:オンライン		00001-00001-00001	A0000004		E840F260C430	Intel(R) Core(TM) i3-212
5	2014/02/24 15:39:00	1:キー作成	1:正常登録	1:オンライン		00001-00001-00001	A0000005		E840F260C430	Intel(R) Core(TM) i3-212
6	2014/03/24 15:42:00	3:認証登録	1:正常登録	1:オンライン		00001-00001-00001	A0000001	50533-21818	E840F260C430	Intel(R) Core(TM) i3-212
7	2014/03/24 15:42:00	3:認証登録	1:正常登録	1:オンライン		00001-00001-00001	A0000003	17958-26503	E840F260C430	Intel(R) Core(TM) i3-212
8	2014/03/24 15:42:00	3:認証登録	1:正常登録	1:オンライン		00001-00001-00001	A0000002	78359-54685	E840F260C430	Intel(R) Core(TM) i3-212
9	2014/03/25 18:17:00	1:キー作成	1:正常登録	1:オンライン		12345-12345-12345	1234ABCD		E840F260C430	Intel(R) Core(TM) i3-212
10	2014/03/27 10:47:13	3:認証登録	1:正常登録	1:オンライン		12345-12345-12345	1234ABCD	25672-67548	E840F260C430	Intel(R) Core(TM) i3-212
11	2014/03/27 10:47:18	4:認証解除	1:正常解除	1:オンライン		12345-12345-12345	1234ABCD	25672-67548	E840F260C430	Intel(R) Core(TM) i3-212

・自動 No.

レコード追加時に自動的に付番されます。
初期値（シード）、増分（インクリメント）ともに 1 です。

・日時

ログが記録された日時です。

・処理区分

ログが書き込まれる際に、行われていた処理を示します。

・ステータスフラグ

ログが書き込まれる際に、その処理における結果状態を示します。

「処理区分」と「ステータスフラグ」の定義は次の通りです。

処理区分	ステータスフラグ
1:キー作成	1:正常登録
	2:重複データ
	3:何らかの原因
	11:同時実行違反
2:キー削除	1:正常削除
	2:該当キーなし
	3:何らかの原因
3:認証登録	1:正常登録
	2:ライセンス超え
	3:該当キーなし
	4:何らかの原因
	5:重複データ
4:認証解除	1:正常解除
	3:該当データなし
	4:何らかの原因
5:キー作成 （自動ナンバリング）	1:正常登録
	2:重複データ
	3:何らかの原因
	11:同時実行違反
6:キー作成 （表形式）	1:正常登録
	2:重複データ
	3:何らかの原因
	11:同時実行違反

7:キー編集 （表形式）	1:正常登録
	2:重複データ
	3:何らかの原因
	11:同時実行違反
8:キー削除 （選択削除）	1:正常削除
	2:該当キーなし
	3:何らかの原因
9:キー削除 （全行削除）	1:正常削除
	2:該当キーなし
	3:何らかの原因
10:クラッシュ解除	1:正常解除
	2:該当データなし
	3:何らかの原因
11:キー作成 （インポート）	1:正常登録
	2:重複データ
	3:何らかの原因
	11:同時実行違反
12:キー作成 （ランダム生成）	1:正常登録
	2:重複データ
	3:何らかの原因
	11:同時実行違反
13：有効期限の更新	1:正常更新

・メモ

認証キー編集時の検索条件や認証登録時の有効期限などさまざまな情報を記録します。

・MAC アドレス

認証 UI ライブラリ（DLL）の UseMacAddress プロパティ、または APIUseMacAddress プロパティが True に設定されている場合、ログが書き込まれる際に、エンドユーザ PC の MAC アドレスを最大で 5 個記録します。

MAC アドレスは LAN ポートなどのネットワーク機器に固有な物理アドレスです。これを利用することでエンドユーザ PC の不正利用時の認証識別情報の精度を上げます。

・CPU 情報

認証 UI ライブラリ（DLL）の UseCpuInfo プロパティ、または APIUseCpuInfo プロパティが True に設定されている場合、ログが書き込まれる際に、エンドユーザ PC の CPU 情報を記録します。

これを利用することでエンドユーザ PC の不正利用時の認証識別情報の精度を上げます。

・IP アドレス

ログが書き込まれる際にエンドユーザ PC 側の（グローバル）IP アドレスを記録します。

なお、ローカル PC で Web サービスを実行している場合などの localhost を表す IP アドレスは、IPv4 で″127.0.0.1″、IPv6 で″::1″が記録されます。

・PC 名

エンドユーザ PC の PC 名が記録されます。

●アプリケーション難読化の必要性

ここでは、貴社が考慮するべき大変重要な事項について記載いたします。
それは、.NET アプリケーションの難読化の必要性です。

.NET 用に作成したアプリケーションのアセンブリ(.EXE や.DLL など)は、中間言語(IL)で作成されているので、簡単にリバースエンジニアリングをされてしまいます。
.NET アプリケーションの逆コンパイルは、無償の逆コンパイラなどを利用していとも簡単に実行できます。逆コンパイルにより、単にコードの内容を知られるだけにとどまらず、パスワードなどのログイン情報や暗号化情報などの文字列も明らかになります。これらを放置するとその危険性がシステム全体におよび、貴社のソフトウェアビジネスに大きな損害を与えかねません。

これらのことは「認証レスキュー！」に関わらず一般的な問題なのですが、「認証レスキュー！」の場合を考えてみます。

最終的にエンドユーザに配布されるのは「認証レスキュー！」で提供される認証 UI ライブラリ(DLL)とそれを利用して作成した貴社のアプリケーションです。
この内、認証 UI ライブラリ(DLL)は難読化を施した上で弊社(ニュートン)より出荷されています。
しかし、貴社のアプリケーションが Visual Basic(.NET)や C#で作成された場合は貴社サイドで難読化が必要となります。

貴社のアプリケーションを難読化すれば、悪意を持ったエンドユーザ(または第三者)が逆コンパイルしてもパスワードなどのログイン情報や暗号化情報などが露呈することはありません。

不正逆コンパイル対策のために弊社の難読化ツールなどで.NET アプリケーションを「難読化」されることを強く推奨いたします。

「難読化ツール」の詳細につきましては、弊社の「Spices.NET JP」Web サイトをご覧ください。

「認証レスキュー ！.NET」ユーザーズガイド （3.0.3）

2025 年 3 月 17 日　第 4 版発行

NEWTONE
株式会社ニュートン

著者　　株式会社ニュートン
発行所　株式会社ニュートン
　　www.newtone.co.jp