

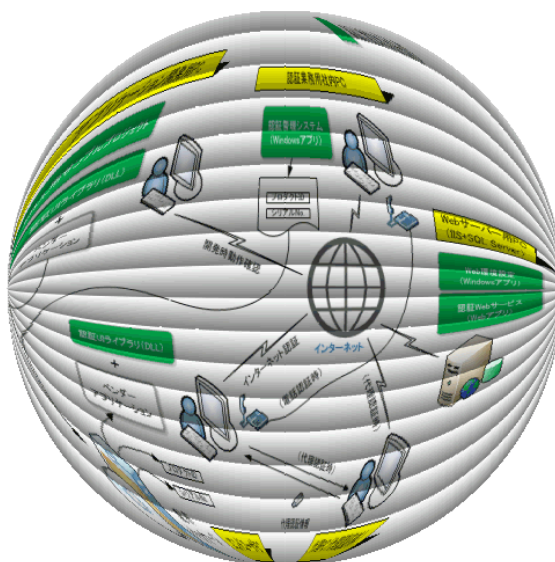
# ライセンス認証(アクティベーション) 実装ソリューション

- ◆ 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 UI ライブラリ(DLL)

## 認証レスキュー! 2

### ユーザーズガイド

(2.2.1)



## - 目次 -

●概要.....	4
●主な注目機能一覧.....	5
●インストールされる内容.....	6
●動作環境.....	6
●運用イメージ.....	7
●処理・設定一覧.....	8
●インストール.....	10
1.「Web サーバー用 PC」へのインストール.....	10
2.「認証業務用社内 PC」へのインストール.....	12
3.「アプリケーション開発用 PC」へのインストール.....	13
4.インストールの終了後.....	13
●初期設定(Web サーバー用 PC と認証業務用社内 PC).....	15
1.Web サーバー用 PC の「環境設定」処理.....	15
■データテーブル新規作成.....	17
■NR 登録ライセンスの管理.....	18
■NR 登録ライセンスの警告.....	21
■「登録」ボタン.....	22
■データベース更新.....	24
■データベースのバックアップ.....	24
■データベースの復元.....	24
2.認証業務用社内 PC の「認証管理システム」.....	25
■環境設定.....	25
■認証キー作成(自動ナンバリング).....	27
■ラベル印刷.....	29
●Web サービスとデータベースを Microsoft Azure で利用する方法.....	32
1.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.....	33
2.認証レスキュー！用のデータベースを Microsoft Azure の「SQL データベース」に配置する.....	33
3.認証レスキュー！の Web サーバー用 PC の「環境設定」処理を完了する.....	37
4.Microsoft Azure の「Web アプリ」に認証レスキュー！の Web サービスを配置する.....	38
5.認証管理システムの「環境設定」処理で Web サービスの URL を設定する.....	39
6.その他の注意点.....	40
●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.....	41
■認証 UI ライブラリ機能一覧.....	41
<クラス>.....	41
<プロパティ>.....	42
<メソッド>.....	49
代理認証機能について.....	59
■認証 UI ライブラリの参照.....	62
Visual Studio 2010(Visual Basic 2010/C# 2010)の場合.....	62
Visual Studio 6.0(Visual Basic 6.0)の場合.....	62
Visual C++の場合.....	64
■認証 UI ライブラリ(DLL)を利用したコーディング.....	66
Visual Basic 2010 の場合.....	66
C# 2010 の場合.....	71
Visual Studio 6.0(Visual Basic 6.0)の場合.....	76
Visual C++の場合.....	81
■認証 UI ライブラリ(DLL)の配布.....	87
●認証レスキュー！で使う主なテーブルの概要.....	88
<認証キーテーブル>.....	88

＜認証データテーブル＞ .....	88
＜認証キーテーブル＞の「プラス許可数」項目の必要性 .....	90
＜認証ログテーブル＞ .....	91
●「認証管理システム」のその他の処理説明 .....	94
■認証キー作成（表形式） .....	95
■認証キー作成（個別） .....	96
■認証キー作成（ランダム生成） .....	97
■認証キー作成（インポート） .....	98
■認証キー編集（表形式） .....	99
■認証キー削除（表形式） .....	101
■認証キー削除（個別） .....	102
■認証状況 .....	103
■ログの表示 .....	105
■電話認証登録の対応 .....	106
■電話認証解除の対応 .....	107
●有効期限機能の利用方法 .....	110
●アプリケーション難読化の必要性 .....	114

## ●概要

「認証レスキュー！2(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 2012 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！

## ●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

### ◆拡張機能

- ・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
- ・マイクロソフト社クラウド Microsoft Azure 対応
- ・代理認証機能
- ・レンタル機能
- ・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

### ◆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！2 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	<ul style="list-style-type: none"> <li>・認証 Web サービス (Web アプリケーション)</li> <li>・Web 環境設定 (Windows アプリケーション)</li> <li>・SQL Server 2012 Express + Management Studio</li> <li>・体験版ライセンス (10 登録ライセンス) ※製品版の基本パックに付属する「500 登録ライセンス」はインストールとは別の形で提供されます。</li> </ul>
認証業務用 PC	<ul style="list-style-type: none"> <li>・認証管理システム (Windows アプリケーション)</li> </ul>
アプリケーション開発用 PC	<ul style="list-style-type: none"> <li>・認証 UI ライブラリ (DLL)、同タイプライブラリ (TLB)</li> <li>・サンプルプロジェクト Visual Basic 2010 用、C# 2010 用、Visual Basic 6.0 用、Visual C++ 2010 用</li> </ul>
ユーザズガイドなど	

## ●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 2012 R2 / 2012 / 2008 R2 / 2008

日本語 Microsoft Windows 10 / 8.1 / 8 / 7 / Vista (x86、x64 対応)

### ・「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 / 7.5 / 7.0

### ・「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 2012 R2 / 2012 / 2008 R2 / 2008

日本語 Microsoft Windows 10 / 8.1 / 8 / 7 / Vista (x86、x64 対応)

日本語 Microsoft Windows Server 2003 R2 / 2003

日本語 Microsoft Windows XP (x86、x64 対応)

### ・「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

日本語 Microsoft Windows Server 2012 R2 / 2012 / 2008 R2 / 2008

日本語 Microsoft Windows 10 / 8.1 / 8 / 7 / Vista (x86、x64 対応)

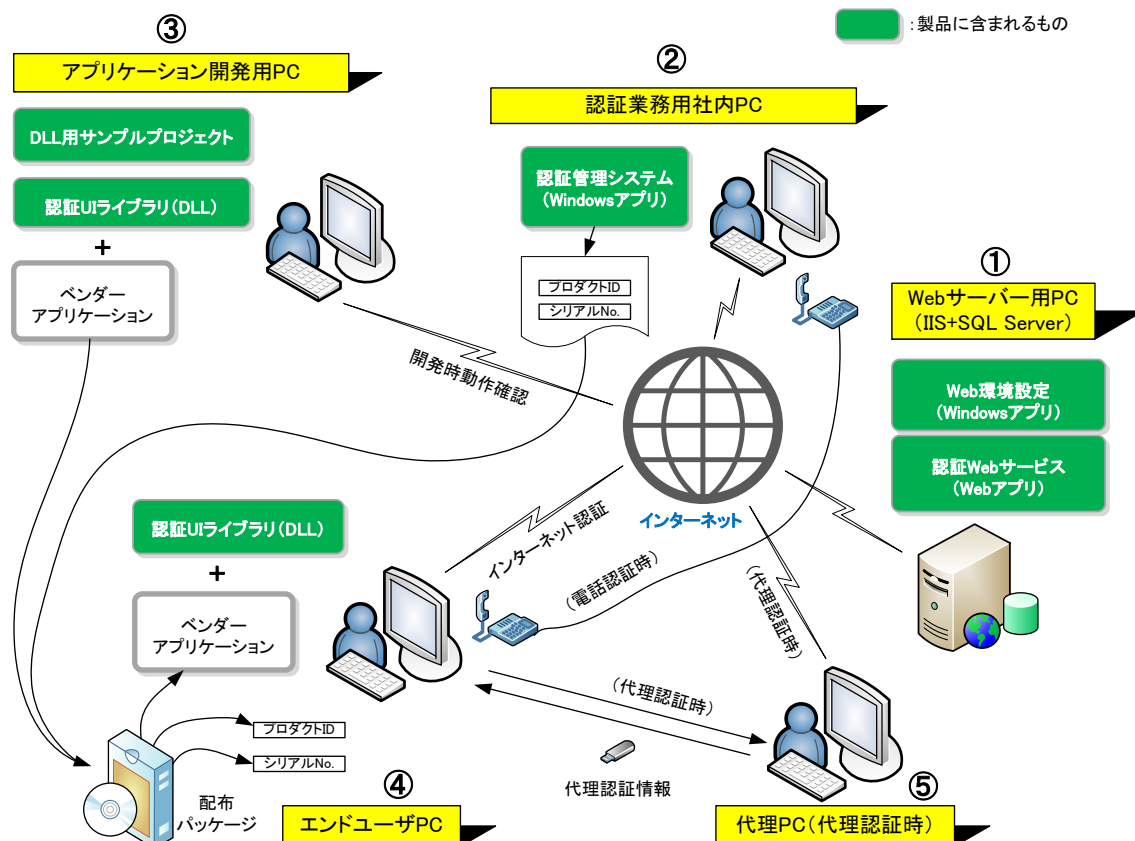
日本語 Microsoft Windows Server 2003 R2 / 2003

日本語 Microsoft Windows XP (x86、x64 対応)

### ・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 2015 / 2013 / 2012 / 2010 / 2008、Visual Studio 6.0

## ●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。

## ●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
  - NR2 規定/NR2 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>

Visual Basic 2010 用

C# 2010 用

Visual Basic 6.0 用

Visual C++ 2010 用

#### <プロパティ>

- ・ベンダアプリケーション開始レジストリキーパス
- ・電話で認証時の電話番号
- ・暗号化時のパスワード
- ・暗号化時の Salt 文字列
- ・猶予(試用)日数
- ・猶予(試用)期間の名称
- ・レンタル日数



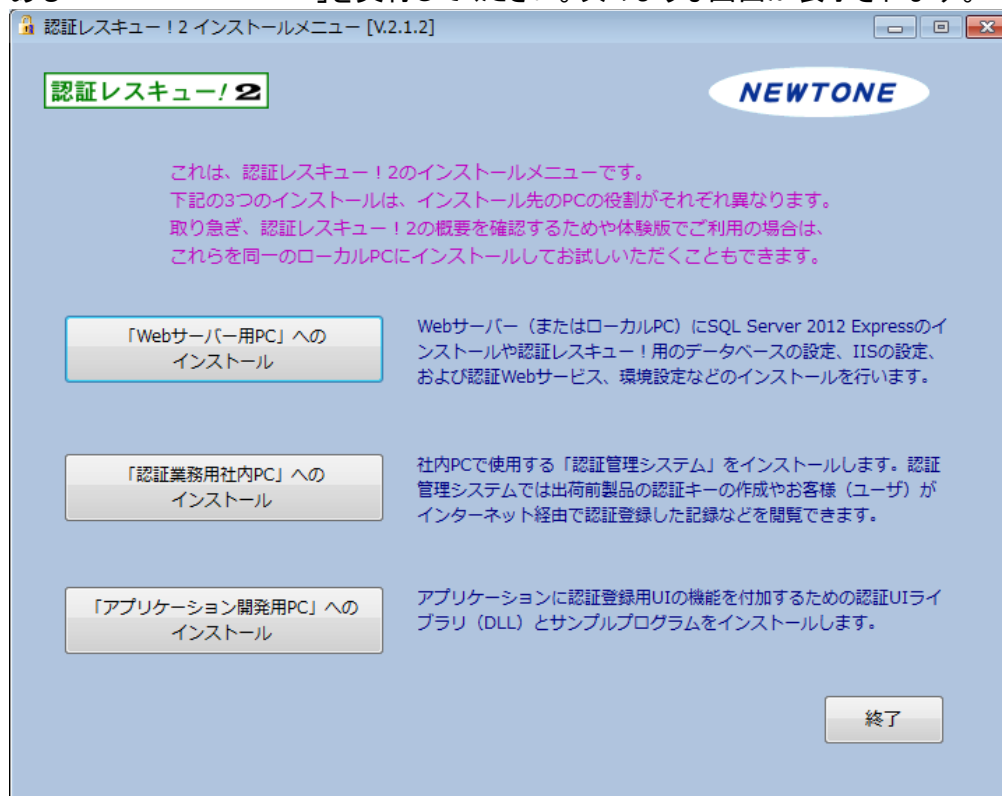
- ・レンタル期間の名称
- ・MAC アドレスの使用
- ・CPU 情報の使用
- ・Web サービスの URL
- ・Web サービス確認パスワード
- ・Web サービスのタイムアウト
- ・Web サービス時の基本認証の使用
- ・Web サービス時の基本認証ユーザ名
- ・Web サービス時の基本認証パスワード
- ・プロダクト ID の桁数
- ・シリアル No.の桁数
- ・認証登録時の設定プロダクト ID
- ・認証登録時の設定シリアル No.

#### <メソッド>

- ・「認証状態確認」機能
- ・「認証状態オンライン確認」機能
- ・「認証状態表示」処理の呼び出し
- ・「認証登録/インターネット」処理の呼び出し
- ・「認証登録/電話」処理の呼び出し
- ・「認証解除/インターネット」処理の呼び出し
- ・「認証解除/電話」処理の呼び出し
- ・「認証登録状態回復」処理の呼び出し
- ・「認証解除状態回復」処理の呼び出し
- ・「有効期限の更新」処理の呼び出し
- ・「代理認証登録/準備」処理の呼び出し
- ・「代理認証登録/確定」処理の呼び出し
- ・「代理認証解除/準備」処理の呼び出し
- ・「代理認証登録/実行」処理の呼び出し
- ・「代理認証解除/実行」処理の呼び出し

## ●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「NR2InstallMenu.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。

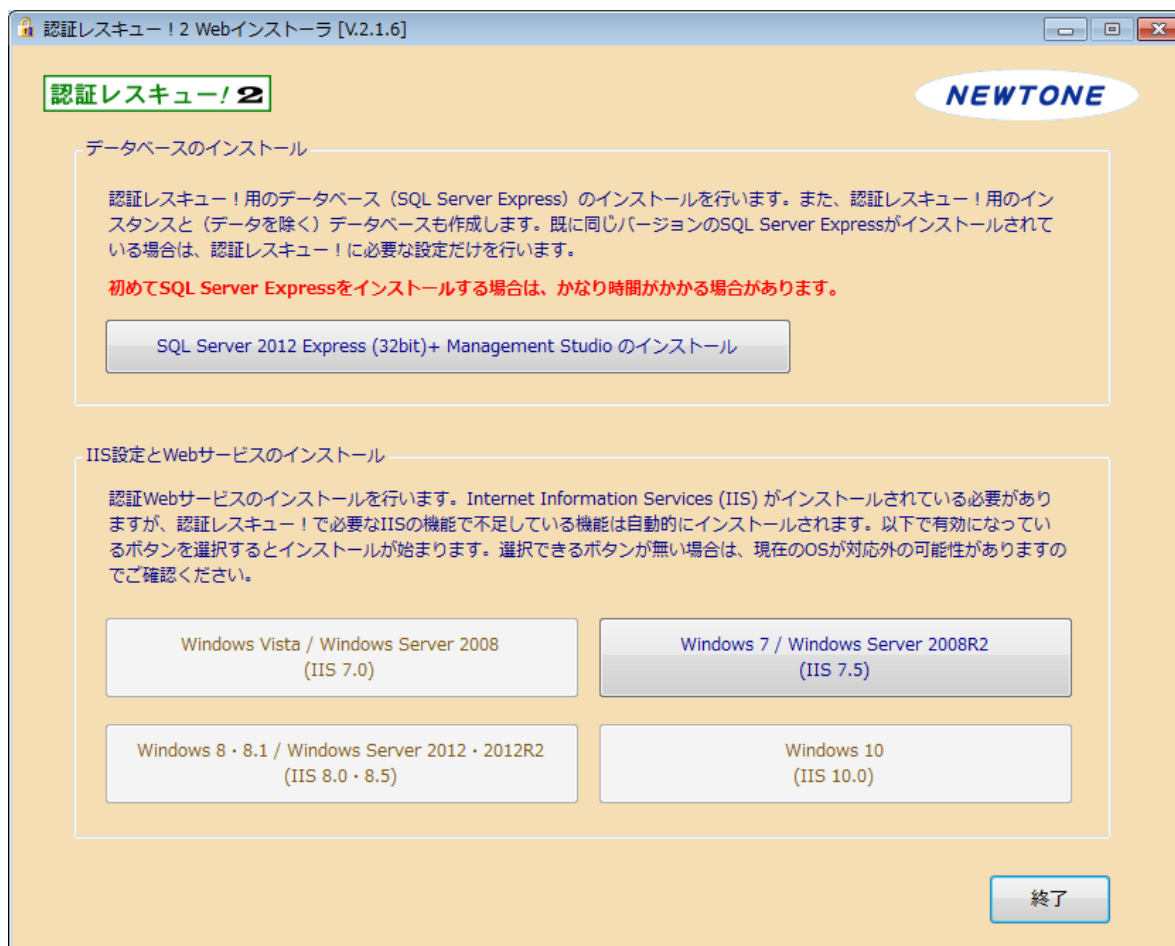
3つのインストールボタンがあり、インストール先のPCの役割がそれぞれ異なります。

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 2012 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。

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。

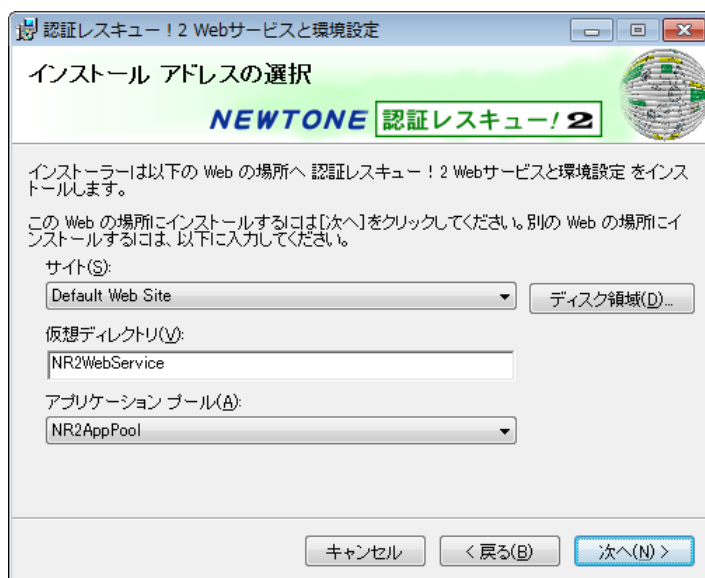
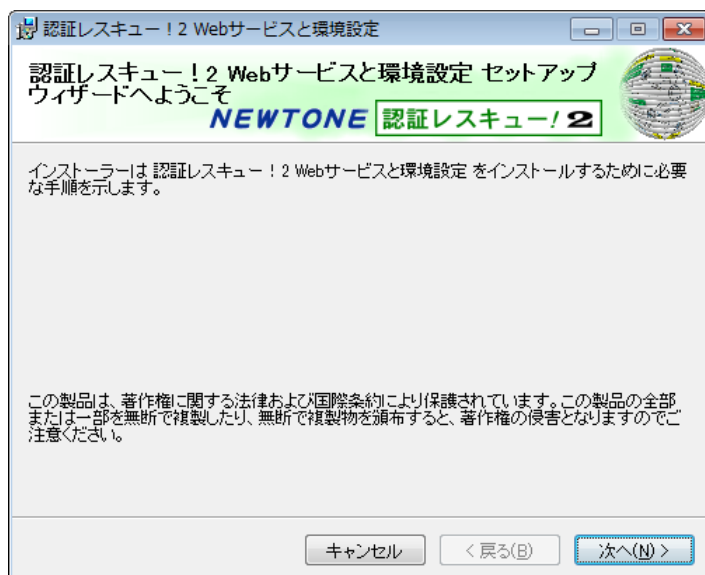
## ・データベースのインストール

認証レスキュー！用のデータベース(SQL Server Express)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) がインストールされている必要がありますが、認証レスキュー！で必要な IIS の機能で不足している機能は自動的にインストールされます。以下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。

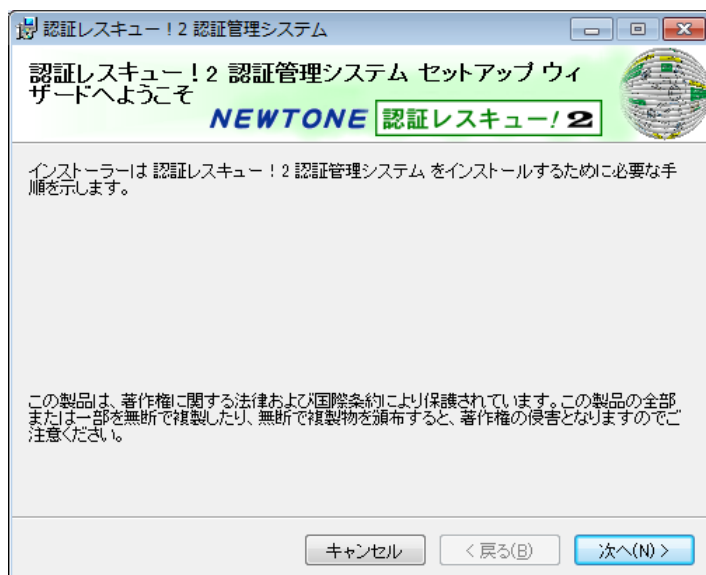


この画面では、サイト、仮想ディレクトリ、アプリケーションプールを指定しますが、通常はデフォルト(初期設定)のままで「次へ」ボタンを押します。以降は画面の指示に従ってください。

マイクロソフト社のクラウドサービス Microsoft Azure の「Web アプリ (旧 Web サイト)」に Web サービスを配置する場合は、一度この「IIS 設定と Web サービスのインストール」をローカル PC にインストールします。その後、ローカル PC の Web サービスのフォルダ内のすべてのファイルと Web サービス環境設定データ(WebServEnv.wai)ファイル(後述)を FTP などを使用して Microsoft Azure の「Web アプリ (旧 Web サイト)」に手動でコピーする必要があります。詳しくは、後述の「Microsoft Azure で利用する方法」をご覧ください。

## 2. 「認証業務用社内 PC」へのインストール

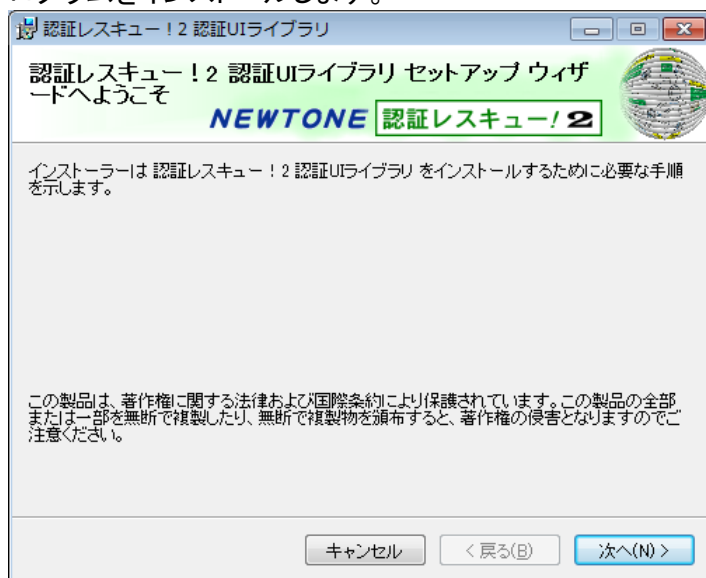
社内 PC で使用する「認証管理システム」をインストールします。認証管理システムでは出荷前製品の認証キーの作成やお客様(エンドユーザ)がインターネット経由で認証登録した記録などを閲覧できます。



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

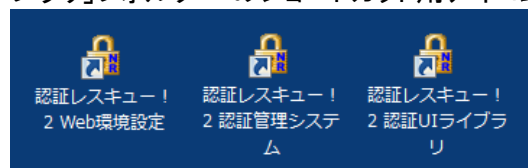
アプリケーションに認証登録用 UI の機能を付加するための認証 UI ライブラリ(DLL)とサンプルプログラムをインストールします。



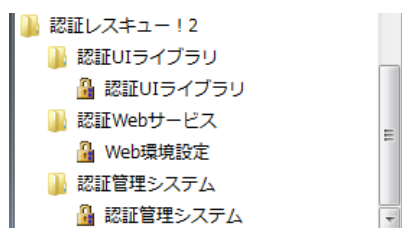
インストールするには画面の指示に従ってください。

### 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！ Web 環境設定」へのショートカット、「認証レスキュー！ 認証管理システム」へのショートカット、「認証レスキュー！ 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。



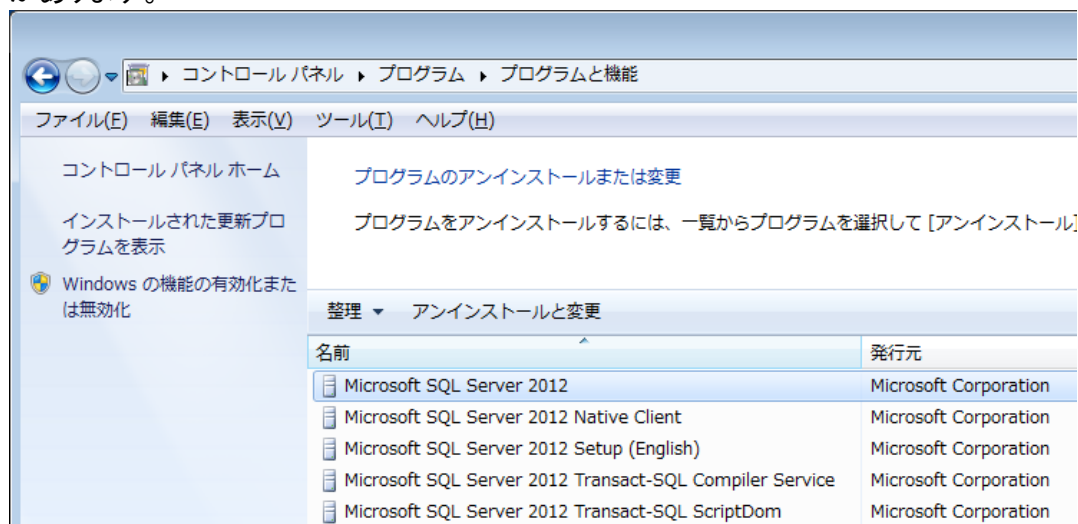
また、プログラムメニューには次のように「認証レスキュー！」が登録されます。



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 2012 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。



## ●初期設定（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！ Web 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\NR2\NR2Web\WebAdmin.exe、

＜64bitOS の場合＞ C:\Program Files (x86)\Newtone\NR2\NR2Web\WebAdmin.exe です。

ここでは、設定が必須で省略できない項目についてだけ説明します。

#### ・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。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。

### NR2 規定

認証レスキュー！（以降 NR2）での規定のデータベースです。実際の運用には通常、この「NR2 規定」を選択します。NR2 では、Web サーバー用 PC への「Web サービスと環境設定」のインストール時に SQL Server 2012 Express のインストールを選択できます。SQL Server 2012 Express をインストールして NR2 のインスタンス(Newtone)を作成し、NR2 規定のデータベース(NR2)を作成します。この段階では、データベース(NR2)にテーブルはまだありません。

社内システム用 PC に「認証管理システム」のインストール後、認証管理システム(InsideSystem.exe)のメニューより「テーブルデータ新規作成」処理を実行することでテーブルが作成されます。

その後は、「認証キー作成」処理で出荷前のパッケージ製品の登録を行うことで認証キーテーブルにデータが格納され、お客様(エンドユーザ)が入手したパッケージ製品の認証登録作業を行うことでインターネットを通して認証データテーブルに認証情報が記録されることになります。

### NR2 サンプルデータベース

認証レスキュー！（以降 NR2）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 NR2 全体を把握したい場合などに選択します。

Web サーバー用 PC へ「Web サービスと環境設定」をインストール、社内システム用 PC に「認証管理システム」をインストールするところまでは、データベース名が「NR2」ではなく、「NR2SAMPLE」になることを除いて上記の「NR2 規定」の場合と同様です。

認証管理システムのインストール後、認証管理システム(InsideSystem.exe)のメニューより「テーブルデータ新規作成」処理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！（以降 NR2）用のデータベースを置く場合やマイクロソフト社のクラウドサービス Microsoft Azure の「Web アプリ（旧 Web サイト）」に Web サービスを配置し、同「SQL データベース」に NR2 データベースを配置する場合に選択します。

Microsoft Azure で利用する場合や Web サーバー用 PC とは異なる(データベース)サーバーで利用する場合の利用方法は後述の説明ページをご覧ください。

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「NR2 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「NR2 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「NR2 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「NR2 サンプルデータベース」の場合

データテーブル新規作成

現在、認証Webサービスの環境設定でデータベースに「NR2サンプルデータベース」が選択されています。  
サンプルデータベースを作成します。  
サンプルデータベースは下記の桁数や許可が固定になっており、指定はできません。

「プロダクトID」と「シリアルNo.」の桁数設定

プロダクトID: 17 ? (最大28桁)

シリアルNo.: 8 ? (最大28桁)

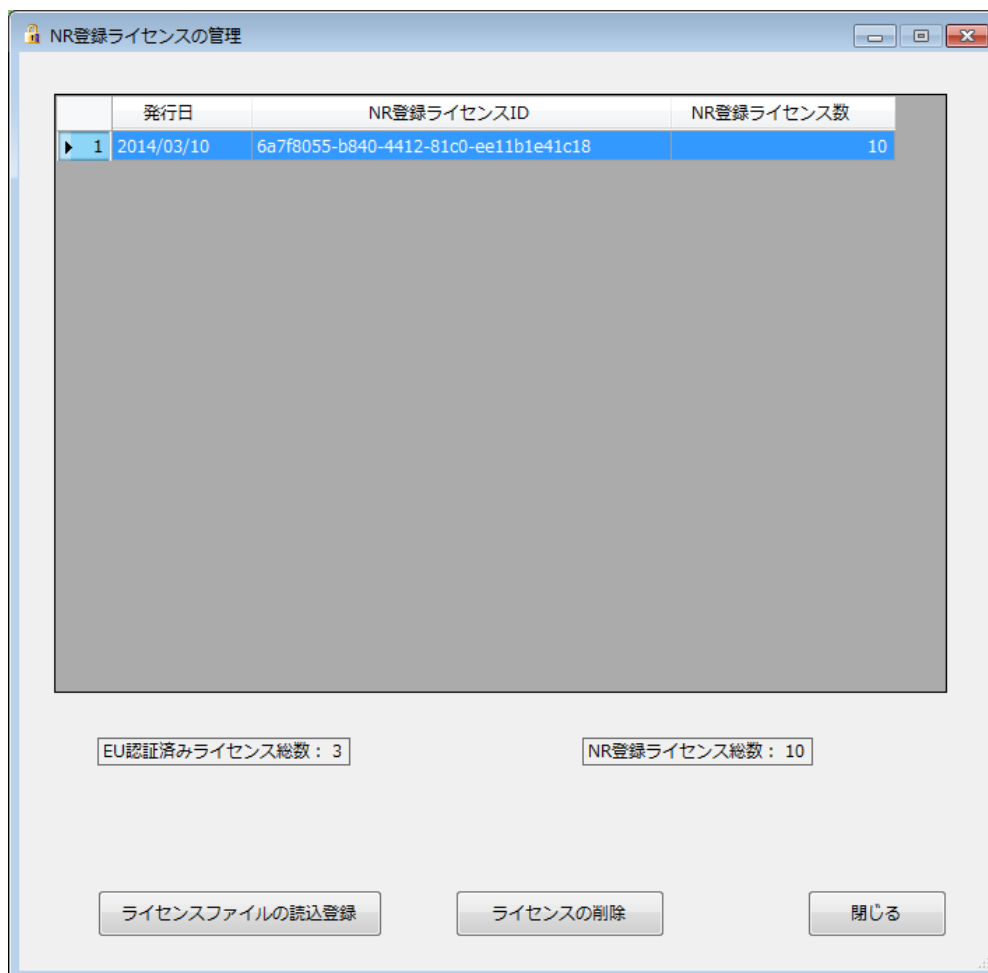
テーブルを新規作成

終了

「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。

#### ■NR 登録ライセンスの管理

環境設定の「NR 登録ライセンスの管理」ボタンを押すと次のような画面が表示されます。



この「ライセンスの管理」処理では、お客様（エンドユーザ、以降 EU）が認証登録してきたライセンス総数の確認と弊社（ニュートン）から入手いただいた NR（認証レスキュー！）登録ライセンス数を管理できます。

「EU 認証済みライセンス総数」に EU が認証登録してきたライセンスの総数が、「NR 登録ライセンス総数」に現在登録されている貴社分の NR 登録ライセンスの総数がそれぞれ表示されます。

「ライセンスファイルの読み登録」ボタンで入手した NR 登録ライセンスを登録します。「ライセンスの削除」ボタンで登録してあった NR 登録ライセンスを削除します。

貴社のパッケージソフトを EU に配布して EU がライセンス登録をした場合、貴社 Web サーバーなどの認証レスキュー！用データベースの認証データテーブルにその登録ライセンスがレコード数として記録されます。

認証レスキュー！の運用では、その「EU 登録済みライセンス」の総数以上の NR 登録ライセンスを貴社にご用意いただく必要があります。

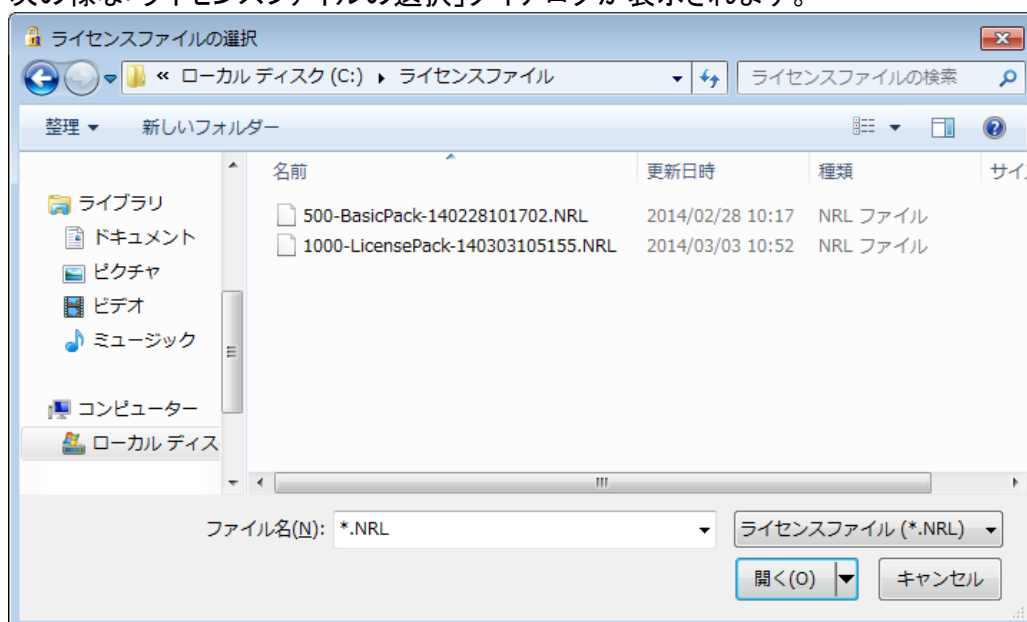
以下に NR 登録ライセンスの一覧表を示します。

＜NR 登録ライセンス一覧＞

種類	提供形態	想定 EU 登録済み ライセンス総数
体験版ライセンス(10 登録ライセンス)	体験版に付属	10 ライセンス分
基本パック(500 登録ライセンス付属)	製品に付属	500 ライセンス分
1,000 登録ライセンスパック	オプション(別売)	1,000 ライセンス分
5,000 登録ライセンスパック	オプション(別売)	5,000 ライセンス分
10,000 登録ライセンスパック	オプション(別売)	10,000 ライセンス分
50,000 登録ライセンスパック	オプション(別売)	50,000 ライセンス分
100,000 登録ライセンスパック	オプション(別売)	100,000 ライセンス分
無制限登録ライセンスパック	オプション(別売)	制限なし

それではここで実際に、「ライセンスファイルの読込登録」ボタンを押して、NR 登録ライセンスを登録します。

次の様な「ライセンスファイルの選択」ダイアログが表示されます。



ここで、貴社が入手した NR 登録ライセンスファイルを指定します。  
基本パックに付属している 500 登録ライセンスファイルであれば、ディスクやメールで入手した 500-BasicPack-xxxxxxxxxxxx.NRL といったファイルを指定します。

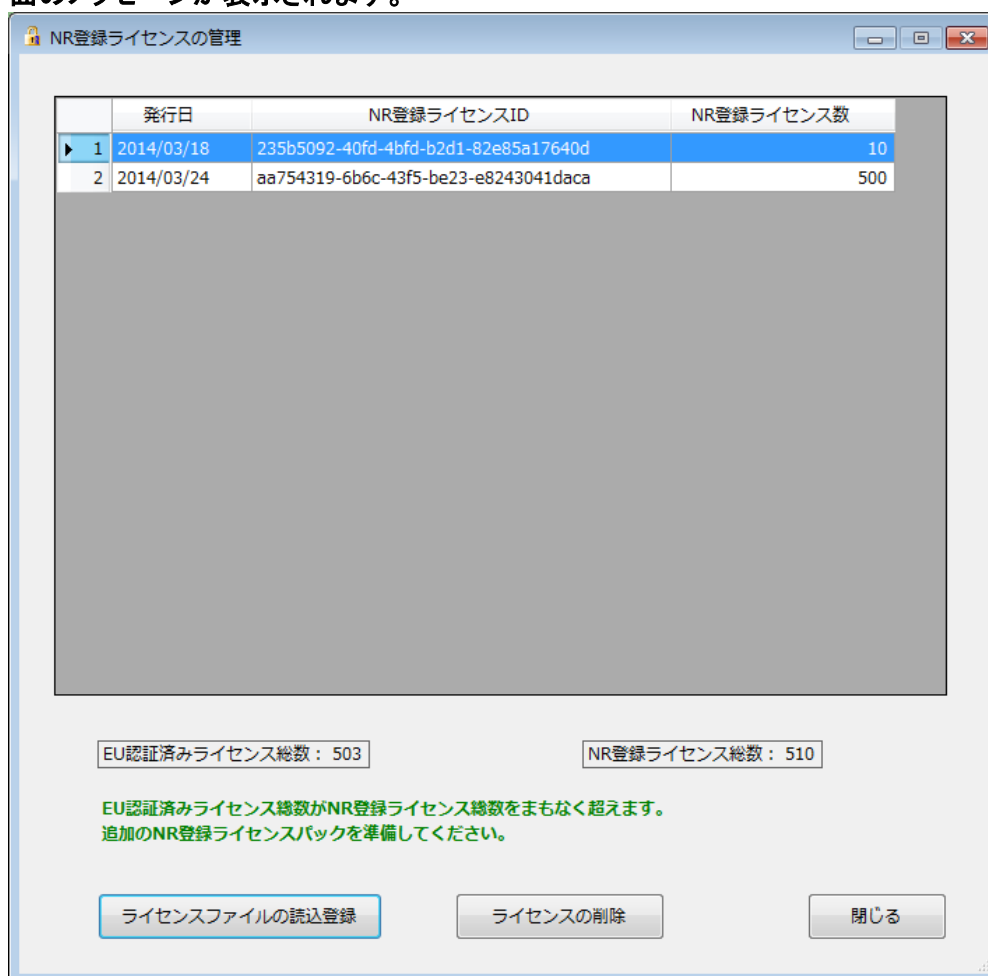
また、体験版や基本パックの場合は、インストール先がデフォルトなら、  
＜32bitOS の場合＞ C:\Program Files\Newtone\NR2\NR2Web  
＜64bitOS の場合＞ C:\Program Files (x86)\Newtone\NR2\NR2Web  
に、「10-TrialLicense-xxxxxxxxxxxx.NRL」といった 10 登録分の NR 登録ライセンスファイルがあるのでそれを利用できます。

このダイアログで「開く」ボタンを押すと指定した NR 登録ライセンスファイルが読み込まれ、「NR 登録ライセンスの管理」画面に反映されます。

## ■NR 登録ライセンスの警告

この「ライセンスの管理」処理で、NR 登録ライセンスに関する警告が表示される場合があります。

[1] NR 登録ライセンス総数が EU 登録済みライセンス総数の 90%を超えた場合に次のような画面のメッセージが表示されます。



この場合、各 PC での処理の動作は次のようになります。

### ・認証業務用社内 PC での「認証管理システム」

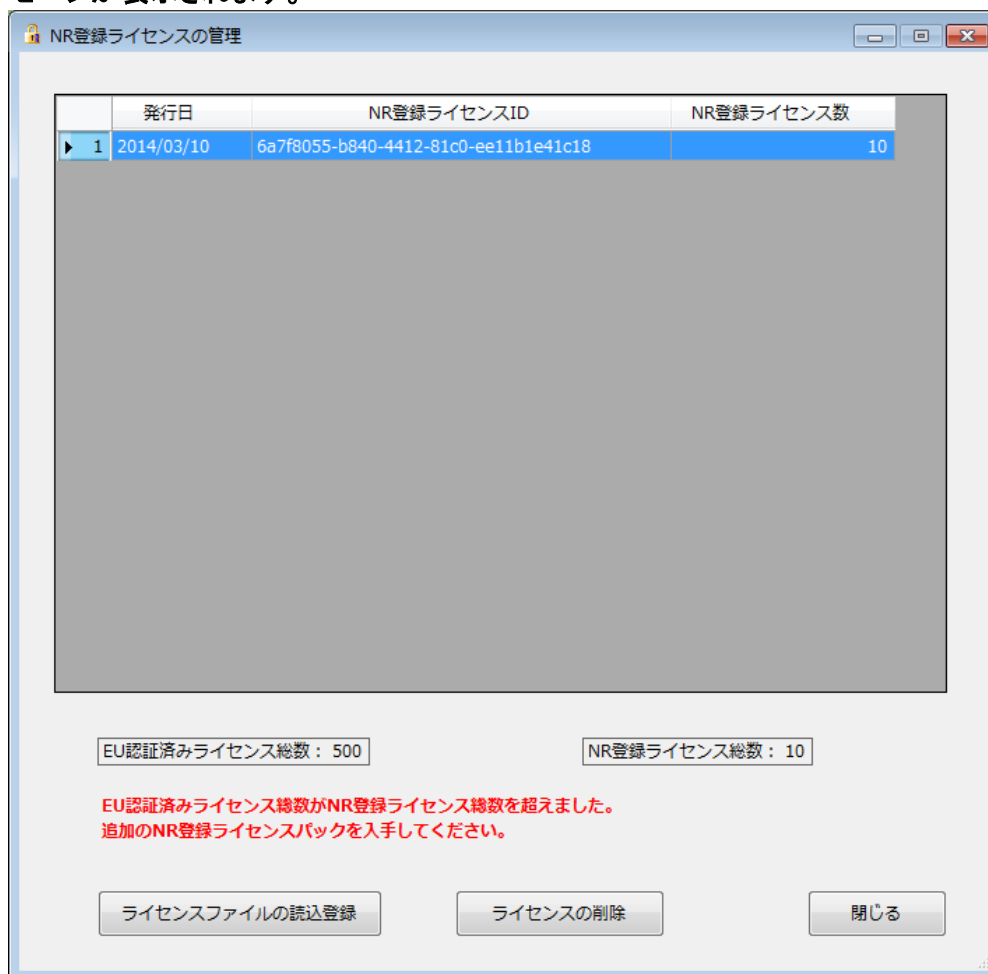
認証レスキュー！用のデータベースのデータアクセス箇所では次のメッセージを表示し処理は続行します。

”EU 認証済みライセンス総数が NR 登録ライセンス総数をまもなく超えます。追加の NR 登録ライセンスパックを準備をしてください。”

### ・認証 UI ライブラリ(DLL)を使用する貴社アプリケーションが動作する EU の PC

認証レスキュー！用のデータベースのデータアクセス箇所ではメッセージを表示せず、処理は続行されます。

[2] NR 登録ライセンス総数が EU 登録済みライセンス総数を超えた場合は次のような画面のメッセージが表示されます。



この場合、各 PC での処理の動作は次のようになります。

・認証業務用社内 PC での「認証管理システム」

認証レスキュー！用のデータベースのデータアクセス箇所では次のメッセージを表示し処理を中断します。

”EU 認証済みライセンス総数が NR 登録ライセンス総数を超えました。追加の NR 登録ライセンスパックを入手してください”

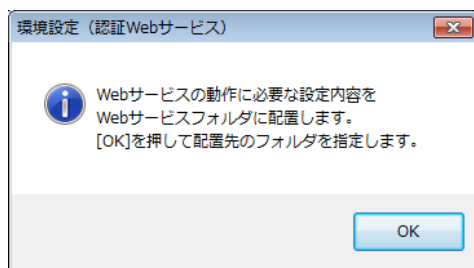
・認証 UI ライブラリ(DLL)を使用する貴社アプリケーションが動作する EU の PC

認証レスキュー！用のデータベースのデータアクセス箇所では次のメッセージを表示し処理を中断します。

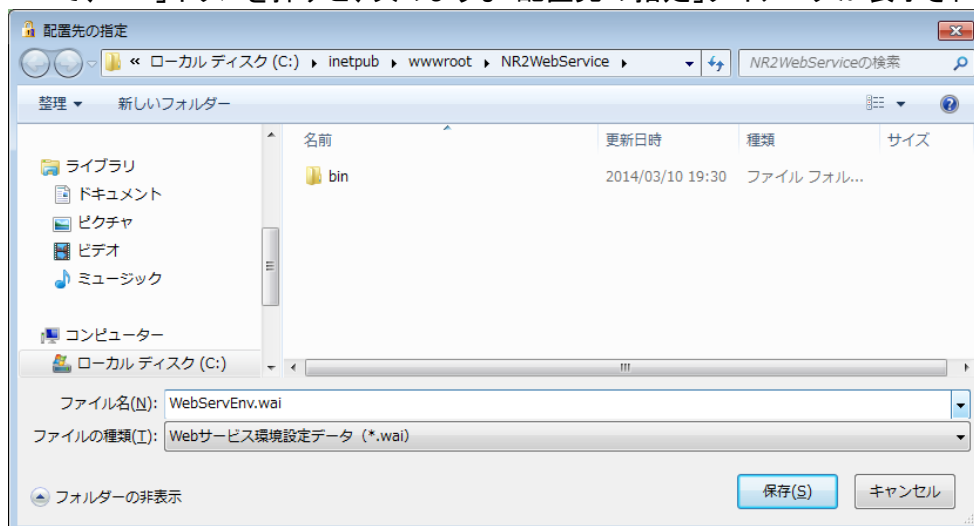
”NR 登録ライセンスに問題があります。アプリケーションベンダにご連絡ください。”

■「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。



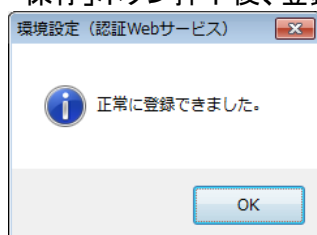
環境設定で入力した情報の内、Web サービスの動作に必要な情報を Web サービス環境設定データ (WebServEnv.wai) ファイルとして、この「配置先の指定」ダイアログで IIS の認証レスキュー！用の Web サービスが配置してあるフォルダにも出力します。通常は、自動的に(例: C:\inetpub\wwwroot\NR2WebService などの)適切な出力先が表示されますので、確認してそのまま「保存」ボタンを押します。

マイクロソフト社のクラウドサービス Microsoft Azure の「Web アプリ (旧 Web サイト)」に Web サービスを配置する場合は、この Web サービス環境設定データ (WebServEnv.wai) ファイルも Web サービスと同じフォルダにコピーする必要があります。この後の「NR 登録ライセンスの管理」でライセンスを登録した場合も Web サービス環境設定データ (WebServEnv.wai) ファイルが更新されますが、更新されるたびに Microsoft Azure の「Web アプリ (旧 Web サイト)」に上書き配置 (コピー) する必要があります。

Microsoft Azure の「Web アプリ (旧 Web サイト)」への配置については詳しくは、後述の「Web サービスとデータベースを Microsoft Azure で利用する方法」をご覧ください。

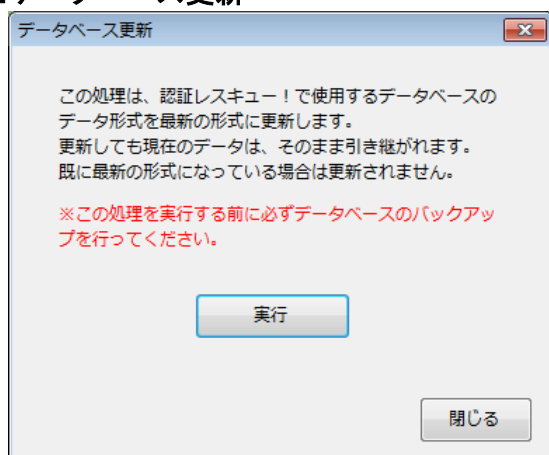
なお、環境設定で入力した情報で Web サービスの動作と関係しない情報は、Web サービス環境設定データ (WebServEnv.wai) ファイルではなく、この「環境設定」を実行している PC のレジストリに記録されます。

「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。



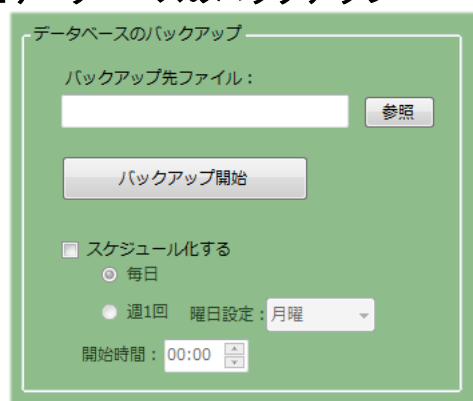
以下は必要に応じて行う処理を説明します。

## ■データベース更新



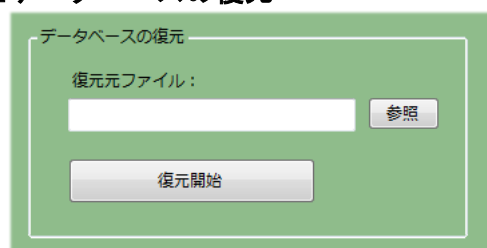
この処理は、認証レスキュー！で使用するデータベースのデータ形式を最新の形式に更新します。更新しても現在のデータは、そのまま引き継がれます。既に最新の形式になっている場合は更新されません。

## ■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！ 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\NR2\NR2InsideSystem\InsideSystem.exe、

＜64bitOS の場合＞ C:\Program Files (x86)\Newtone\NR2\NR2InsideSystem\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。

### ■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。

環境設定 (認証管理システム)

Webサービス

URL:

確認パスワード:

タイムアウト:  秒

☐ 基本認証を使用する

ユーザ名:

パスワード:

Webサービス接続確認

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:

ポート:

ユーザ名:

パスワード:

ログイン設定

☐ 起動時にログインダイアログを表示する

ユーザ名:

パスワード:

☐ レンタル機能を使用する (重要)

登録 終了

ここでは、設定が必須で省略できない項目についてだけ説明します。

#### •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。  
以下にいくつか例を示します。

自 PC のローカルホストの Web サーバー (IIS) にアクセスする例:  
`http://localhost/Nr2WebService/Service.asmx`

自社 Web サーバー (IIS) にアクセスする例:  
`http://www.newtone.co.jp/Nr2WebService/Service.asmx`

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

#### •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 (+-\*/!#\$%&()=¥@<>?)

上記の設定をしたら「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。

「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。

### ■認証キー作成（自動ナンバリング）

次に、認証キーを作成します。3種類の作成方法がありますが、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。

「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。

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 2016/03/28 ~ 2016/03/28

プロダクトID:  ? (未入力:指定なし)

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	日時
▶ 1	00001-00001-00001	A0000001	1	0	<input type="checkbox"/>	2016/05/31	2014/02/24 15:38:00
2	00001-00001-00001	A0000002	1	0	<input type="checkbox"/>	2016/05/31	2014/02/24 15:38:00
3	00001-00001-00001	A0000003	1	0	<input type="checkbox"/>	2016/05/31	2014/02/24 15:38:00
4	00001-00001-00001	A0000004	1	0	<input type="checkbox"/>	2016/05/31	2014/02/24 15:39:00
5	00001-00001-00001	A0000005	1	0	<input type="checkbox"/>	2016/05/31	2014/02/24 15:39:00
6	12345-12345-12345	1234ABCD	5	0	<input checked="" type="checkbox"/>	2018/03/31	2014/03/25 18:17:00

終了

以下に各項目を説明します。

### ・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、無期限で利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

### ■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。

ラベル印刷

作成した認証キーの【プロダクトID】や【シリアルNo.】のラベル印刷を行います。

検索

プロダクトID:  ? (未入力: 指定なし)

シリアルNo.: 先頭指定文字列:  ? (未入力: 指定なし)

検索実行

すべて選択 すべて選択解除 印刷プレビュー 印刷

	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	作成日時
▶ 1	<input checked="" type="checkbox"/>	00001-00001-00001	A00000001	1	0	<input checked="" type="checkbox"/>	2016/05/31	2014/02/24 15:38:00
2	<input type="checkbox"/>	00001-00001-00001	A00000002	1	0	<input type="checkbox"/>	2016/05/31	2014/02/24 15:38:00
3	<input type="checkbox"/>	00001-00001-00001	A00000003	1	0	<input type="checkbox"/>	2016/05/31	2014/02/24 15:38:00
4	<input type="checkbox"/>	00001-00001-00001	A00000004	1	0	<input type="checkbox"/>	2016/05/31	2014/02/24 15:38:00
5	<input type="checkbox"/>	00001-00001-00001	A00000005	1	0	<input type="checkbox"/>	2016/05/31	2014/02/24 15:38:00
6	<input type="checkbox"/>	12345-12345-12345	1234ABCD	5	0	<input checked="" type="checkbox"/>	2017/03/31	2014/03/25 18:17:00

終了

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。

印刷プレビュー

ページ(P) 1

プロダクトID : 00001-00001-00001 シリアルNo. : ABC00001	プロダクトID : 00001-00001-00001 シリアルNo. : ABC00001	プロダクトID : 00001-00001-00001 シリアルNo. : ABC00001	プロダクトID : 00001-00001-00001 シリアルNo. : ABC00001
プロダクトID : 00001-00001-00001 シリアルNo. : ABC00002	プロダクトID : 00001-00001-00001 シリアルNo. : ABC00002	プロダクトID : 00001-00001-00001 シリアルNo. : ABC00002	プロダクトID : 00001-00001-00001 シリアルNo. : ABC00002
プロダクトID : 00001-00001-00001 シリアルNo. : ABC00003	プロダクトID : 00001-00001-00001 シリアルNo. : ABC00003	プロダクトID : 00001-00001-00001 シリアルNo. : ABC00003	プロダクトID : 00001-00001-00001 シリアルNo. : ABC00003
プロダクトID : 00001-00001-00001 シリアルNo. : ABC00004	プロダクトID : 00001-00001-00001 シリアルNo. : ABC00004	プロダクトID : 00001-00001-00001 シリアルNo. : ABC00004	プロダクトID : 00001-00001-00001 シリアルNo. : ABC00004
プロダクトID : 00001-00001-00001 シリアルNo. : ABC00005	プロダクトID : 00001-00001-00001 シリアルNo. : ABC00005	プロダクトID : 00001-00001-00001 シリアルNo. : ABC00005	プロダクトID : 00001-00001-00001 シリアルNo. : ABC00005

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、それぞれの利用目的があるからです。たとえば、次のような用途です。

- 1 枚目：パッケージ内の CD (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

メーカー例：ヒサゴ「A4 タックシール」GB871 など

## ●Web サービスとデータベースを Microsoft Azure で利用する方法

ここでは、「認証レスキュー！」の Web サービスとデータベース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。

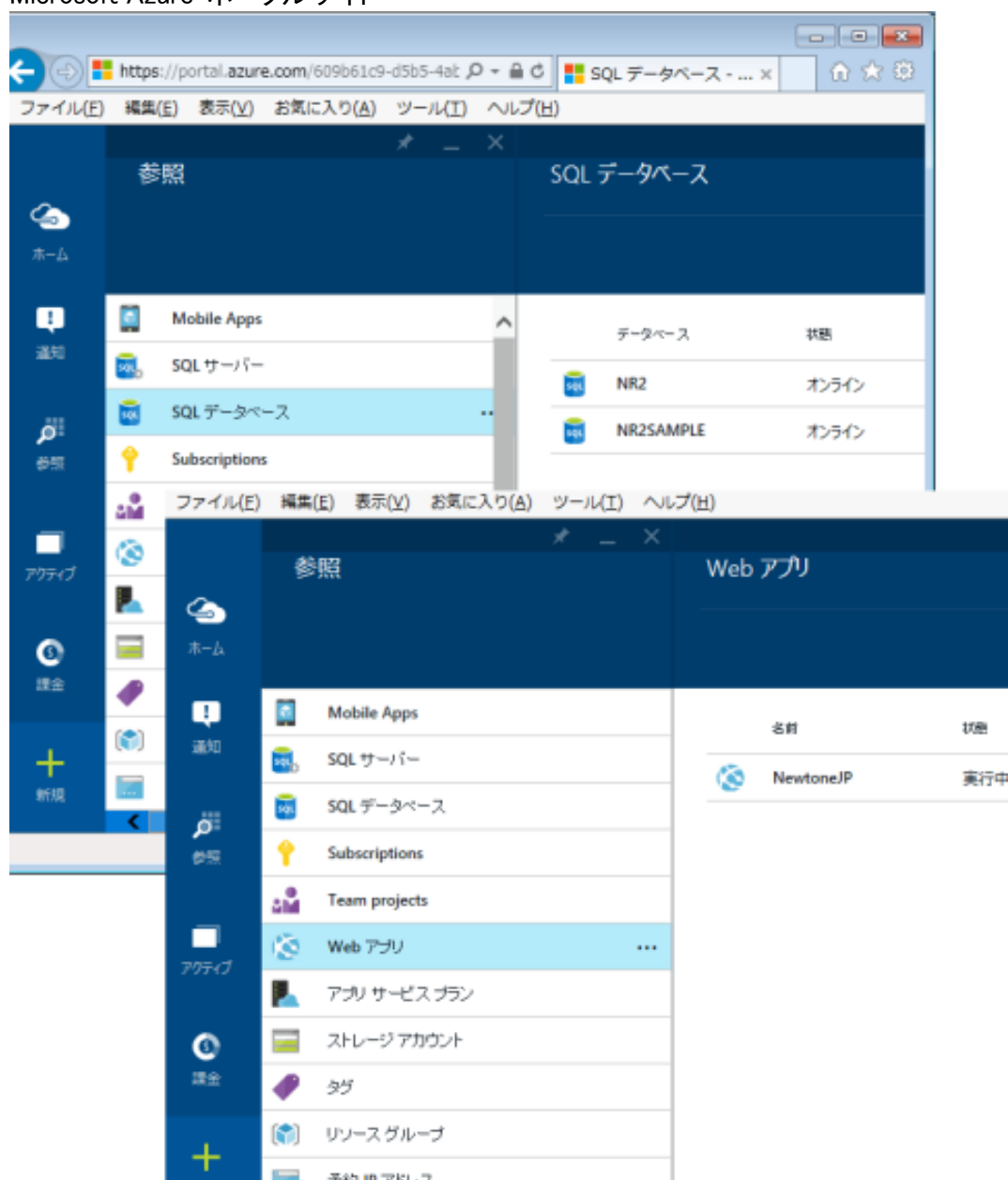
まず、貴社がマイクロソフト社に申込み、Microsoft Azure の「Web アプリ（旧 Web サイト）」機能と「SQL データベース」機能が使用できる状態が必要です。

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「Web アプリ（旧 Web サイト）」機能と「SQL データベース」機能を使用できる状態を前提としています。

下図は、Microsoft Azure の「Web アプリ（旧 Web サイト）」機能と「SQL データベース」機能を使用している場合の Web 上の管理サイト画面の一例です。

Microsoft Azure ポータルサイト





## Microsoft Azure ポータルサイト(旧)



以下の手順を行います。

### 1. 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

仮のサーバーPC、またはローカル PC に、前述の「Web サーバー用 PC」へのインストールを行います。

### 2. 認証レスキュー！用のデータベースを Microsoft Azure の「SQL データベース」に配置する

認証レスキュー！用のデータベースは前述の「Web サーバー用 PC」へのインストールで「データベースのインストール」を行うと(まだテーブルデータの無い)データベースが作成されます。「データベースのインストール」では SQL Server 2012 Express + Management Studio がインストールされます。Microsoft Azure の「SQL データベース」への認証レスキュー！用のデータベースの配置には、この Management Studio を利用すると簡単です。

以下にその手順を示します。

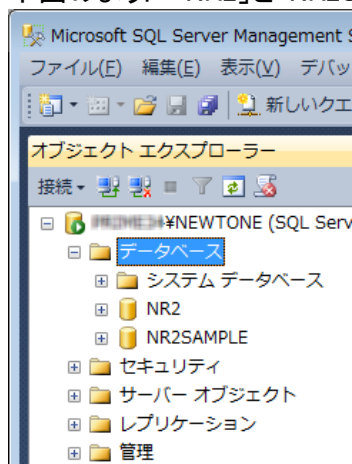
スタートメニューから Microsoft SQL Server 2012→SQL Server Management Studio を選択します。



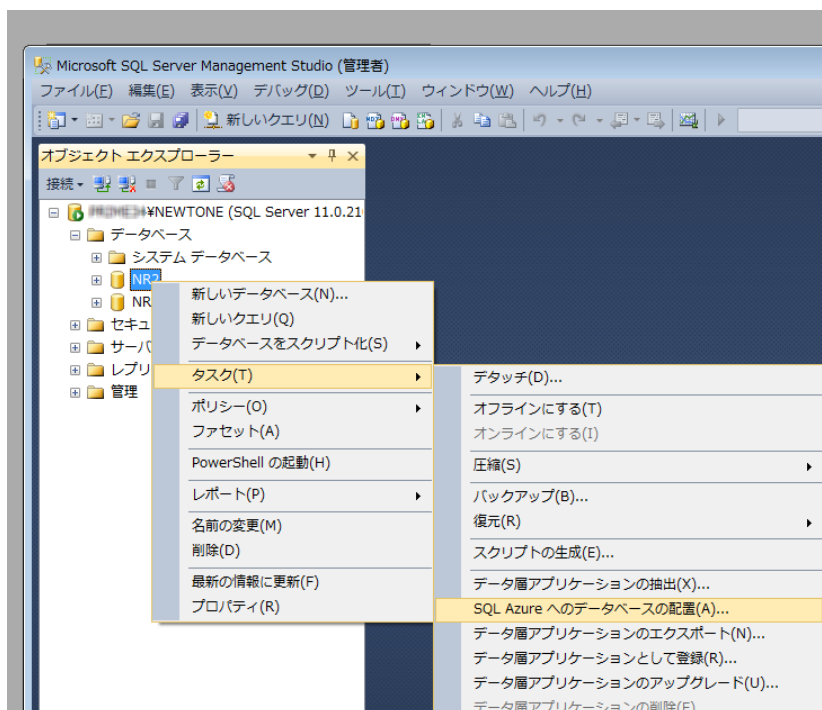
「サーバーへの接続」ダイアログで NEWTONE インスタンスを選択して、接続ボタンを押します。



下図のように「NR2」と「NR2SAMPLE」という 2 つのデータベースがあります。



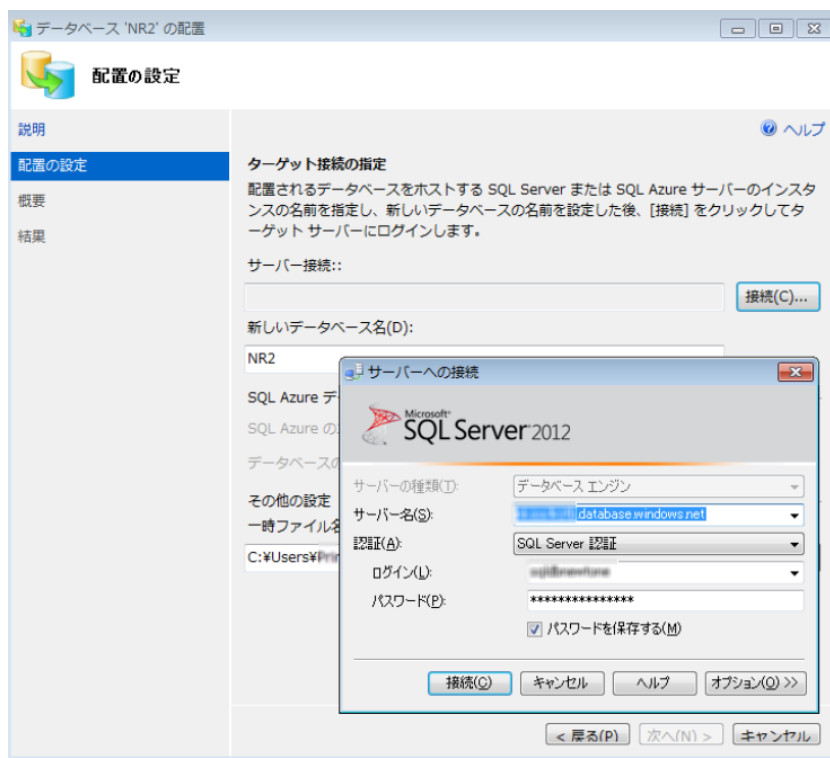
たとえば、データベース「NR2」を Microsoft Azure に配置(コピー)する場合は、オブジェクトエクスプローラーの「NR2」上で右クリックしてコンテキストメニュー内の「タスク」→「SQL Azure へのデータベースの配置」を選択します。



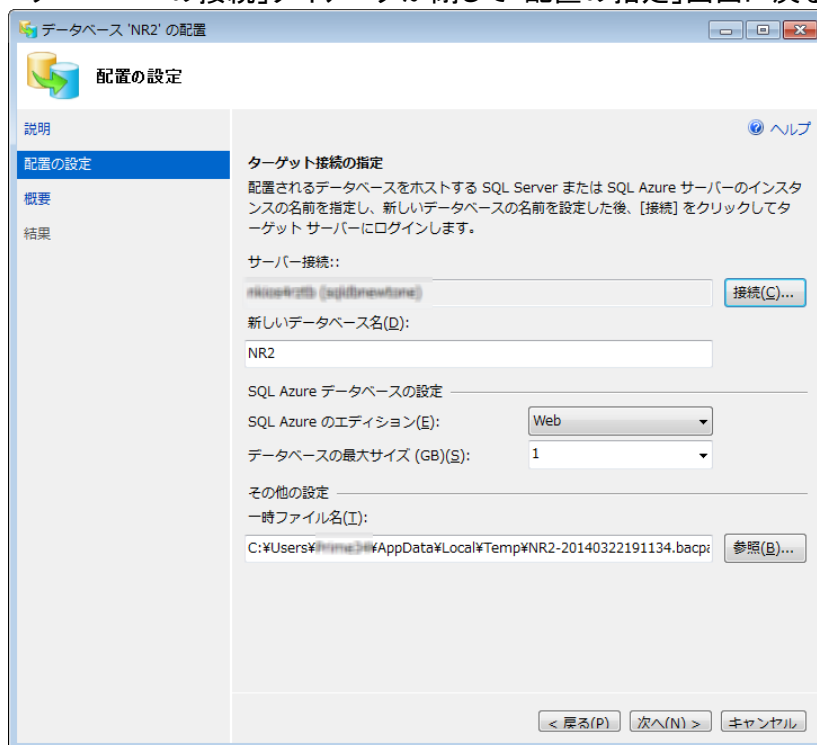
下図のようにデータベースの配置ウィザードが表示されます。  
この画面の説明にあるように Microsoft Azure の SQL データベースのアカウント情報を準備します。「次へ」を選択します。



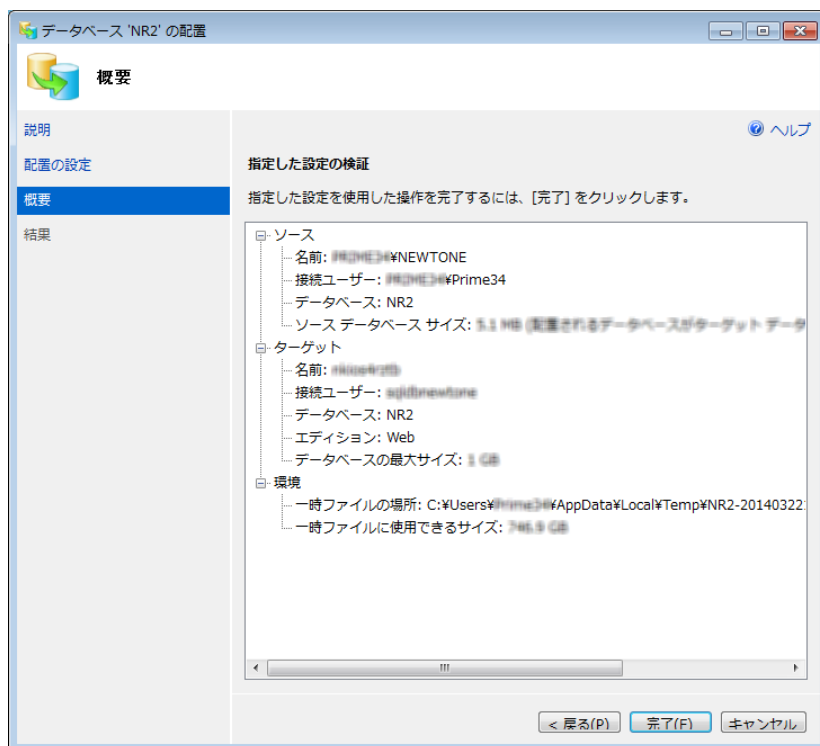
下図で、サーバー接続の「接続」ボタンを押して、「サーバーへの接続」ダイアログが表示されるので、サーバー名や認証(方法)、ログイン(名)、パスワードを Microsoft Azure の SQL データベースのアカウント情報から入力して、接続ボタンを押します。



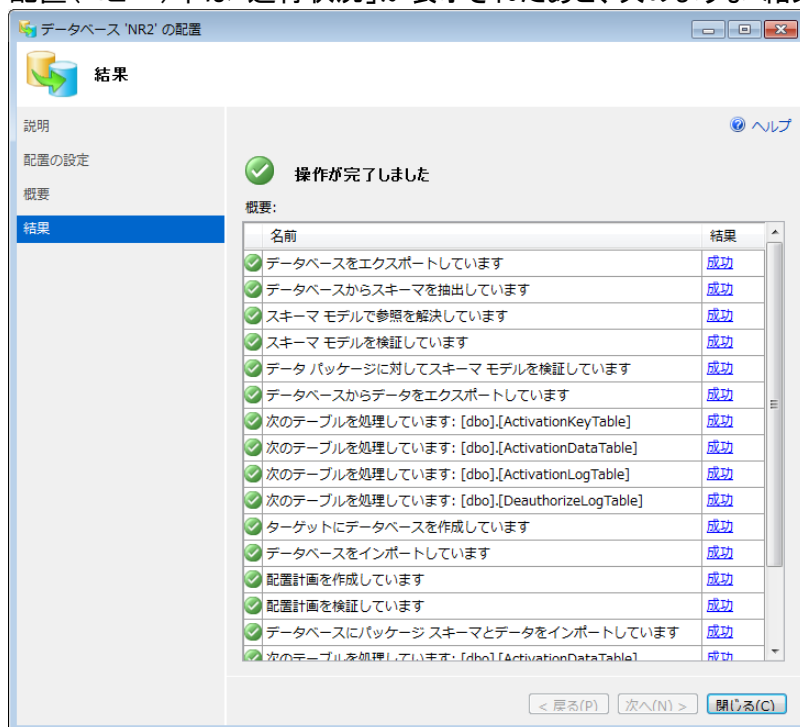
「サーバーへの接続」ダイアログが閉じて「配置の指定」画面に戻るなので「次へ」を選択します。



次に表示される「概要」画面で「完了」ボタンを押します。



配置(コピー)中は「進行状況」が表示されたあと、次のような「結果」画面が表示されます。



「閉じる」ボタンを押して、認証レスキュー！用のデータベース「NR2」の Microsoft Azure の「SQL データベース」への配置(コピー)は終了です。

### 3. 認証レスキュー！の Web サーバー用 PC の「環境設定」処理を完了する

認証レスキュー！の Web サーバー用 PC の「環境設定」のデータベースの指定で「任意接続文字列(Azure など)」を選択して、下のテキストボックスに Azure の「SQL データベース」への接続文字列を入力します。

この接続文字列の確認方法は次の通りです。

Microsoft Azure の SQL データベースの管理ページで、「接続文字列の表示」リンクを押すといくつかの接続文字列が表示されます。その中で、「ADO.NET」用の接続文字列を使用します。その接続文字列の（{ここにパスワードを挿入}となっている）パスワード部分を Microsoft Azure の SQL データベースで設定した実際のパスワードに変えます。

#### <Microsoft Azure の SQL データベースの接続文字列の表示例>

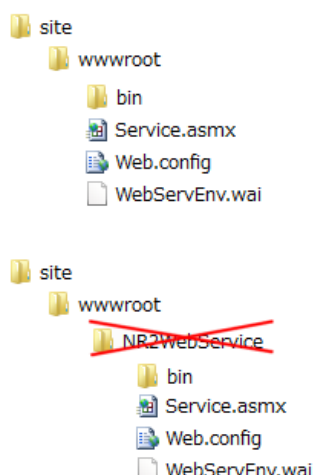
設定したら必ず「データベース接続確認」ボタンで接続を確認してください。

#### 4.Microsoft Azure の「Web アプリ」に認証レスキュー! の Web サービスを配置する

仮のサーバーPC、またはローカル PC に前述の「Web サーバー用 PC」へのインストールを行います。その結果、その PC の IIS のサイトに「NR2WebService」仮想サイトがインストールされます。

エクスプローラでそのサイトの実態フォルダ(例: C:\inetpub\wwwroot\NR2WebService など)内のすべてのファイルとフォルダとを FTP などを利用して Microsoft Azure の「Web アプリ (旧 Web サイト)」のルート(例: xxxxxx/site/wwwroot/)に配置(コピー)します。

この際、Microsoft Azure の「Web アプリ (旧 Web サイト)」の仕様で、Web サービスページ (Service.asmx)は必ず Microsoft Azure の「Web アプリ (旧 Web サイト)」のルート(例: xxxxxx/site/wwwroot/)である必要があります。  
wwwroot の下に作ったフォルダに配置してはいけません。  
たとえば次の場合です。



上図では、上が正しい配置です。下の配置例は URL をそのように指定しても Web サービス実行時に Microsoft Azure からエラーが返りますので注意してください。

Microsoft Azure の「Web アプリ (旧 Web サイト)」側の FTP ホスト名は、たとえば ftp://xxxx-xxxx-xxx-001.ft.azurewebsites.windows.net/site/wwwroot/ などと表します。これは、Microsoft Azure の Web アプリ (旧 Web サイト)の管理ページで確認できます。

この配置するファイルの中には、認証レスキュー！の Web サービス環境設定データ (WebServEnv.wai)が含まれます。  
このファイルは、「Web サーバー用 PC」へのインストール直後には存在しません。  
前述の Web サーバー用 PC の「環境設定」処理を完了すると上記の IIS のフォルダ(例: C:\inetpub\wwwroot\NR2WebService など)内に保存されます。  
このため、上記 3 の「Web サーバー用 PC の「環境設定」処理を完了する」のステップは省略できませんので、ご注意ください。

## 5. 認証管理システムの「環境設定」処理で Web サービスの URL を設定する

認証業務用社内 PC の認証管理システムの「環境設定」処理で Web サービスの URL を設定します。  
次の画面です。

この URL の確認方法は次の通りです。

Microsoft Azure の Web アプリ（旧 Web サイト）の管理ページで、「サイトの URL」で表示されている文字列の左に“http://”を加え、右に“/Service.asmx”を加えた文字列を上図の「環境設定」の Web サービスの URL テキストボックスに設定します。たとえば、  
http://xxxxxxxxxxxxxxxxx.azurewebsites.net/Service.asmx  
といった文字列です。  
設定したら必ず「Web サービス接続確認」ボタンで接続を確認してください。

この URL は、貴社アプリケーションで利用する認証 UI ライブラリ (DLL) の WebServiceURL プロパティに設定する必要がありますのでご注意ください。

## 6. その他の注意点

Microsoft Azure の仕様でシステム標準時刻は UTC (世界協定時) になっています。これにより、「認証管理システム」の「認証状況」処理などで日付を指定して検索する場合、日本では 9 時間を引いた日付で検索する必要があります。  
例えば、2014 年 4 月 1 日の 15:00 のタイムスタンプのデータを検索する場合は、9 時間引いても 2014 年 4 月 1 日の 6:00 で、同じ日付なので 2014/04/01 で検索できます。  
しかし、2014 年 4 月 1 日の 6:00 のタイムスタンプのデータを検索する場合は、9 時間引くと 2014 年 3 月 31 日の 21:00 となり、2014/03/31 で検索しないとヒットしませんので、ご注意ください。  
なお、この場合もデータに記録されるタイムスタンプそのものは操作した (日本) 日時が記録されます。



## ●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

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\NR2\NR2DLL

<64bitOS の場合> C:\Program Files (x86)\Newtone\NR2\NR2DLL

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/Framework4.0 および NRDLL/Framework3.5	Newtone.NR.dll	認証 UI ライブラリ(DLL) 本体
	Newtone.NR.tlb	タイプライブラリ (VisualBasic6.0 使用時必要)
	NewtoneNRvcpp.dll	VC++用ラッパーDLL (VC++使用時必要)
	NewtoneNRvcpp.tlb	タイプライブラリ (VC++使用時必要)
SampleProject	VisualBasic2010 フォルダ	Visual Basic 2010 用のサンプルプロジェクト
	CSharp2010 フォルダ	C# 2010 用のサンプルプロジェクト
	VisualBasic6.0 フォルダ	Visual Basic 6.0 用のサンプルプロジェクト
	VC++2010 フォルダ	Visual C++ 2010 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。

以下に認証 UI ライブラリの利用方法を説明します。

### ■認証 UI ライブラリ機能一覧

アセンブリ: Newtone.NR (Newtone.NR.dll 内)

名前空間: Newtone.NR

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: NewtoneNRvcpp (NewtoneNRvcpp.dll 内)

名前空間: NewtoneNRvcpp

### <クラス>

内容	クラス名
認証に関する各種機能の提供	Activation

## &lt;プロパティ&gt;

内容	プロパティ名	データ型		
		VB2010 /VB6.0	C#	VC++
ベンダアプリケーション開始レジストリキーパス	VendorsProductStartRegistryKeyPath	String	string	BSTR
<p>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。</p> <p>(例)</p> <p>“Software¥Newtone¥NinshoRescue¥NR-200¥SampleProject”</p> <p>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。</p> <p>たとえば、</p> <p>HKEY_LOCAL_MACHINE¥SOFTWARE¥ABC</p> <p>というレジストリパスは次のようになります。</p> <p>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、</p> <p>HKEY_LOCAL_MACHINE¥SOFTWARE¥ABC</p> <p>そのままです。</p> <p>しかし、64bitOS上で32bitアプリケーションが動作する場合は、</p> <p>HKEY_LOCAL_MACHINE¥SOFTWARE¥Wow6432Node¥ABC</p> <p>となります。</p> <p>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</p> <p>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。</p> <p>アプリA内での設定コード例：</p> <p>VendorsProductStartRegistryKeyPath = “Software¥Company¥A”</p> <p>アプリB内での設定コード例：</p>				

VendorsProductStartRegistryKeyPath = "Software¥Company¥B"				
<b>電話で認証時の電話番号</b> 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。	TelephoneNumber	String	string	BSTR
<b>暗号化時のパスワード</b> 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 は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。	EncryptionPassword	String	string	BSTR
<b>暗号化時の Salt 文字列 (8 文字以上)</b> 上記の暗号化時の 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"	EncryptionSaltString	String	string	BSTR
<b>猶予 (試用) 日数 (デフォルト: 0 日、設定可能範囲: 1~365)</b> エンドユーザがライセンス認証登録をし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。	TrialPeriod	VB2010: Integer / VB6: Long	int	long
<b>猶予 (試用) 期間の名称 (デフォルト: "猶予 (試用)")</b> 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	TrialPeriodName	String	string	BSTR

す。 (例)“体験版”				
<b>レンタル日数</b> <b>(デフォルト:0 日、設定可能範囲:1~1100)</b> エンドユーザが初めて認証してからアプリケーションが動作する期間を指定します。 1(日)~1100(日)の間で設定できます。 この RentalPeriod プロパティを 0 に設定すると、レンタル期間は無いことになります。 ※お客様のパッケージ製品にマルチライセンス(例:10 ライセンス)が含まれる場合はレンタル機能は使用できません。レンタル機能を使用するにはシングルライセンス(1 ライセンス)の製品である必要があります。また、レンタル機能を使用する場合は「電話で認証登録」機能は利用できません。	RentalPeriod	VB2010: Integer / VB6: Long	int	long
<b>レンタル期間の名称</b> <b>(デフォルト:“レンタル”)</b> たとえば、ActivateStatusDisp メソッドでは認証状態が表示されます。現在、レンタル中ならば、「レンタル期間残日数:100 日」といったように表示されます。その中の「レンタル」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 RentalPeriodName プロパティです。 (例)“使用可能”	RentalPeriodName	String	string	BSTR
<b>MAC アドレスの使用</b> <b>(デフォルト:True)</b> <b>※「代理認証」に関する処理は MAC アドレスは必須</b> 認証レスキュー！の認証 UI ライブラリでは、認証登録時に認証識別情報としてエンドユーザ 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 UseMacAddress 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。なお、「代理認証」に関する処理ではこの UseMacAddress プロパティにかかわらず MAC アドレスが識別情報として利用されます。	UseMacAddress	Boolean	bool	VARIANT_BOOL
<b>CPU 情報の使用</b> <b>(デフォルト:True)</b> 認証レスキュー！の認証 UI ライブラリでは、認証登録時に認証識別情報としてエンドユーザ PC の CPU 情報を記録しま	UseCpuInfo	Boolean	bool	VARIANT_BOOL

す。この UseCpuInfo 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。				
<b>Web サービスの URL</b>	<b>WebServiceURL</b>	String	string	BSTR
<p>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。</p> <p>以下に例を示します。</p> <p>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/Nr2WebService/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。</p> <p>自社 Web サーバー (IIS) にアクセスさせる例:  “http://www.newtone.co.jp/Nr2WebService/Service.asmx”</p> <p>クラウドサービス Microsoft Azure の Web アプリ (旧 Web サイト) に配置した Web サービスを利用する例:  “http://newtonecojp.azurewebsites.net/Service.asmx”</p>				
<b>Web サービス確認パスワード (8 文字以上)</b>	<b>WebServiceCheckPassword</b>	String	string	BSTR
<p>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。</p> <p>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。</p> <ul style="list-style-type: none"> <li>・大文字の英字 (A～Z)</li> <li>・小文字の英字 (a～z)</li> <li>・数字 (0～9)</li> <li>・記号 (+-*/!#\$%&amp;()=¥@&lt;&gt;?)</li> </ul>				
<b>Web サービスのタイムアウト (デフォルト: 60 秒)</b>	<b>WebServiceTimeout</b>	VB2010: Integer / VB6:	long	long
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。		Long		
<b>Web サービス時の基本認証の使用 (デフォルト:False)</b>	<b>WebServiceUseBasicAuthentication</b>	Boolean	bool	VARIANT_BOOL
<p>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。</p> <p>初期値は、False (基本認証を使用しない) です。</p> <p>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。</p> <p>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。</p> <ol style="list-style-type: none"> <li>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。</li> <li>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。</li> <li>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。</li> </ol> <p>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。</p>				
<b>Web サービス時の基本認証ユーザ名</b>	<b>WebServiceBasicAuthenticationUserName</b>	String	string	BSTR
上記の基本認証使用時に、そのユーザ名を指定します。				
<b>Web サービス時の基本認証パスワード</b>	<b>WebServiceBasicAuthenticationPassword</b>	String	string	BSTR
上記の基本認証使用時に、そのパスワードを指定します				
<b>プロダクト ID の桁数 (デフォルト:17)</b>	<b>ProductIdNumberOfDigits</b>	VB2010: Integer / VB6: Long	long	long
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。				
<b>シリアル No. の桁数 (デフォルト:8)</b>	<b>SerialNoNumberOfDigits</b>	VB2010: Integer / VB6: Long	long	long
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。				
<b>認証登録時の設定プロダクト ID (デフォルト:空文字列)</b>	<b>SetProductID</b>	String	string	BSTR

<p>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製品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。</p> <p>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製品ID用のテキストボックスにこのプロパティ値がグレースアウト表示され文字列コピーはできるが入力できません。</p> <p>このプロパティは認証登録系のメソッドに対し、製品 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製品 ID で自動的に書き換えられるものではありません。</p> <p>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)</p> <p><b>SetProductID、SetSerialNoプロパティの目的</b>  製品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。</p> <p><b>SetProductID、SetSerialNoプロパティの利用例</b>  製品IDやシリアルNo.にレンタル製品とそうではない通常(売切り)製品の識別を持たせるとします。  製品IDとシリアルNo.は、認証レスキュー！のメソッドのUIではなく、貴社側独自のプログラムでエンドユーザの入力を受け必要なチェック後製品IDやシリアルNo.をこのプロパティに設定したあとレンタル製品かどうかに応じてレンタル日数(RentalPeriod)プロパティを設定してから認証登録系のメソッドを呼び出すといった流れになります。</p>				
--	--	--	--	--

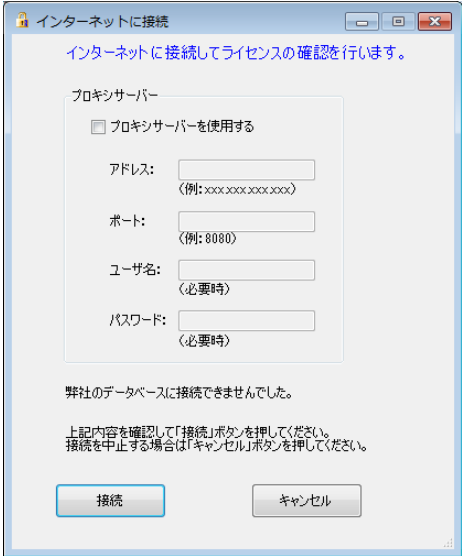



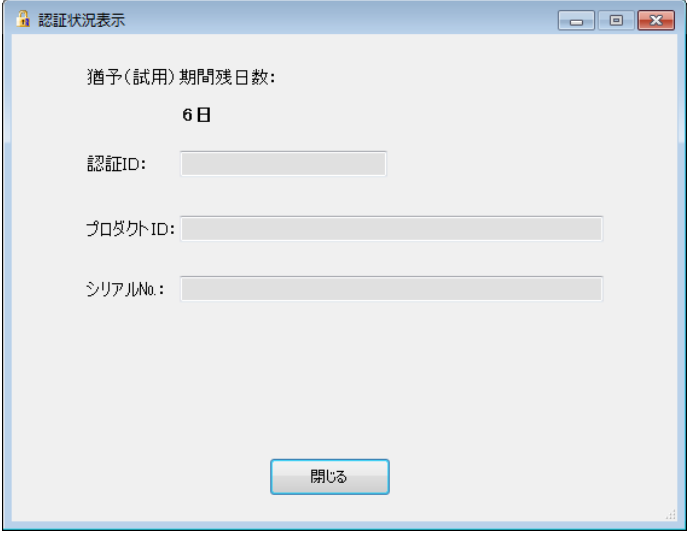
認証登録時の設定シリアル No. (デフォルト:空文字列)	SetSerialNo	String	string	BSTR
<p>このプロパティが空文字列の場合は、認証登録系のメソッドで表示されるダイアログのシリアルNo.用のテキストボックスは、エンドユーザによる入力が可能となります。</p> <p>このプロパティが空文字列ではない場合は、認証登録系のメソッドで表示されるダイアログ中のシリアルNo.用のテキストボックスにこのプロパティ値がグレースアウト表示され文字列コピーはできるが入力できません。</p> <p>このプロパティは認証登録系のメソッドに対し、シリアルNo.を渡すためだけに利用され、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。</p> <p>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)</p> <p>このプロパティの目的と利用例は、<b>SetProductID</b> プロパティの説明をご覧ください。</p>				



<メソッド>

内容	メソッド名・データ型・戻値(引数:無)
「認証状態確認」機能	<p><b>ActivateStatusCheck</b></p> <p><b>データ型</b> VB2010: Integer/VB6: Long C#: int/VC++: long</p> <p><b>戻値</b> 0: 猶予期限切れ(猶予有効時) 1~365: 猶予日数有 400: 未認証(猶予無効時) 500: 認証済み 1000: レンタル期限切れ(レンタル有効時) 1001~2100: レンタル残日数 1~1100(戻値から 1000 を引いて使用する) -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: 終了した有効期限(2000/01/01~2100/12/31) 20000101~21001231: まだ有効な有効期限(2000/01/01~2100/12/31)</p> <p><b>&lt;戻り値が-1 の場合の補足説明&gt;</b> 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。</p>
<p>「認証状態オンライン確認」機能</p> <p>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。</p> <p>通常の(レンタル期間がない)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 となってしまう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</p>	<p><b>ActivateStatusCheckOnline</b></p> <p><b>データ型</b> VB2010: Integer/VB6: Long C#: int/VC++: long</p> <p><b>戻値</b> 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) -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) -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)</p> <p><b>&lt;戻り値が-1 の場合の補足説明&gt;</b> 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)</p>

<p>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。</p>	<p>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。</p> <p><b>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」ダイアログを表示する。</b></p>  <p><b>エンドユーザの選択肢は次の通り</b></p> <ul style="list-style-type: none"> <li>・「接続」ボタン→現在の内容で再度接続する。</li> <li>・「キャンセル」ボタン→このダイアログを閉じて、当該メソッドの戻り値として「-999:その他エラー（接続できないなど）」を返してメソッド終了</li> </ul> <p>そ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。</p>	
<p>「認証状態表示」処理の呼び出し</p>	<p>ActivateStatusDisp</p>	<p>データ型 VB: Boolean</p>

	<p>C#: bool VC++: VARIANT_BOOL <b>戻値</b> 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)</p>
	
	

# 「認証登録/インターネット」処理の呼び出し

ActivateRegisterInternet

## データ型

VB: Boolean

C#: bool

VC++: VARIANT\_BOOL

## 戻値

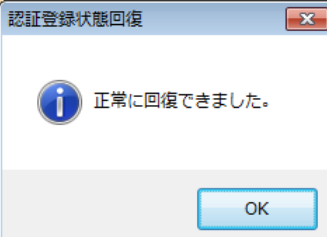
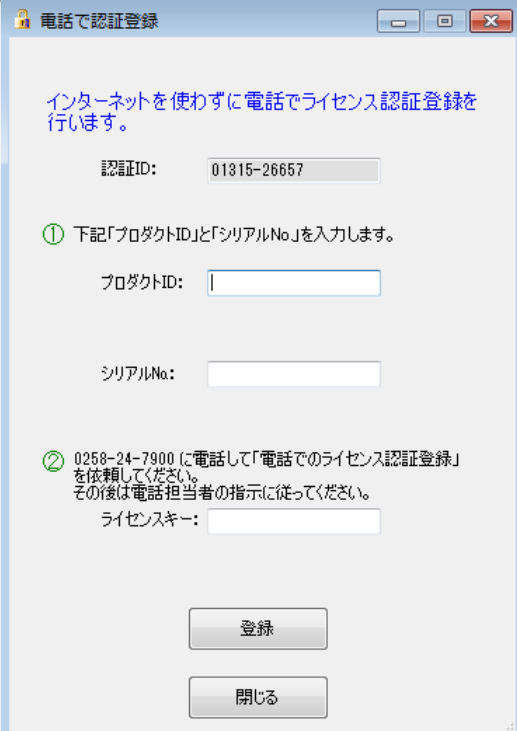
True: 正常

False: エラー（未設定や範囲を超えているプロパティがある）

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。

「OK」ボタンを押すと次の「認証登録状態回復」処理を呼び出します。

「実行」ボタンを押して回復が成功すると次のダイアログが表示されます。

		
<p>「認証登録/電話」処理 の呼び出し</p>	<p>ActivateRegisterTelephone</p>	<p><b>データ型</b> VB: Boolean C#: bool VC++: VARIANT_BOOL</p> <p><b>戻値</b> 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)</p>
		

# 「認証解除/インターネット」処理の呼び出し

ActivateRemoveInternet

## データ型

VB: Boolean

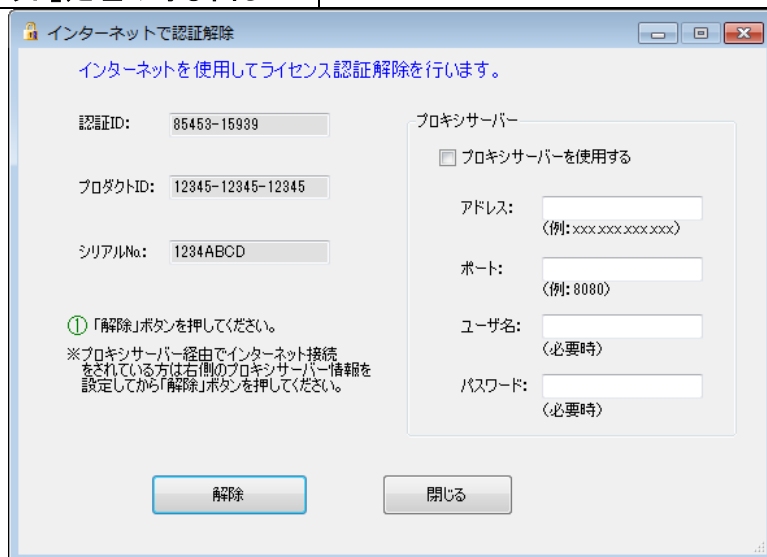
C#: bool

VC++: VARIANT\_BOOL

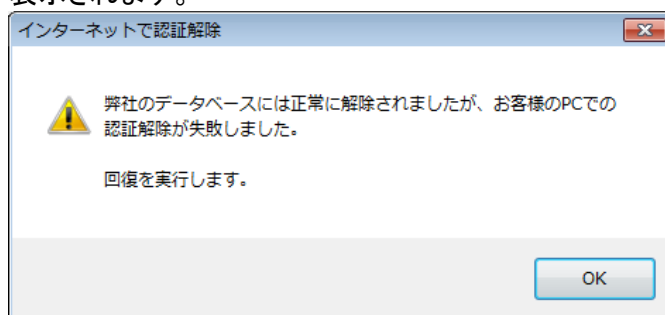
## 戻値

True: 正常

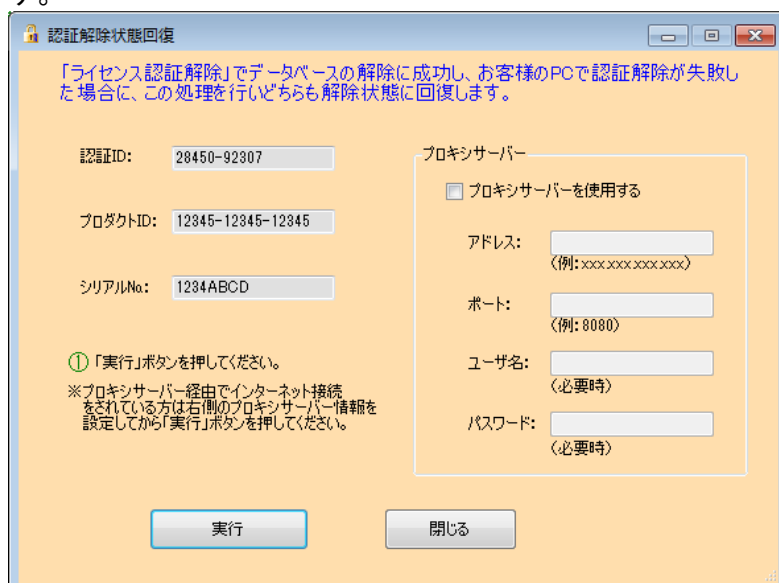
False: エラー (未設定や範囲を超えているプロパティがある)



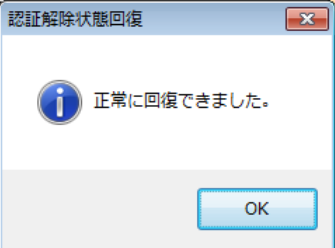
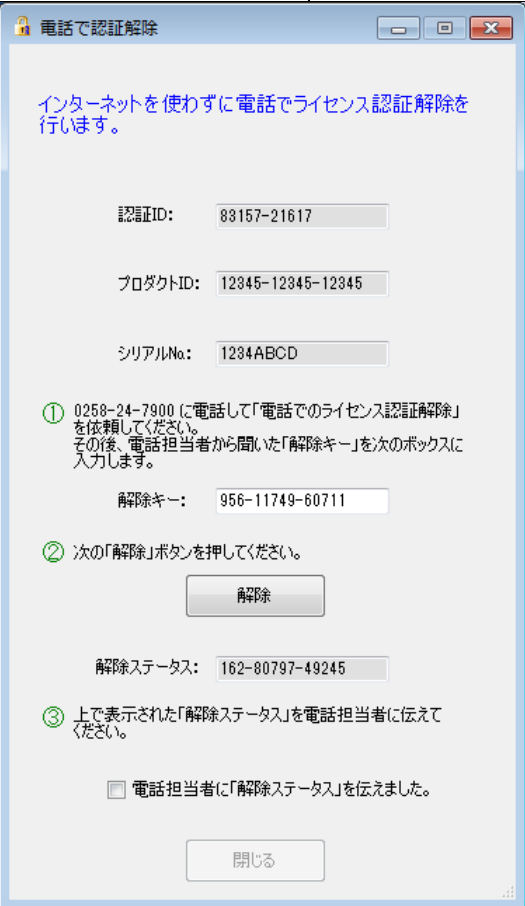
なお、何らかの原因で、データベースは解除済、エンドユーザ PC のレジストリが登録状態になった場合は、次のようなダイアログが表示されます。

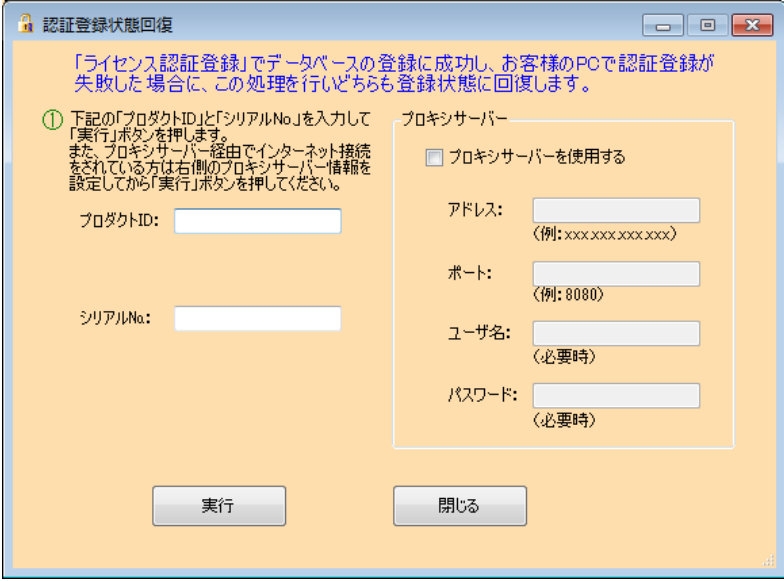
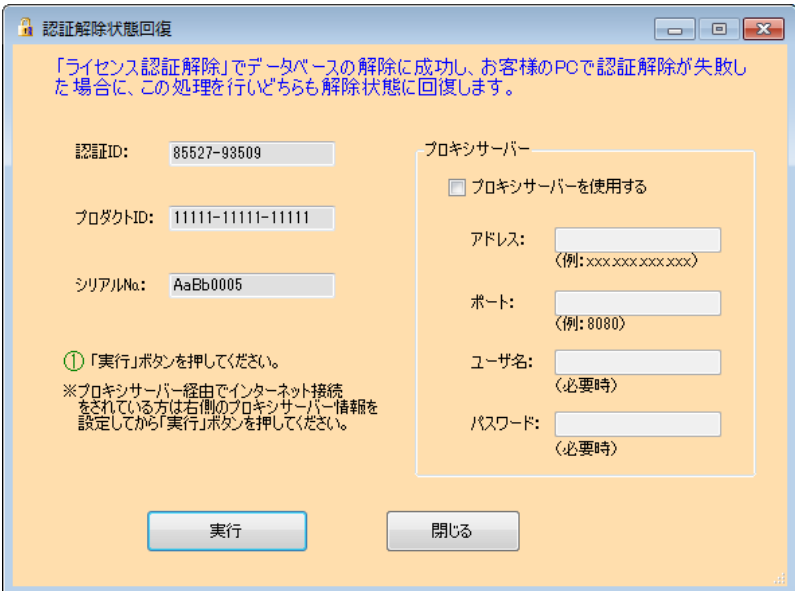


「OK」ボタンを押すと次の「認証解除状態回復」処理を呼び出します。



「実行」ボタンを押して回復が成功すると次のダイアログが表示されます。

		
<b>「認証解除/電話」処理の呼び出し</b>	<b>ActivateRemoveTelephone</b>	<b>データ型</b> VB: Boolean C#: bool VC++: VARIANT_BOOL
		<b>戻値</b> 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)
<b>「認証登録状態回復」処理の呼び出し</b>	<b>RestoreRegisterStatus</b>	<b>データ型</b> VB: Boolean C#: bool VC++: VARIANT_BOOL <b>戻値</b> True: 正常

<p>「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。</p> 	<p>False: エラー (未設定や範囲を超えているプロパティがある)</p>
<p>「認証解除状態回復」処理の呼び出し</p>	<p>RestoreCancelStatus</p> <p><b>データ型</b> VB: Boolean C#: bool VC++: VARIANT_BOOL</p>
<p>「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。</p> 	<p><b>戻値</b> 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)</p>
<p>「有効期限の更新」処理の呼び出し</p>	<p>UpdateOfExpirationDate</p> <p><b>データ型</b> VB: Boolean C#: bool VC++: VARIANT_BOOL</p>



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。

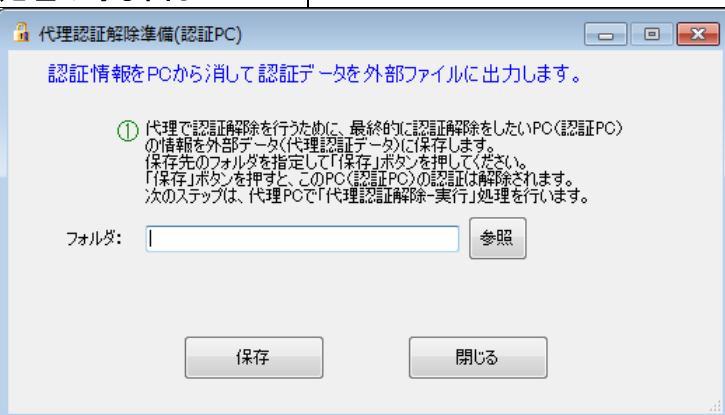
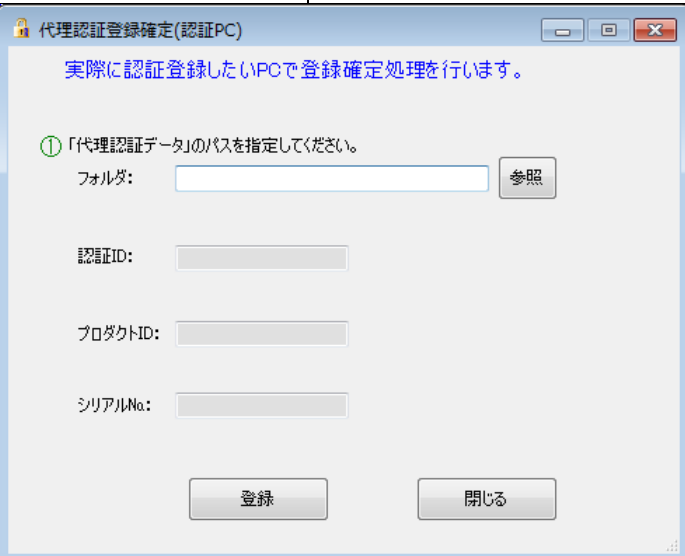
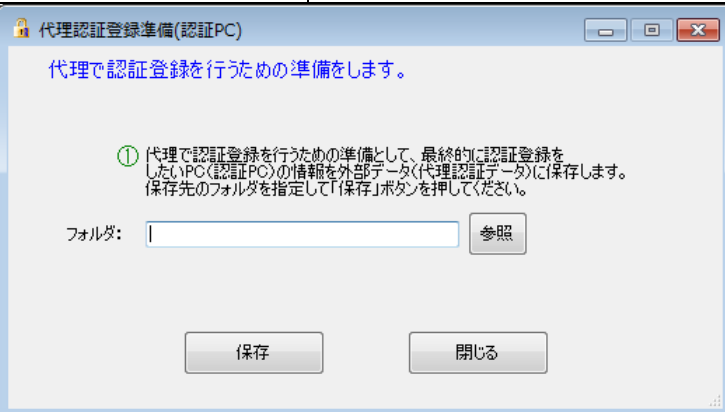
エンドユーザが代理認証を利用している場合で、有効期限によるライセンスを更新する場合はその方法を使います。

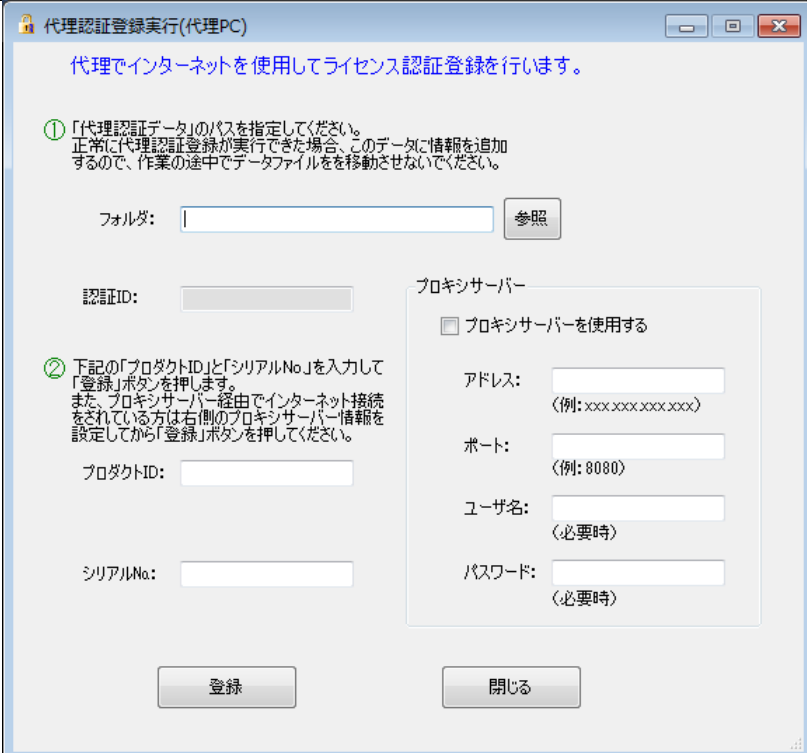
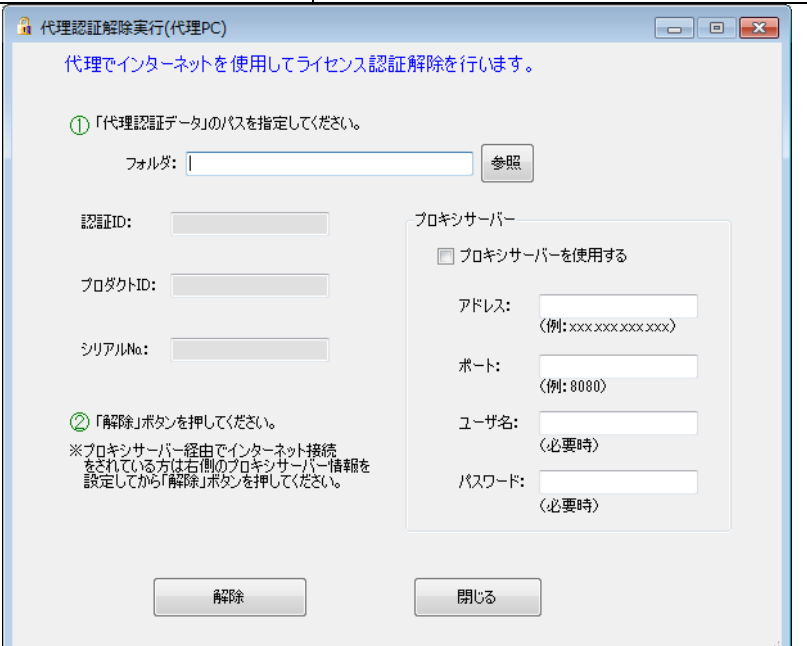
## 戻値

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）

「代理認証登録/準備」 処理の呼び出し	ProxyActivateRegisterPrepare	<b>データ型</b> VB: Boolean C#: bool VC++: VARIANT_BOOL <b>戻値</b> 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)
「代理認証登録/確定」 処理の呼び出し	ProxyActivateRegisterFix	<b>データ型</b> VB: Boolean C#: bool VC++: VARIANT_BOOL <b>戻値</b> 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)
「代理認証解除/準備」 処理の呼び出し	ProxyActivateRemovePrepare	<b>データ型</b> VB: Boolean C#: bool VC++: VARIANT_BOOL <b>戻値</b> 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)
「代理認証登録/実行」 処理の呼び出し (代理 PC で利用)	ProxyActivateRegisterExecute	<b>データ型</b> VB: Boolean C#: bool



	<p>VC++: VARIANT_BOOL 戻値 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)</p>
<p>「代理認証解除/実行」 処理の呼び出し (代理 PC で利用)</p>	<p>ProxyActivateRemoveExecute</p>
	<p>データ型 VB: Boolean C#: bool VC++: VARIANT_BOOL 戻値 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)</p>

## 代理認証機能について

### 上記 5 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. 「代理認証登録/実行」処理の呼び出し (代理 PC で利用)/ProxyActivateRegisterExecute

## 5.「代理認証解除/実行」処理の呼び出し(代理 PC で利用)/ProxyActivateRemoveExecute

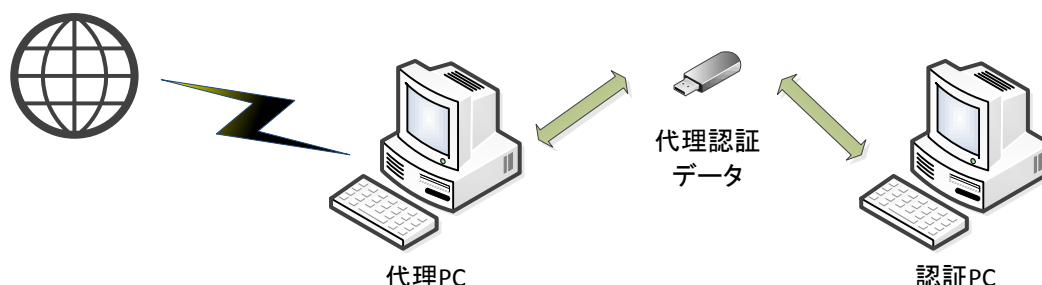
は代理で認証する機能です。

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み込み「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み込み「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み込み「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。

### ◆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。

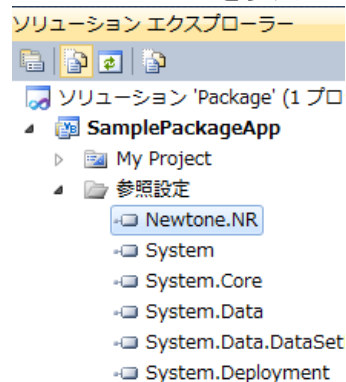
したがって、貴社は最終的に認証したいPC用のアプリケーションの他に代理PCで代理処理をするための(**ProxyApp**のような)アプリケーションを作成し配布する必要があります。

## ■認証 UI ライブラリの参照

### Visual Studio 2010（Visual Basic 2010/C# 2010）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.dll を参照します。

＜Newtone.NR.dll を参照した後の参照設定の様子（Visual Basic 2010）＞



Newtone.NR.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。

#### .NET Framework4.0 用の DLL

＜32bitOS の場合＞ C:\Program Files\Newtone\NR2\NR2DLL\NRDLL\Framework4.0

＜64bitOS の場合＞ C:\Program Files (x86)\Newtone\NR2\NR2DLL\NRDLL\Framework4.0

#### .NET Framework3.5 用の DLL

＜32bitOS の場合＞ C:\Program Files\Newtone\NR2\NR2DLL\NRDLL\Framework3.5

＜64bitOS の場合＞ C:\Program Files (x86)\Newtone\NR2\NR2DLL\NRDLL\Framework3.5

### Visual Studio 6.0（Visual Basic 6.0）の場合

認証 UI ライブラリ(DLL)を Visual Basic (以降 VB)6 から利用する方法は次の通りです。

次の 2 つのファイルが必要です。

Newtone.NR.dll

Newtone.NR.tlb

これら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\NR2\NR2DLL\NRDLL\Framework4.0

＜64bitOS の場合＞ C:\Program Files (x86)\Newtone\NR2\NR2DLL\NRDLL\Framework4.0

#### 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\v4.0.30319\regasm.exe “<DllPath>Newtone.NR.dll” /tl

b:"<DllPath>Newton.NR.tlb" /codebase /nologo

(例:)<32bitOS の場合>

C:¥WINDOWS¥Microsoft.NET¥Framework¥v4.0.30319¥regasm.exe "C:¥Program Files¥Newton¥NR2¥NR2DLL¥NRDLL¥Framework4.0¥Newton.NR.dll" /tlb:"C:¥Program Files¥Newton¥NR2¥NR2DLL¥NRDLL¥Framework4.0¥Newton.NR.tlb" /codebase /nologo

(例:)<64bitOS の場合>

C:¥WINDOWS¥Microsoft.NET¥Framework¥v4.0.30319¥regasm.exe "C:¥Program Files (x86)¥Newton¥NR2¥NR2DLL¥NRDLL¥Framework4.0¥Newton.NR.dll" /tlb:"C:¥Program Files (x86)¥Newton¥NR2¥NR2DLL¥NRDLL¥Framework4.0¥Newton.NR.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¥v4.0.30319¥regasm.exe "<DllPath>Newton.NR.dll" /tlb:"<DllPath>Newton.NR.tlb" /u /nologo

(例:)<32bitOS の場合>

C:¥WINDOWS¥Microsoft.NET¥Framework¥v4.0.30319¥regasm.exe "C:¥Program Files¥Newton¥NR2¥NR2DLL¥NRDLL¥Framework4.0¥Newton.NR.dll" /tlb:"C:¥Program Files¥Newton¥NR2¥NR2DLL¥NRDLL¥Newton.NR.tlb" /u /nologo

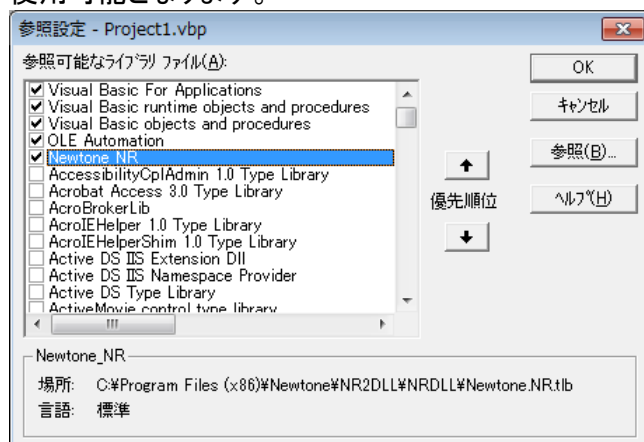
(例:)<64bitOS の場合>

C:¥WINDOWS¥Microsoft.NET¥Framework¥v4.0.30319¥regasm.exe "C:¥Program Files (x86)¥Newton¥NR2¥NR2DLL¥NRDLL¥Framework4.0¥Newton.NR.dll" /tlb:"C:¥Program Files (x86)¥Newton¥NR2¥NR2DLL¥NRDLL¥Newton.NR.tlb" /u /nologo

## 手順 2: VB6 の IDE で DLL を参照

上記の DLL の登録後、VB6 の IDE 上で次の通り参照設定を行います。

メニューバーからプロジェクト→参照設定ダイアログで、「Newton.NR」にチェックを入れると DLL が使用可能となります。



## 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。

Newton.NR.dll

NewtonNRvcpp.dll

NewtonNRvcpp.tlb

これらは、認証レスキュー！2 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\Newton\NR2\NR2DLL\NRDLL\Framework4.0

<64bitOS の場合> C:\Program Files (x86)\Newton\NR2\NR2DLL\NRDLL\Framework4.0

### 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\v4.0.30319\regasm.exe "<DllPath>NewtonNRvcpp.dll"
/tlb:"<DllPath>NewtonNRvcpp.tlb" /codebase /nologo
```

(例:)<32bitOS の場合>

```
C:\WINDOWS\Microsoft.NET\Framework\v4.0.30319\regasm.exe "C:\Program Files\Newton\NR2\NR2DLL\NRDLL\Framework4.0\NewtonNRvcpp.dll" /tlb:"C:\Program Files\Newton\NR2\NR2DLL\NRDLL\Framework4.0\NewtonNRvcpp.tlb" /codebase /nologo
```

(例:)<64bitOS の場合>

```
C:\WINDOWS\Microsoft.NET\Framework\v4.0.30319\regasm.exe "C:\Program Files (x86)\Newton\NR2\NR2DLL\NRDLL\Framework4.0\NewtonNRvcpp.dll" /tlb:"C:\Program Files (x86)\Newton\NR2\NR2DLL\NRDLL\Framework4.0\NewtonNRvcpp.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\v4.0.30319\regasm.exe "<DllPath>NewtonNRvcpp.dll"
/tlb:"<DllPath>NewtonNRvcpp.tlb" /u /nologo
```

(例:)<32bitOS の場合>

```
C:\WINDOWS\Microsoft.NET\Framework\v4.0.30319\regasm.exe "C:\Program Files\Newton\NR2\NR2DLL\NRDLL\Framework4.0\NewtonNRvcpp.dll" /tlb:"C:\Program Files\Newton\NR2\NR2DLL\NRDLL\Framework4.0\NewtonNRvcpp.tlb" /u /nologo
```



(例:)<64bitOS の場合>

```
C:¥WINDOWS¥Microsoft.NET¥Framework¥v4.0.30319¥regasm.exe "C:¥Program Files (x86)¥Newt
one¥NR2¥NR2DLL¥NRDLL¥Framework4.0¥NewtoneNRvcpp.dll" /tlb:"C:¥Program Files (x86)¥Ne
wtone¥NR2¥NR2DLL¥NRDLL¥Framework4.0¥NewtoneNRvcpp.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！2 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(NewtoneNRvcpp.tlb)の Path を指定する必要があります。認証レスキュー！2 の VC++用のサンプルプロジェクトではタイプライブラリ(NewtoneNRvcpp.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。

サンプルプロジェクト PackageApp → PackageApp.h

サンプルプロジェクト ProxyApp → ProxyAppDlg.h

(コード例)

// 認証レスキュー2 の VC++用の tlb をインポート

```
#import "..¥¥¥¥¥NRDLL¥¥Framework4.0¥¥NewtoneNRvcpp.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

### Visual Basic 2010 の場合

付属のサンプルプロジェクト(NR2DLL¥SampleProject¥VisualBasic2010¥PackageApp)の例で説明します。

最初に、Newton.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 Newton.NR.Activation = New Newton.NR.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
```

```
        ' -----  
        ' Newton.NR.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¥Newton¥NinshoRescue¥NR-200¥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 = "0258-24-7900"  
        ' 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) "  
        ' レンタル日数 (デフォルト: 0日、設定可能範囲: 1~1100)  
        Class1.myActivate.RentalPeriod = 0  
        ' レンタル期間の名称 (デフォルト: "レンタル")  
        Class1.myActivate.RentalPeriodName = "レンタル"  
        ' 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/NR2WebService/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 = ""

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 = "[ID:" + ret.ToString + "]" + Class1.myActivate.TrialPeriodName + "期間
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)
```

```
Case 1 To 365 ' 猶予日数有
```

```
stat = "[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

    Case 400 ' 未認証（猶予無効時）
        stat = "[ID:" + ret.ToString + "]" + "ライセンスが認証されていません。" + vbCrLf +
            "メニューは実行できません。" + vbCrLf + "「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)

    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
    Case 1000 ' レンタル期限切れ
        stat = "[ID:" + ret.ToString + "]" + Class1.myActivate.RentalPeriodName + "期間
            期限が切れました。" + vbCrLf + "一度、認証解除を行ってから" + vbCrLf + "新しいシリアルNo. を使って認
            証登録を行ってください。"
        MessageBox.Show(stat, "認証確認", MessageBoxButtons.OK,
            MessageBoxIcon.Exclamation)
    Case 1001 To 2100 ' レンタル日数有
        stat = "[ID:" + ret.ToString + "]" + Class1.myActivate.RentalPeriodName + "期間
            残日数は" + (ret - 1000).ToString + "日です。" + vbCrLf + "続行します。"
        MessageBox.Show(stat, "認証確認", MessageBoxButtons.OK,
            MessageBoxIcon.Information)
        GroupBox1.Enabled = True
    Case -999 ' 認証済ハードウェア情報不一致
        stat = "[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)

    Case -1 ' その他エラー
        stat = "[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)

    Case Else ' 想定外
        stat = "[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)

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

End Class

```

```

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# 2010 の場合

付属のサンプルプロジェクト(NR2DLL¥SampleProject¥CSharp2010¥PackageApp)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 Newton.NR.Activation myActivate = new Newton.NR.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)
    {
        //-----
        //Newton.NR.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¥¥NR-200¥¥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 = "0258-24-7900";
        //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）";
        //レンタル日数（デフォルト：0日、設定可能範囲：1～1100）
        Class1.myActivate.RentalPeriod = 0;
        //レンタル期間の名称（デフォルト："レンタル"）
        Class1.myActivate.RentalPeriodName = "レンタル";
        //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/NR2WebService/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 = "";

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 = "[ID:" + ret.ToString() + "]" + Class1.myActivate.TrialPeriodName + "期  
間残日数は" + ret.ToString() + "日です。" + "¥r" + "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;
            return;
        }

        if (ret >= 1001 && ret <= 2100)
        {

```



```

        //レンタル日数有
        stat = "[ID:" + ret.ToString() + "]" + Class1.myActivate.RentalPeriodName + "期間残
日数は" + (ret-1000).ToString() + "日です。" + "¥r" + "続行します。";
        MessageBox.Show(stat, "認証確認", MessageBoxButtons.OK, MessageBoxIcon.Information);
        GroupBox1.Enabled = true;
        return;
    }

    switch (ret)
    {
        case 0:
            //期限切れ（猶予有効時）
            stat = "[ID:" + ret.ToString() + "]" + Class1.myActivate.TrialPeriodName + "期
間期限が切れました。" + "¥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.Show(stat, "認証確認", MessageBoxButtons.OK,
            MessageBoxIcon.Exclamation);

            break;
        case 400:
            //未認証（猶予無効時）
            stat = "[ID:" + ret.ToString() + "]" + "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.Show(stat, "認証確認", MessageBoxButtons.OK,
            MessageBoxIcon.Exclamation);

            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;

            break;
        case 1000:
            //レンタル期限切れ
            stat = "[ID:" + ret.ToString() + "]" + Class1.myActivate.RentalPeriodName + "
期間期限が切れました。" + "¥r" + "一度、認証解除を行ってから" + "¥r" + "新しいシリアルNo.を使っ
て認証登録を行ってください。";
            MessageBox.Show(stat, "認証確認", MessageBoxButtons.OK,
            MessageBoxIcon.Exclamation);

            break;

        case -999:
            //認証済ハードウェア情報不一致
            stat = "[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);

            break;
        case -1:
            //その他エラー
            stat = "[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);
    }

```

```

        break;
    default:
        //想定外
        stat = "[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);

        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 Studio 6.0 (Visual Basic 6.0) の場合

付属のサンプルプロジェクト(NR2DLL¥SampleProject¥VisualBasic6.0¥PackageApp)の例で説明します。

最初に、Newton.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標準モジュール Module1 内で Public で宣言しています。

```
Public myActivate As New Newton.NR.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。

<Form1 内のコード>

```
Private Sub Form_Load()
```

```

'-----
'Newton.NR.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)

```
myActivate.VendorsProductStartRegistryKeyPath = "Software¥Newton¥NinshoRescue¥NR-200¥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)

```
myActivate.TelephoneNumber = "0258-24-7900"
```

'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)

```
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)

```
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)

```
myActivate.TrialPeriod = 0
```

'猶予期間の名称(デフォルト:"猶予(試用)")

```
myActivate.TrialPeriodName = "猶予(試用)"
```

'レンタル日数(デフォルト:0 日、設定可能範囲:1~1100)

```
myActivate.RentalPeriod = 0
```

'レンタル期間の名称(デフォルト:"レンタル")

```
myActivate.RentalPeriodName = "レンタル"
```

'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

```
myActivate.UseMacAddress = True
```

'CPU 情報の使用(デフォルト:True)

```
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)

```
myActivate.WebServiceURL = "http://localhost/NR2WebService/Service.asmx"
```

'Web サービス時の基本認証の使用(デフォルト:False)

```
myActivate.WebServiceUseBasicAuthentication = False
'Web サービス時の基本認証ユーザ名
myActivate.WebServiceBasicAuthenticationUserName = ""
'Web サービス時の基本認証パスワード
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)
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)
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)
myActivate.SerialNoNumberOfDigits = 8
'Web サービスのタイムアウト(デフォルト:60 秒)
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)
myActivate.SetProductID = ""
'認証登録時の設定シリアル No.(デフォルト:空文字列)
myActivate.SetSerialNo = ""
```

```
CertificationStatus '認証状況の確認
End Sub
```

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。

<Form1 内のコード>

```
Private Sub CertificationStatus()

'-----
'認証状況の確認
'-----

'Command1、Command2、Command3、Frame1 の Enabled プロパティを False にして無効とする。
Frame1.Enabled = False
Command1.Enabled = False
Command2.Enabled = False
Command3.Enabled = False

Dim ret As Integer
Dim stat As String '表示メッセージ

ret = myActivate.ActivateStatusCheck()

'認証状況確認
Select Case ret
    Case 0 '期限切れ(猶予有効時)
        stat = "[ID:" + Str(ret) + "]" + myActivate.TrialPeriodName + "期間期限が切れました。" + 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。"
        MsgBox stat, vbExclamation, "認証確認"
```

```

Case 1 To 365 '猶予日数有
    stat = "[ID:" + Str(ret) + "]" " + myActivate.TrialPeriodName + "期間残日数は"
    " + Str(ret) + "日です。" + vbCr + "続行します。"
    MsgBox stat, vbInformation, "認証確認"
    '猶予(試用)残日数があるので、Button1、Button2、Button3を含む
    'Frame1 の Enabled プロパティを True にして有効とする。
    Frame1.Enabled = True
    'Command1、Command2、Command3 を有効にする。
    Command1.Enabled = True
    Command2.Enabled = True
    Command3.Enabled = True

```

```

Case 400 '未認証(猶予無効時)
    stat = "[ID:" + Str(ret) + "]" " + "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。"
    MsgBox stat, vbExclamation, "認証確認"

```

```

Case 500 '認証済み
    '認証済みなので、Button1、Button2、Button3 を含む
    'Frame1 の Enabled プロパティを True にして有効とする。
    Frame1.Enabled = True
    'Command1、Command2、Command3 を有効にする。
    Command1.Enabled = True
    Command2.Enabled = True
    Command3.Enabled = True

```

```

Case 1000 'レンタル期限切れ
    stat = "[ID:" + Str(ret) + "]" " + myActivate.RentalPeriodName + "期間期限が
    切れました。" + vbCr + "一度、認証解除を行ってから" + vbCr + "新しいシリアル No.を使って
    認証登録を行ってください。"
    MsgBox stat, vbExclamation, "認証確認"

```

```

Case 1001 To 2100 'レンタル日数有
    stat = "[ID:" + Str(ret) + "]" " + myActivate.RentalPeriodName + "期間残日数
    は" + Str(ret - 1000) + "日です。" + vbCr + "続行します。"
    MsgBox stat, vbInformation, "認証確認"

    'Frame1 の Enabled プロパティを True にして有効とする。
    Frame1.Enabled = True
    'Command1、Command2、Command3 を有効にする。
    Command1.Enabled = True
    Command2.Enabled = True
    Command3.Enabled = True

```

```

Case -999 '認証済ハードウェア情報不一致
    stat = "[ID:" + Str(ret) + "]" " + "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。"
    MsgBox stat, vbExclamation, "認証確認"

```

```

Case -1 'その他エラー

```

```
stat = "[ID:" + Str(ret) + "]" " + "認証状況確認中に何らかのエラーが発生しま  
した。"
```

```
MsgBox stat, vbExclamation, "認証確認"
```

```
Case Else '想定外  
stat = "[ID:" + Str(ret) + "]" " + "認証状況確認中に想定外のエラーが発生しま  
した"
```

```
MsgBox stat, vbExclamation, "認証確認"
```

```
End Select
```

```
End Sub
```

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。

<Form2 内のコード>

```
Private Sub Command1_Click()  
    ' 認証状況表示  
    If myActivate.ActivateStatusDisp() = False Then  
        MsgBox "エラー"  
    End If  
End Sub
```

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。

<Form2 内のコード>

```
Private Sub Command2_Click()  
  
    '認証登録/インターネット  
    If myActivate.ActivateRegisterInternet() = False Then  
        MsgBox "エラー"  
    End If  
  
End Sub
```

```
Private Sub Command3_Click()  
  
    '認証登録/電話  
    If myActivate.ActivateRegisterTelephone() = False Then  
        MsgBox "エラー"  
    End If  
  
End Sub
```

```
Private Sub Command4_Click()  
  
    '認証解除/インターネット  
    If myActivate.ActivateRemoveInternet() = False Then  
        MsgBox "エラー"
```

End If

End Sub

Private Sub Command5\_Click()

’認証解除/電話

If myActivate.ActivateRemoveTelephone() = False Then

MsgBox “エラー”

End If

End Sub

Private Sub Command6\_Click()

’代理認証登録/準備

If myActivate.ProxyActivateRegisterPrepare() = False Then

MsgBox “エラー”

End If

End Sub

Private Sub Command7\_Click()

’代理認証登録/確定

If myActivate.ProxyActivateRegisterFix() = False Then

MsgBox “エラー”

End If

End Sub

Private Sub Command8\_Click()

’代理認証解除/準備

If myActivate.ProxyActivateRemovePrepare() = False Then

MsgBox “エラー”

End If

End Sub



## Visual C++の場合

付属のサンプルプロジェクト(NR2DLL¥SampleProject¥VC++2010¥PackageApp)の例で説明します。

最初に、NewtonNrvcpp.IActivation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PackageApp.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！2のVC++用タイプライブラリをインポートします。

```
#import "..¥¥..¥¥..¥¥NRDLL¥¥Framework4.0¥¥NewtonNrvcpp.tlb"
```

認証レスキュー！2のVC++用タイプライブラリの名前空間を宣言します。

```
using namespace NewtonNrvcpp;
```

```
class CPackageAppApp : public CWinApp
{
public:
    CPackageAppApp();
```

```
    . . .
```

```
    //認証レスキューActivationクラスのポインタの宣言
    IActivationPtr m_pActivation;
```

```
    . . .
```

```
};
```

```
//m_pActivationを各ダイアログから呼べるようにCPackageAppAppクラスの変数を宣言
```

```
extern CPackageAppApp theApp;
```

次にPackageAppDlg.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NewtonNrvcpp.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 CPackageAppDlg::OnInitDialog()
{
```

```
    . . .
```

```
    // COM の初期化
```

```
    HRESULT hr = CoInitialize(NULL);
```

```
    // vcppActivationクラスのinterfaceポインタを作る
```

```
    IActivationPtr pIAct(__uuidof(vcppActivation));
```

```
    // IActivationクラスのインスタンスを作成
```

```
    theApp.m_pActivation = pIAct;
```

```
    //-----
```

```
    //Newton.NR.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）
theApp.m_pActivation->VendorsProductStartRegistryKeyPath =
"Software¥¥Newtone¥¥NinshoRescue¥¥NR-200¥¥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 = "0258-24-7900";
//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）";
//レンタル日数（デフォルト：0日、設定可能範囲：1～1100）
theApp.m_pActivation->RentalPeriod = 0;
//レンタル期間の名称（デフォルト："レンタル"）
theApp.m_pActivation->RentalPeriodName = "レンタル";
//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/NR2WebService/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 = "";

CertificationStatus(); //認証状況の確認

...

}
```

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 CPackageAppDlg::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 CPackageAppDlg::CertificationStatus()
{
    //-----
    //認証状況の確認
    //-----

    //Button1、Button2、Button3を無効とする。
    Button1.EnableWindow(false);
    Button2.EnableWindow(false);
    Button3.EnableWindow(false);

    CString stat;
    //表示メッセージ

    long ret = theApp.m_pActivation->ActivateStatusCheck();
    CString strRet;
    strRet.Format(_T("%d"), ret);
    CString tpName = theApp.m_pActivation->TrialPeriodName;
    CString rpName = theApp.m_pActivation->RentalPeriodName;

    if (ret >= 1 && ret <= 365)
    {
        //猶予日数有
        stat = _T("[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);

        return;
    }

    if (ret >= 1001 && ret <= 2100)
    {
        //レンタル日数有
        CString strRet2;
        strRet2.Format(_T("%d"), ret - 1000);
        stat = _T("[ID:") + strRet + _T("] ") + rpName + _T("期間残日数は") + strRet2 + _T("日
です。¥r続行します。");
        MessageBox(stat, _T("認証確認"), MB_OK);
        Button1.EnableWindow(true);
        Button2.EnableWindow(true);
    }
}

```

```

        Button3.EnableWindow(true);
        return;
    }

    switch (ret)
    {
        case 0:
            //期限切れ（猶予有効時）
            stat = _T("[ID:]") + strRet + _T(" ") + tpName + _T("期間期限が切れました。¥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);

            break;

        case 400:
            //未認証（猶予無効時）
            stat = _T("[ID:]") + strRet + _T(" ") + _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);

            break;

        case 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);

            break;

        case 1000:
            //レンタル期限切れ
            stat = _T("[ID:]") + strRet + _T(" ") + rpName + _T("期間期限が切れました。¥r一度、認証解除を行ってから¥r新しいシリアルNo.を使って認証登録を行ってください。");
            MessageBox(stat, _T("認証確認"), MB_OK);
            break;

        case -999:
            //認証済ハードウェア情報不一致
            stat = _T("[ID:]") + strRet + _T(" ") + _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);

            break;

        case -1:
            //その他エラー
            stat = _T("[ID:]") + strRet + _T(" ") + _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);

            break;

        default:
            //想定外
            stat = _T("[ID:]") + strRet + _T(" ") + _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);
    }
}

```

```

        break;
    }
}

```

以下では諸々の具体的な使用法の例を紹介します。PackageApp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 Dlg1::OnBnClickedButton1()
{
    if (theApp.m_pActivation->ActivateStatusDisp() == false)
        MessageBox(_T("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。

```

//認証登録/インターネット
void Dlg1::OnBnClickedButton2()
{
    if (theApp.m_pActivation->ActivateRegisterInternet() == false)
        MessageBox(_T("エラー"));
}

//認証登録/電話
void Dlg1::OnBnClickedButton3()
{
    if (theApp.m_pActivation->ActivateRegisterTelephone() == false)
        MessageBox(_T("エラー"));
}

//認証解除/インターネット
void Dlg1::OnBnClickedButton4()
{
    if (theApp.m_pActivation->ActivateRemoveInternet() == false)
        MessageBox(_T("エラー"));
}

//認証解除/電話
void Dlg1::OnBnClickedButton5()
{
    if (theApp.m_pActivation->ActivateRemoveTelephone() == false)
        MessageBox(_T("エラー"));
}

//代理認証登録/準備
void Dlg1::OnBnClickedButton6()
{
    if (theApp.m_pActivation->ProxyActivateRegisterPrepare() == false)
        MessageBox(_T("エラー"));
}

//代理認証登録/確定
void Dlg1::OnBnClickedButton7()

```

```
{
    if (theApp.m_pActivation->ProxyActivateRegisterFix() == false)
        MessageBox(_T("エラー"));
}

//代理認証解除/準備
void Dlg1::OnBnClickedButton8()
{
    if (theApp.m_pActivation->ProxyActivateRemovePrepare() == false)
        MessageBox(_T("エラー"));
}

//認証状態オンライン確認
void Dlg1::OnBnClickedButton9()
{
    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")
¥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("   -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) ");

    MessageBox(msg, _T("【認証状態オンライン確認メソッド】"));
}

//認証登録状態復元メソッド
void Dlg1::OnBnClickedButton10()
{
    if (theApp.m_pActivation->RestoreRegisterStatus() == false)
        MessageBox(_T("エラー"));
}

//認証解除状態復元メソッド
void Dlg1::OnBnClickedButton11()
{
    if (theApp.m_pActivation->RestoreCancelStatus() == false)
        MessageBox(_T("エラー"));
}
```

## ■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。

**Newton.NR.dll**

**Newton.NR.tlb** (Visual Basic 6.0 で作成したアプリケーションの配布時)

**NewtonNRvcpp.dll** (Visual C++で作成したアプリケーションの配布時)

**NewtonNRvcpp.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！2 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。

### .NET Framework4.0 用の DLL

<32bitOS の場合> C:\Program Files\Newton\NR2\NR2DLL\NRDLL\Framework4.0

<64bitOS の場合> C:\Program Files (x86)\Newton\NR2\NR2DLL\NRDLL\Framework4.0

### .NET Framework3.5 用の DLL

<32bitOS の場合> C:\Program Files\Newton\NR2\NR2DLL\NRDLL\Framework3.5

<64bitOS の場合> C:\Program Files (x86)\Newton\NR2\NR2DLL\NRDLL\Framework3.5

※認証 UI ライブラリ(DLL)の動作には、.NET Framework4.0 または、Framework4.5 が必要です。

### ・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 Basic 6.0 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。

詳しくは、

SampleProject\VisualBasic6.0 フォルダの

【認証 UI ライブラリ(DLL)を VB6 から利用する方法】.txt

をご覧ください。

#### <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(NewtoneNRvcpp.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++2010 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	22222-2222	***** *****
"	"	"	A01-0003	33333-3333	***** *****
"	4 ライセンス	00001-00002-00004	A04-0001	44444-4444	***** *****
"	"	"	"	55555-5555	***** *****
"	"	"	"	66666-6666	***** *****
BBB システム Ver.1	1 ライセンス	00002-00001-00001	B01-0001	77777-7777	***** *****
"	10 ライセンス	00002-00001-00010	B10-0001	88888-8888	***** *****
"	"	"	"	99999-9999	***** *****

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、これから 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 N o.	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。

## ●「認証管理システム」のその他の処理説明

認証業務用社内 PC の「認証管理システム」の処理で先の初期設定で説明していない下図中の赤枠の処理について簡単に説明します。



### ※検索結果の表示上限数について

以降の各処理中の認証キーテーブルや認証データテーブル、およびログテーブルの検索機能では、検索条件の結果が 1 万行を超えるとメッセージが表示され検索条件を絞り込むように案内されます。

ただし、「認証状況」における認証データテーブルの検索結果の行数に関しては、Excel ファイルへの出力機能があるため制限はありません。

それでは、各処理を簡単に説明します。

## ■認証キー作成（表形式）

認証キー作成（表形式）

パッケージ出荷前に製品の認証キー情報を表形式で作成します。

行番号を押して一行選択し、[Delete]キーで削除することができます。

	プロダクトID	シリアル No.	ライセンス数	有効期限利用	有効期限 (例: 2016/03/26)
▶▶ 1			1	<input type="checkbox"/>	

作成      認証キー一覧      終了

※作成した認証キー情報の一覧を表示します。

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。

プロダクトID:  ?

シリアルNo:  ?

ライセンス数:  ?

有効期限設定 ?

☐ 有効期限を利用する      有効期限: 2016/03/27

作成      認証キー一覧      終了

※作成した認証キー情報の一覧を表示します。

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作成します。

プロダクトIDが同一で既に存在するシリアルNoは生成されません。

プロダクトID:  ?

シリアルNo.  ?

現在設定されているシリアルNo.の桁数は8桁です。

上位固定文字列  ?

ランダム指定

- ☒ 数字のみ
- ☐ 数字と英字(大文字)
- ☐ 数字と英字(小文字)
- ☐ 数字と英字(大小文字)

ナンバリング数  0 ?

ライセンス数:  1 ?

有効期限設定 ?

☐ 有効期限を利用する      有効期限: 2016/03/26

作成      認証キー一覧      終了

※作成した認証キー情報の一覧を表示します。

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5列で、1列目はプロダクトID、2列目はシリアルNo.、3列目はライセンス数、4列目は有効期限利用、5列目は有効期限です。  
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。

インポートするファイルを指定してください。

ファイル:

行番号を押して一行選択し、[Delete]キーで削除することができます。

	プロダクトID	シリアル No.	ライセンス数	有効期限利用	有効期限 (例: 2016/03/26)
▶▶ 1			1	<input type="checkbox"/>	

※作成した認証キー情報の一覧を表示します。

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 5 列で、1 列目はプロダクト ID、2 列目はシリアル No.、3 列目はライセンス数、4 列目は有効期限利用、5 列目は有効期限です。4 列目の有効期限利用は「0: 利用しない、1: 利用する」として設定しておきます。

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。

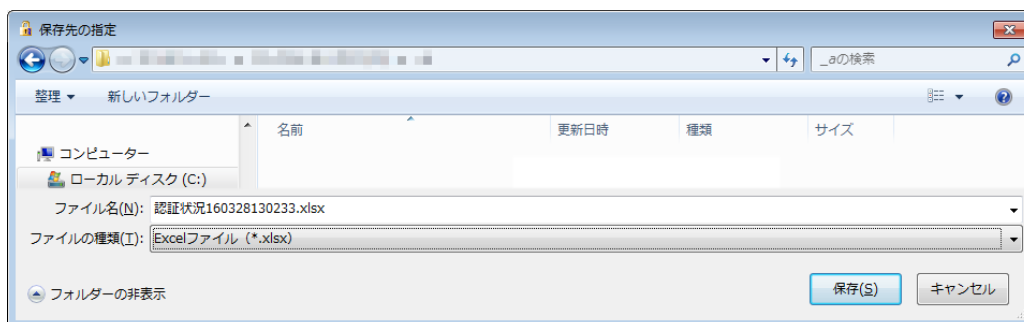
「作成」ボタン:

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。

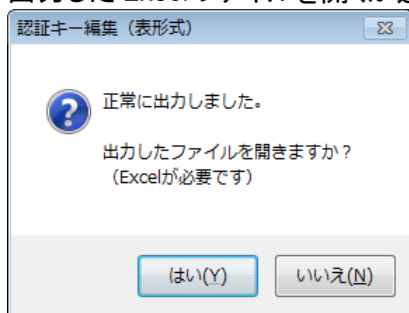
「認証キー一覧」ボタン:

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。

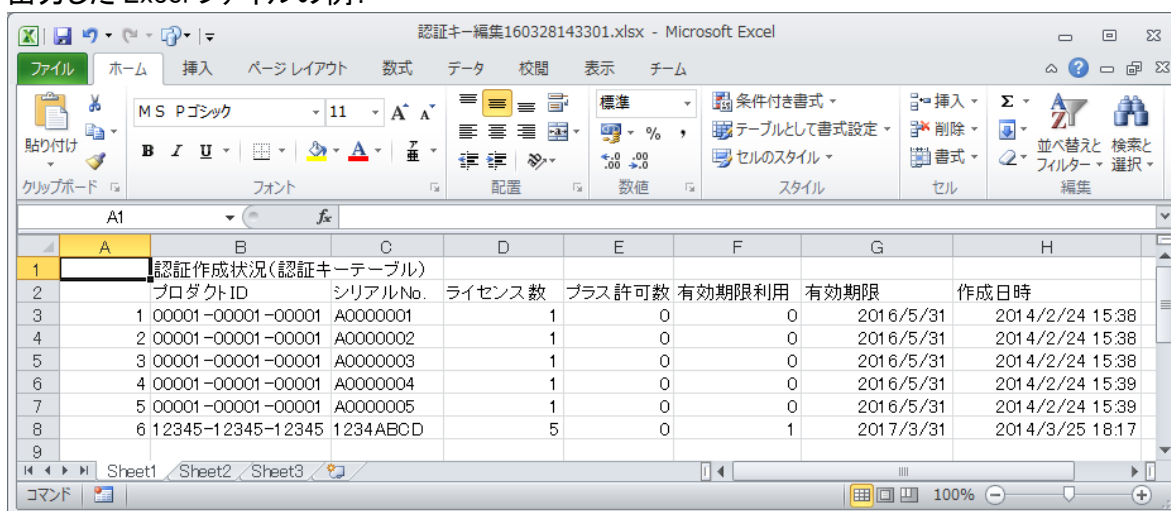




出力した Excel ファイルを開くか選択します。



出力した Excel ファイルの例:



### ■ 認証キ一削除（表形式）

**認証キー削除（表形式）**

製品の認証キー情報を検索して削除します。

検索

プロダクトID:  ? (未入力:指定なし)

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: 2016/03/28)	作成日時
▶ 1	00001-00001-00001	A00000001	1	0	<input type="checkbox"/>	2016/05/31	2014/02/24 15:38:00
2	00001-00001-00001	A00000002	1	0	<input type="checkbox"/>	2016/05/31	2014/02/24 15:38:00
3	00001-00001-00001	A00000003	1	0	<input type="checkbox"/>	2016/05/31	2014/02/24 15:38:00
4	00001-00001-00001	A00000004	1	0	<input type="checkbox"/>	2016/05/31	2014/02/24 15:39:00
5	00001-00001-00001	A00000005	1	0	<input type="checkbox"/>	2016/05/31	2014/02/24 15:39:00
6	12345-12345-12345	1234ABCD	5	0	<input checked="" type="checkbox"/>	2018/03/31	2014/03/25 18:17:00

選択削除      全行削除      終了

現在選択されている行を削除します。      検索で表示されている現在のデータをすべて削除します。

「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。

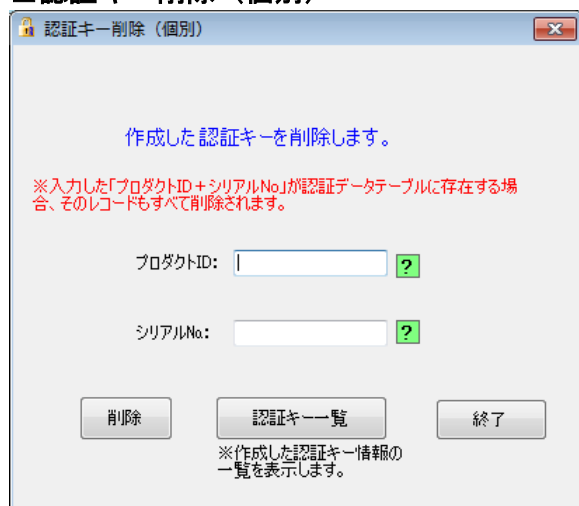
「選択削除」ボタン:

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）



作成した 認証キーを削除します。

※入力した「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、そのレコードもすべて削除されます。

プロダクトID:  ?

シリアルNo:  ?

削除 認証キー一覧 終了

※作成した認証キー情報の一覧を表示します。

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。

## ■ 認証状況

**認証状況**

パッケージ出荷前に作成した認証キー情報と、現在のユーザーの認証登録状況を表示します。

検索

プロダクトID:  ? <未入力:指定なし>

シリアルNo.: 先頭指定文字列:  ? <未入力:指定なし>

認証作成状況日付: ☐ 指定する 2016/03/28 ~ 2016/03/28

認証登録状況日付: ☐ 指定する 2016/03/28 ~ 2016/03/28

**検索実行**

**認証作成状況(認証キーテーブル一覧)**

	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: 2016/03/28)
▶ 1	00001-00001-00001	A0000001	1	0	<input type="checkbox"/>	2016/05/31
2	00001-00001-00001	A0000002	1	0	<input type="checkbox"/>	2016/05/31
3	00001-00001-00001	A0000003	1	0	<input type="checkbox"/>	2016/05/31
4	00001-00001-00001	A0000004	1	0	<input type="checkbox"/>	2016/05/31
5	00001-00001-00001	A0000005	1	0	<input type="checkbox"/>	2016/05/31
6	12345-12345-12345	1234ABCD	5	0	<input checked="" type="checkbox"/>	2018/03/31

**認証登録状況(認証データテーブル一覧)**

	プロダクトID	シリアル No.	認証ID	ライセンスキー	作成日時	MACアドレス 1	MACア ドレス2
▶ 1	00001-00001-00001	A0000001	50533-21818	606126473573993	2014/03/24 15:42:00	E840F260C430	
2	00001-00001-00001	A0000002	17958-26503	660128081831738	2014/03/24 15:42:00	E840F260C430	
3	00001-00001-00001	A0000003	78359-54685	082259796100439	2014/03/24 15:42:00	E840F260C430	
4	12345-12345-12345	1234ABCD	28158-08530	281263669417894	2016/03/28 14:04:51	BC5FF4B0E9DA	

Excelファイル出力      終了

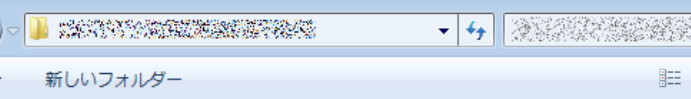
パッケージ出荷前に作成した認証キー情報(上表)と現在のエンドユーザによるライセンスの認証登録状況(下表)を表示します。

「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。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。



保存先の指定

整理 ▾ 新しいフォルダー

コンピューター

ローカルディスク (C:)

名前	更新日時
認証状況140323220233.xlsx	

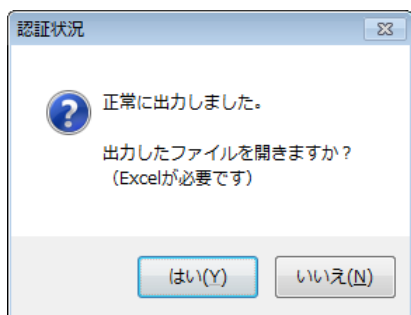
ファイル名(N): 認証状況140323220233.xlsx

ファイルの種類(I): Excelファイル (\*.xlsx)

フォルダーの非表示

保存(S) キャンセル

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:

認証状況160328153906.xlsx - Microsoft Excel

	A	B	C	D	E	F	G	H	I	J	K	L
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	プラス許可数	有効期限利用	有効期限	作成日時				
3	1	00001-00001-00001	A0000001	1	0	0	2016/5/31	2014/2/24 15:38				
4	2	00001-00001-00001	A0000002	1	0	0	2016/5/31	2014/2/24 15:38				
5	3	00001-00001-00001	A0000003	1	0	0	2016/5/31	2014/2/24 15:38				
6	4	00001-00001-00001	A0000004	1	0	0	2016/5/31	2014/2/24 15:39				
7	5	00001-00001-00001	A0000005	1	0	0	2016/5/31	2014/2/24 15:39				
8	6	12345-12345-12345	1234ABCD	5	0	1	2017/3/31	2014/3/25 18:17				
9												
10												
11		認証登録状況(認証データテーブル)										
12		プロダクトID	シリアルNo.	認証ID	ライセンスキー	作成日時	MACアドレス1	MACアドレス2	MACアドレス3	MACアドレス4	MACアドレス5	MACアドレス6
13	1	00001-00001-00001	A0000001	50533-21818	606126473573983	2014/3/24 15:42	EB40F260C430					Intel(R) Core(TM) i7-4790K CPU @ 3.90GHz
14	2	00001-00001-00001	A0000002	17958-26503	660128081831738	2014/3/24 15:42	EB40F260C430					Intel(R) Core(TM) i7-4790K CPU @ 3.90GHz
15	3	00001-00001-00001	A0000003	78359-54685	082259796100439	2014/3/24 15:42	EB40F260C430					Intel(R) Core(TM) i7-4790K CPU @ 3.90GHz
16	4	12345-12345-12345	1234ABCD	28158-08530	281263669417894	2016/3/28 14:04	BC5FF4B0E9DA					Intel(R) Core(TM) i7-4790K CPU @ 3.90GHz
17												



## ■ ログの表示

**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。

検索

日付: ☐ 指定する 2016/03/28 ~ 2016/03/28

プロジェクトID:  ? (未入力:指定なし)

シリアルNo.: 先頭指定文字列:  ? (未入力:指定なし) 検索実行

---

ログの削除

☒ 全削除 ☐ 日付指定 2016/03/28 分まで ログの削除実行 終了

自動No.	作成日時	処理	ステータス	認証区分	メモ	プロジェクトID	シリアルNo.
7	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
8	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
9	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
10	10 2016/03/25 16:18:18	3認証登録	1正常登録	1オンライン	有効期限: 2016/05/31	12345-12345-12345	1234ABCD
11	11 2016/03/26 16:22:11	7キー作成(編集)	1正常登録		【検索条件】指定なし	-	-
12	12 2016/03/27 16:58:43	4認証解除	1正常解除	1オンライン		12345-12345-12345	1234ABCD
13	13 2016/03/27 17:02:16	3認証登録	1正常登録	1オンライン	有効期限: 2016/12/31	12345-12345-12345	1234ABCD
14	14 2016/03/27 17:04:13	4認証解除	1正常解除	1オンライン		12345-12345-12345	1234ABCD
15	15 2016/03/27 17:04:41	7キー作成(編集)	1正常登録		【検索条件】指定なし	-	-
16	16 2016/03/27 17:05:01	3認証登録	1正常登録	1オンライン	有効期限: 2017/03/31	12345-12345-12345	1234ABCD
17	17 2016/03/27 17:53:26	7キー作成(編集)	1正常登録		【検索条件】指定なし	-	-
18	18 2016/03/27 17:53:33	13:有効期限の更新	1正常更新	1オンライン	有効期限: 2018/03/31	12345-12345-12345	1234ABCD
19	19 2016/03/28 08:50:06	4認証解除	1正常解除	1オンライン		12345-12345-12345	1234ABCD
20	20 2016/03/28 13:49:36	7キー作成(編集)	1正常登録		【検索条件】指定なし	-	-
21	21 2016/03/28 14:04:51	3認証登録	1正常登録	1オンライン	有効期限: 2018/03/31	12345-12345-12345	1234ABCD
22	22 2016/03/28 14:15:05	7キー作成(編集)	1正常登録		【検索条件】指定なし	-	-

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。

## ■電話認証登録の対応

電話認証登録の対応

お客様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。

① 以下の項目を(電話で)お客様から聞いて入力後、「登録」ボタンを押してください。

認証ID:  ?

プロダクトID:  ?

シリアルNo.:  ?

登録

結果  
ライセンスキー:

② 正常に登録された場合は結果に表示された「ライセンスキー」をお客様に伝えて、お客様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。

終了

電話で認証登録

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行います。

認証ID: 78235-13266

① 下記「プロダクトID」と「シリアルNo.」を入力します。

プロダクトID:

シリアルNo.:

② 0258-24-7900 に電話して「電話でのライセンス認証登録」を依頼してください。  
その後は電話担当者の指示に従ってください。

ライセンスキー:

登録

閉じる

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（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

電話認証解除の対応

お客様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。

☒ オフライン （お客様のPCが動作している場合）

☐ クラッシュ （お客様のPCが動作不能になった場合）

① 以下の項目について（電話で）お客様から聞いて入力後「解除キーの表示」ボタンを押します。

認証ID:  ?

プロダクトID:  ?

シリアルNo.:  ?

解除キーの表示

解除キー:

② 次に、表示された「解除キー」をお客様に伝えて、お客様画面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。

③ お客様から「解除ステータス」を聞いて次の解除ステータスボックスに入力します。

解除ステータス:

④ 次の解除ボタンを押します。

解除

⑤ 正常に解除できた場合は、お客様に「閉じる」ボタンで処理を終了してもらいます。

終了

### エンドユーザ PC 上の画面

電話で認証解除

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行います。

認証ID:  88157-21617

プロダクトID:  12345-12345-12345

シリアルNo.:  1234ABCD

① 0259-24-7900 に電話して「電話でのライセンス認証解除」を依頼してください。  
その後、電話担当者から聞いた「解除キー」を次のボックスに入力します。

解除キー:  956-11749-60711

② 次の「解除」ボタンを押してください。

解除

解除ステータス:  162-80797-49245

③ 上で表示された「解除ステータス」を電話担当者に伝えてください。

☐ 電話担当者に「解除ステータス」を伝えました。

閉じる

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い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(ユーザの PC が動作不能になった場合)

電話認証解除の対応

お客様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。

☐ オフライン (お客様のPCが動作している場合)

☒ クラッシュ (お客様のPCが動作不能になった場合)

① 以下の項目について(電話で)お客様から聞いて入力後「解除キーの表示」ボタンを押します。

認証ID:  ?

プロダクトID:  ?

シリアルNo.:  ?

解除キーの表示

解除キー:

② 次に、表示された「解除キー」をお客様に伝えて、お客様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。

③ お客様が正常に解除できた場合は下の「解除」ボタンを押してください。

解除

① 以下の項目について(電話で)お客様から聞いて入力後「解除」ボタンを押してください。

プロダクトID:  ?

シリアルNo.:  ?

解除

② お客様に次のようにお話しください。  
「こちらでクラッシュしたPCの解除を行いましたので電話の後でもう一度、認証登録を行ってください。」

終了

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、貴社はお客様（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。

### ■貴社側作業① - 認証キー作成処理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。

### ■エンドユーザ（お客様）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>

インターネットで認証登録

インターネットを使用してライセンス認証登録を行います。

認証ID: 75452-95779

① 下記の「プロダクトID」と「シリアルNo.」を入力して「登録」ボタンを押します。  
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「登録」ボタンを押してください。

プロダクトID: 12345-12345-12345

シリアルNo.: 1234ABCD

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:  (例: xxx.xxx.xxx.xxx)

ポート:  (例: 8080)

ユーザ名:  (必要時)

パスワード:  (必要時)

登録 閉じる

## <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。

認証状況表示

有効期限:  
2017年03月31日まで

認証ID: 37033-50921

プロダクトID: 12345-12345-12345

シリアルNo.: 1234ABCD

閉じる

## ■貴社側作業② - 有効期限によるライセンスの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。





<「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。

## ●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て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 ! 2」ユーザーズガイド (2.2.1)

2014 年 3 月 24 日 初版発行

2016 年 3 月 30 日 第 11 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  
新潟県長岡市寿 1-6-43  
[www.newtone.co.jp](http://www.newtone.co.jp)

Copyright © Newtone Corporation

本書は、法律に定めのある場合または権利者の承諾のある場合を除いて、いかなる方法においても複製・複写することはできません。