User’s Guide to

Implement Auto Update Feature

– Version 1.0.0.0

Table of Contents

[1.0 Overview 3](#_Toc402426217)

[1.1 Description 3](#_Toc402426218)

[2.0 Project Assumptions 3](#_Toc402426219)

[3.0 Project Exclusions 3](#_Toc402426220)

[4.0 System Constraints 3](#_Toc402426221)

[5.0 Implementing Auto update 3](#_Toc402426222)

[6.0 Security 4](#_Toc402426223)

[7.0 Appendix 4](#_Toc402426224)

# Overview

## Description

This document is intended to describe how we can implement auto update feature for existing/new application. We have created generic feature that can be leveraged by other applications. For auto update, the following applications, and extension class were created

* AutoUpdate.exe: Download manager, installing or replacing files.
* AutoUpdates.Extensions.dll: Extension class that was created to help applications that intend to implement auto update feature.
* AutoUpdate.GenerateFiles.exe: Console application required by release management or developers to generate the files that needs to be updated. Creates files that are encrypted.

# Project Assumptions

The following are the assumptions associated with this project:

* User will use this to update their standalone features in EMR or other applications.

# Project Exclusions

Project currently excluded the feature when to check for update, an example if you want to check every time at launch of application, this can always be built in to the application that wants to implement auto update.

# System Constraints

The following are the system constraints associated with this project.

**Technology Constraints:**

* The system should have .NET Framework 4.5 and above installed.
* The application that wants to implement the auto update should be compiled at least .Net 2.0 and greater.

# Implementing Auto update

* Copy these two files to folder where in application needs an update, if the exe already exists in parent folder or application folder, there is no need to copy one.
  + Auto update executable file



* + Auto update extension class library



* Create a manifest file for the application, with following name format **NAMEOFAPPLICATION**Manifest.xml, example for eCWPrintPdf, the manifest file was named as eCWPrintPDFManifest.xml

<Application>

<Name>eCWPrintPdf</Name> 🡨 Name of application

<Version>2.3.0.1</Version> 🡨 Version number

<DownloadUrl>[https://ecwupgrade.eclinicalweb.com/Upgrade/test/</DownloadUrl](https://ecwupgrade.eclinicalweb.com/Upgrade/test/%3c/DownloadUrl)> 🡨 Download Url

<DownloadFileName>eCWPrintPdf.dat</DownloadFileName> 🡨 download file name

</Application>

Sample manifest file



* Generate the two files, first would be a zip file which contains the files to be replaced, second file would be your manifest file. To generate file, we have create a console application AutoUpdate.GenerateFiles.exe, which can be leveraged to generate the files.

Steps to generate the files

1. Copy the files AutoUpdates.Extensions.dll, and AutoUpdate.GenerateFiles.exe to same folder



1. Run command line

AutoUpdate.GenerateFiles.exe /InputFile:"C:\temp\eCWPrintPdf.zip" /InputManifest: "C:\temp\eCWPrintPdfManifest.xml"

1. Once the command line utility completes, we would have two files generated with \*.dat extensions.
2. Upload the \*.dat extension file to the server.

# Security

Both the zip file and manifest file on the server are encrypted using Rijndael algorithm.

# Appendix

In print to pdf file, we have used this code snipped to do auto update check, download and install

if (AutoUpdates.Extensions.Utils.UpdatesAvailable("Manifest file name"))

{

var th = new Thread(() =>

{

XmlDocument doc = new XmlDocument();

string sAutoUpdateManifest = System.IO.File.ReadAllText(Application.StartupPath + "\\" + "(Manifest file name)");

doc.LoadXml(sAutoUpdateManifest);

// Loads the settings from manifest file

XmlNodeList DownloadUrl = doc.GetElementsByTagName("DownloadUrl");

XmlNodeList Name = doc.GetElementsByTagName("Name");

XmlNodeList DownloadFileName = doc.GetElementsByTagName("DownloadFileName");

for (int i = 0; i < DownloadUrl.Count; i++)

{

try

{

AutoUpdates.Extensions.Utils.installUpdateRestart("DownloadFileName", "DownloadUrl", "Application.StartupPath", "Name of the application", "Pass true or false based on whether you want to restart", " Command line arguments");

}

catch (Exception ex)

{

// return false;

}

}

});

th.SetApartmentState(ApartmentState.STA);

th.Start();

}