using HtmlAgilityPack;

using log4net;

using Microsoft.Exchange.WebServices.Data;

using PBE;

using System;

using System.Collections.Concurrent;

using System.Collections.Generic;

using System.Configuration;

using System.IO;

using System.Linq;

using System.Net;

using System.Net.Security;

using System.Reflection;

using System.Security.Cryptography.X509Certificates;

using System.ServiceProcess;

using System.Text;

using System.Threading;

using System.Web;

using ProcessEmail;

using System.Data;

namespace NotificationRobot

{

public partial class NotificationRobotSvc : ServiceBase

{

static ExchangeService service;

static string ExchangeServiceUrl = string.Empty;

// FOLDERS

static FolderId ATTDeactivationFolderId = null;

static FolderId ExceptionFolderId = null;

static FolderId IgnoredFolderId = null;

static FolderId ProcessedFolderId = null;

static int PageSize = 50;

static string QCBroadcastWSUrl = string.Empty;

static string QCChatWSUrl = string.Empty;

static string QCResetWSUrl = string.Empty;

static string QCUserName = string.Empty;

static string QCPassword = string.Empty;

static string QCHotPopMessage = string.Empty;

static string QCResetSuccessText = string.Empty;

// Subjects of Emails

static string ATT\_Ref = string.Empty;

static string ATT\_TRANSFER = string.Empty;

private System.Timers.Timer m\_mainTimer;

private bool m\_timerTaskSuccess;

// System.Collections.Concurrent;

static BlockingCollection<string> queue = new BlockingCollection<string>();

private static ILog log = LogManager.GetLogger(MethodBase.GetCurrentMethod().DeclaringType);

public NotificationRobotSvc()

{

InitializeComponent();

}

Thread thread;

protected override void OnStart(string[] args)

{

log4net.Config.XmlConfigurator.Configure();

log.Info("OnStart");

try

{

//create a thread to get the service going, as OnStart has to return to SCM quickly

thread = new Thread(StartService);

thread.Start();

}

catch (Exception ex)

{

string msgErr = "OnStart \\n\\n" + ex.ToString();

ReportError(msgErr);

}

}

protected override void OnStop()

{

log.Info("Service stopping...");

DataAccess.DA\_Dashboard();

m\_mainTimer.Stop();

m\_mainTimer.Dispose();

m\_mainTimer = null;

try

{

thread.Abort();

if (queue != null)

queue.Dispose();

}

catch (Exception ex)

{

string msgErr = "OnStop \\n\\n " + ex.ToString();

//ReportError(msgErr);

}

log.Info("Service stopped.");

}

private void StartService()

{

try

{

GetSettings();

SetFolders();

// asigna credenciales

StartExchangeService();

// conectese a outlook, debe hacerlo cada 30 minutos

Subscribe();

// Timer to audit every hour number of emails processed

m\_mainTimer = new System.Timers.Timer();

m\_mainTimer.Interval = 60 \* 60000; // every one hour

m\_mainTimer.Elapsed += m\_mainTimer\_Elapsed;

m\_mainTimer.AutoReset = false; // makes it fire only once

m\_mainTimer.Start(); // Start

m\_timerTaskSuccess = false;

// Process each email when it comes

var emailConsumer = System.Threading.Tasks.Task.Factory.StartNew(() => GetEmailsFromQueue());

}

catch (Exception ex)

{

string msgErr = "Start Service \\n\\n " + ex.ToString();

//ReportError(msgErr);

}

}

void m\_mainTimer\_Elapsed(object sender, System.Timers.ElapsedEventArgs e)

{

try

{

// do work

DataAccess.DA\_Dashboard();

m\_timerTaskSuccess = true;

}

catch (Exception ex)

{

m\_timerTaskSuccess = false;

}

finally

{

if (m\_timerTaskSuccess) m\_mainTimer.Start();

}

}

private static void GetSettings()

{

try

{

ExchangeServiceUrl = ConfigurationManager.AppSettings["ExchangeServiceUrl"];

QCBroadcastWSUrl = ConfigurationManager.AppSettings["QCBroadcastWSUrl"];

QCChatWSUrl = ConfigurationManager.AppSettings["QCChatWSUrl"];

QCResetWSUrl = ConfigurationManager.AppSettings["QCResetWSUrl"];

QCUserName = ConfigurationManager.AppSettings["QCUserName"];

QCPassword = ConfigurationManager.AppSettings["QCPassword"];

// EMAIL SUBJECTS CONTAINS

ATT\_Ref = ConfigurationManager.AppSettings["AttTermination"];

ATT\_TRANSFER = ConfigurationManager.AppSettings["AttTransfer"];

}

catch (Exception ex)

{

string msgErr = "Get Settings \\n\\n" + ex.ToString();

ReportError(msgErr);

}

}

private void StartExchangeService()

{

try

{

service = new ExchangeService(ExchangeVersion.Exchange2010\_SP2);

//check if proxy is needed to get to the EWS Url

Uri uri = new Uri("https://outlook.office365.com/EWS/Exchange.asmx");

IWebProxy proxy = WebRequest.GetSystemWebProxy();

bool isBypassed = proxy.IsBypassed(uri);

if (!isBypassed)

{

//if proxy is needed, then provide network credentials

proxy.Credentials = CredentialCache.DefaultNetworkCredentials;

service.WebProxy = proxy;

}

//O365 needs explicit web credentials to be provided

//If your client is targeting an Exchange Online or Office 365 Developer Site mailbox, you have to pass explicit credentials.

//https://msdn.microsoft.com/en-us/library/office/dn567668(v=exchg.150).aspx

#if DEBUG

service.Credentials = new WebCredentials("tdal6392@sa.convergys.com", "pvnptczcvqqwmlxy", "sa.convergys.com"); //enter your credentials to test

#else

service.Credentials = new WebCredentials("notification.robot@na.convergys.com", "wjkxyfrdxgqpzxjy", "na.convergys.com");

#endif

service.Url = uri;

log.Info("Service.Credentials");

//AutodiscoverUrl doesn't work with O365

}

catch (Exception ex)

{

string msgErr = "StartExchangeService \\n\\n " + ex.ToString();

ReportError(msgErr);

}

}

private void SetFolders()

{

try

{

FolderView folderview = new FolderView(10);

log.Info("Set Folders");

var results = service.FindFolders(

WellKnownFolderName.Inbox,

new SearchFilter.SearchFilterCollection(

LogicalOperator.Or,

new SearchFilter.IsEqualTo(FolderSchema.DisplayName, "ATTDeactivation"),

new SearchFilter.IsEqualTo(FolderSchema.DisplayName, "Exception"),

new SearchFilter.IsEqualTo(FolderSchema.DisplayName, "Ignored"),

new SearchFilter.IsEqualTo(FolderSchema.DisplayName, "Processed")),

folderview);

foreach (var folder in results)

{

if (folder is Folder)

{

log.Info(" -- " + folder.DisplayName);

switch (folder.DisplayName)

{

case "ATTDeactivation":

ATTDeactivationFolderId = folder.Id;

break;

case "Exception":

ExceptionFolderId = folder.Id;

break;

case "Ignored":

IgnoredFolderId = folder.Id;

break;

case "Processed":

ProcessedFolderId = folder.Id;

break;

default:

break;

}

}

}

if (ATTDeactivationFolderId == null)

ATTDeactivationFolderId = CreateFolder("ATTDeactivation", WellKnownFolderName.Inbox);

if (ExceptionFolderId == null)

ExceptionFolderId = CreateFolder("Exception", WellKnownFolderName.Inbox);

if (IgnoredFolderId == null)

IgnoredFolderId = CreateFolder("Ignored", WellKnownFolderName.Inbox);

if (ProcessedFolderId == null)

ProcessedFolderId = CreateFolder("Processed", WellKnownFolderName.Inbox);

}

catch (Exception ex)

{

string msgErr = "Creating folders \\n\\n" + ex.ToString();

ReportError(msgErr);

}

}

private static FolderId CreateFolder(string DisplayName, WellKnownFolderName parentFolderName)

{

FolderId folderId = null;

try

{

Folder folder = new Folder(service);

folder.DisplayName = DisplayName;

folder.Save(parentFolderName);

folderId = folder.Id;

}

catch (Exception ex)

{

string msgErr = "Create Folder\\n\\n " + ex.ToString();

ReportError(msgErr);

}

return folderId;

}

// Durante un re Subscribe o si bajo servicio, puede que entraron nuevos emails

// lo primero que hace es verificar que nuevos emails hay en inbox para procesarlos

// y los guarda en queue

// solo al inicio luego de un Subscribe

// de ahi cualquier nuevo email entra por OnEvent

private static void GetPendingEmails()

{

log.Info("Getting pending emails.");

try

{

bool process = true;

if (process)

{

int offset = 0;

int pageSize = PageSize;

bool more = false;

ItemView view = new ItemView(pageSize, offset, OffsetBasePoint.Beginning);

// start with the oldest one in the inbox

view.OrderBy.Add(ItemSchema.DateTimeReceived, SortDirection.Ascending);

view.PropertySet = PropertySet.IdOnly;

FindItemsResults<Item> results;

do

{

results = service.FindItems(WellKnownFolderName.Inbox, view);

foreach (var item in results.Items)

{

queue.Add(item.Id.UniqueId);

}

more = results.MoreAvailable;

if (more)

{

view.Offset += pageSize;

}

} while (more);

}

}

catch (Exception ex)

{

string msgErr = "GetPendingEmails\\n\\n " + ex.ToString();

//ReportError(msgErr);

}

}

// llamado OnStart y luego cada 30 minutos

private static void Subscribe()

{

GetPendingEmails();

log.Info("Subscribing...");

try

{

StreamingSubscription subscription = service.SubscribeToStreamingNotifications(

new FolderId[] { WellKnownFolderName.Inbox },

EventType.NewMail);

StreamingSubscriptionConnection connection = new StreamingSubscriptionConnection(service, 30);

connection.AddSubscription(subscription);

connection.OnNotificationEvent += OnNotificationEvent; // for each new email

connection.OnDisconnect += OnDisconnect; // when disconnected every 30 minutes of after error

connection.OnSubscriptionError += OnSubscriptionError; // when an error, it calls OnDisconnect (may delete)

connection.Open();

}

catch (Exception ex)

{

//wait 5 minutes before retrying

Thread.Sleep(5 \* 60 \* 1000);

Subscribe();

}

}

// new email add in queue variable

private static void OnNotificationEvent(object sender, NotificationEventArgs args)

{

try

{

foreach (var notification in args.Events)

{

if (notification.EventType == EventType.NewMail)

{

if (notification is ItemEvent)

{

queue.Add(((ItemEvent)notification).ItemId.UniqueId);

}

}

}

}

catch (Exception ex)

{

string msgErr = "OnNotificationEvent\\n\\n " + ex.ToString();

ReportError(msgErr);

}

}

private static void OnDisconnect(object sender, SubscriptionErrorEventArgs args)

{

CheckConnectionAndSubscription((StreamingSubscriptionConnection)sender);

}

// solo para log, porque siempre genera un OnDisconnect y se vuelve a subscribir

private static void OnSubscriptionError(object sender, SubscriptionErrorEventArgs args)

{

log.Info("OnSubscriptionError");

try

{

if (args != null)

if (args.Exception != null)

{

log.Info(args.Exception.ToString());

}

}

catch (Exception ex)

{

}

}

// Luego de 30 minutos, o un error, o disconnect,

// vuelve a subscribirse es un ReSubscribe, reconnect to outlook

private static void CheckConnectionAndSubscription(StreamingSubscriptionConnection connection)

{

try

{

if (connection != null)

{

if (connection.IsOpen)

{

//Microsoft.Exchange.WebServices.Data.ServiceLocalException:

//Subscriptions can't be removed from an open connection.

connection.Close();

var subscriptions = connection.CurrentSubscriptions;

if (subscriptions != null)

{

foreach (var subscription in subscriptions)

{

try

{

subscription.Unsubscribe();

connection.RemoveSubscription(subscription);

}

catch (Exception ex)

{

log.Error(ex);

}

}

}

}

connection.Dispose();

}

}

catch (Exception ex)

{

string msgErr = "CheckConnectionAndSubscription \\n\\n" + ex.ToString();

ReportError(msgErr);

}

Subscribe();

}

private static bool RunProcess()

{

DateTime dVal = DateTime.Now;

int iDAY = (int)dVal.DayOfWeek;

int iHour = dVal.Hour;

if (iDAY == 7) return true;

//if (iDAY == 6 && iHour > 14) return;

if (iHour > 22 || iHour < 4) return true;

return false;

}

private static void GetEmailsFromQueue()

{

if (RunProcess()) return;

try

{

foreach (var item in queue.GetConsumingEnumerable())

{

ProcessEmail(item);

}

}

catch (Exception ex)

{

DataAccess.UpdateMonitor("getEmailsFromQueue", "1");

string msgErr = "GetEmailsFromQueue \\n\\n" + ex.ToString();

ReportError(msgErr);

}

}

private static void ProcessEmail(string itemId)

{

Item item = null;

StringBuilder sb = new StringBuilder();

try

{

item = Item.Bind(service, itemId);

if (item is EmailMessage)

{

EmailMessage email = item as EmailMessage;

if (!string.IsNullOrEmpty(email.Subject))

{

string sender = email.Sender.Address;

string dtSended = email.DateTimeSent.ToString().Trim();

string subject = email.Subject.ToUpper();

string msg1 = email.Body.Text.ToUpper();

log.Info(sender+ " "+subject);

string AttachFile = "";

try

{

AttachFile = email.Attachments[0].Name.ToString();

if (AttachFile.Contains(".jpg"))

{

AttachFile = email.Attachments[1].Name.ToString();

if (AttachFile.Contains(".jpg"))

AttachFile = email.Attachments[2].Name.ToString();

}

}

catch (Exception es) {

AttachFile = "";

}

// ATT TERMINATION WITH EMAIL

//===========================

if (subject.IndexOf(ATT\_Ref) > 3) // && dtSended == "convergys@myworkday.com")

{

log.Info("ATT EMAIL " + item.Body);

string msgError = ProcessTerminations.MainProcessTermination(item.Body, dtSended, sender, subject, "ATT");

if (msgError != "")

{

ReportError(msgError);

Status.Att2 = Status.Att2 + 1;

}

else

{

DataAccess.UpdateMonitor("", "2");

Status.Att1 = Status.Att1 + 1;

}

item.Move(ATTDeactivationFolderId);

return;

}

// ATT TRANSFERS WITH EXCEL ATTACHED

//==================================

if (subject.IndexOf(ATT\_TRANSFER) > 3) // && dtSended == "convergys@myworkday.com")

{

Status.Transfers = Status.Transfers + 1;

ProcessTransfers.WithAttachment(email, sender);

item.Move(ATTDeactivationFolderId);

return;

}

// PASSWORD RESET FOR QUICK CONNECT, CALLS A WEB SERVICE

string[] allow = new[] { "password reset" };

string[] exclude = new[] { "automatic reply", "notification robot service error" };

bool allowed = allow.Any(s => email.Subject.ToLower().Contains(s.ToLower()));

bool excluded = exclude.Any(s => email.Subject.ToLower().Contains(s.ToLower()));

if (allowed && !excluded)

{

var list = ReadEmail(email.Body);

if (list.Count > 0)

{

//get the template every time, to avoid restarting the service

QCHotPopMessage = ConfigurationManager.AppSettings["QCHotPopMessage"];

bool sent = false;

foreach (ItemRequest itemrequest in list)

{

if (!string.IsNullOrEmpty(itemrequest.EmployeeId))

{

var employee = DataAccess.GetEmployee(itemrequest.EmployeeId);

if (employee != null)

{

if (!string.IsNullOrEmpty(itemrequest.Password))

{

if (itemrequest.Password.Trim().Any(Char.IsWhiteSpace))

{

//ignore

}

else

{

itemrequest.TMEmailAddress = employee.SupervisorEmail;

itemrequest.TMFirstName = employee.SupervisorFirstName;

sent = QCSendHotPopMessage(itemrequest);

if (!sent)

sb.AppendLine("Could not send QuickConnect message to user " + itemrequest.Name + " (" + itemrequest.EmployeeId + ").");

sent = QCResetCredentials(itemrequest);

if (!sent)

sb.AppendLine("Could not reset password in QuickConnect vault for user " + itemrequest.Name + " (" + itemrequest.EmployeeId + ").");

}

}

else

{

sb.AppendLine("Password is empty for employee " + itemrequest.Name + " (" + itemrequest.EmployeeId + ").");

}

}

else

{

sb.AppendLine("Employee " + itemrequest.Name + " (" + itemrequest.EmployeeId + ") not found in Employee database.");

}

}

else

{

sb.AppendLine("Employee ID not found in \"CVG Reference Number\" column for user " + itemrequest.Name + ".");

}

}

var tms = from e in list

where e.TMEmailAddress != null

group e by e.TMEmailAddress into newGroup

orderby newGroup.Key

select newGroup;

foreach (var group in tms.ToList())

{

sent = ComposeEmailAndSend(group.Key, group.ToList());

if (!sent)

{

sb.AppendLine("Could not send notification email to TL " + group.Key);

}

}

}

else

{

sb.AppendLine("No request(s) to process in email.");

}

string message = sb.ToString();

if (string.IsNullOrEmpty(message))

item.Move(ProcessedFolderId);

else

ProcessExceptionEmail(item, message, sender);

}

else

{

item.Move(IgnoredFolderId);

}

}

else

{

ProcessExceptionEmail(item, "Email missing subject.", "");

}

}

else

{

item.Move(IgnoredFolderId);

}

}

catch (Exception ex)

{

string msgErr = "Process Email " + ex.ToString();

ReportError(msgErr);

if (item != null)

ProcessExceptionEmail(item, ex.ToString(), "");

}

}

private static void ReportError(string msgError)

{

DateTime dVal = DateTime.Now;

int iDAY = (int)dVal.DayOfWeek;

int iHour = dVal.Hour;

//if (iDAY == 7) return;

//if (iDAY == 6 && iHour > 14) return;

if (iHour > 22 || iHour < 3) return;

msgError = msgError + " Check logs ORLIWV022 according to this time in D:\\Services\\NotificationRobotSvc\\Logs Also check last and next request for deactivation if it was done " + msgError;

// SendEmails.SendEmail("ERROR IN NOTIFCIATION ROBOT, CHECK LOGS IN SERVER ORLIWV022", msgError);

log.Error(msgError);

}

private static List<ItemRequest> ReadEmail(string emailBody)

{

List<ItemRequest> list = new List<ItemRequest>();

try

{

HtmlDocument document = new HtmlDocument();

string xpath = "//table";

document.LoadHtml(emailBody);

var tables = document.DocumentNode.SelectNodes(xpath);

if (tables != null)

{

var table = tables.FirstOrDefault(); //only get the first table

var rows = table.Descendants("tr");

int items = 0;

foreach (var row in rows)

{

var cells = row.Descendants("td").ToArray();

var item = new ItemRequest();

if (items > 0) //start on 2nd row, 1st one is header

{

item.ResetType = cells[0].InnerText.Trim(); //now mapped to "Tool" column in current DSC email format

item.ToolID = cells[1].InnerText.Trim(); //now mapped to "Login ID format/example" column in current DSC email format

item.Name = cells[2].InnerText.Trim() + " " + cells[3].InnerText.Trim();

item.FirstName = cells[2].InnerText.Trim();

item.LastName = cells[3].InnerText.Trim();

item.CVGReferenceNumber = cells[6].InnerText.Trim();

item.Password = cells[7].InnerText.Trim();

//ticket starts with IM

int position = item.CVGReferenceNumber.ToLower().IndexOf("im");

if (position > 0)

{

item.EmployeeId = item.CVGReferenceNumber.Substring(0, position);

item.Ticket = item.CVGReferenceNumber.Substring(position);

}

list.Add(item);

}

items++;

}

}

}

catch (Exception ex)

{

//string msgError = ex.ToString();

//ReportError(msgError);

}

return list;

}

// QUICK CONNECT PASSWORD RESET

//=====================================================

private static bool QCSendBroadcastMessage(ItemRequest item)

{

bool sent = false;

try

{

string sendTo = string.Empty;

sendTo = item.EmployeeId;

if (!string.IsNullOrEmpty(sendTo))

{

string subject = "Password reset";

string message = "Your password for " + item.ToolID + " has been reset. Your new password is: " + item.Password;

StringBuilder parameters = new StringBuilder();

parameters.Append("to=");

parameters.Append(sendTo);

parameters.Append("&subject=");

parameters.Append(subject);

parameters.Append("&message=");

parameters.Append(HttpUtility.UrlEncode(message, ASCIIEncoding.ASCII));

//additional parameters

//parameters.Append("&queued=Y");

//parameters.Append("&mustAck=Y");

//parameters.Append("&scroll=Y");

//parameters.Append("&sendPage=Y");

//parameters.Append("&days=");

byte[] buffer = Encoding.ASCII.GetBytes(parameters.ToString());

HttpWebRequest webrequest = (HttpWebRequest)WebRequest.Create(QCBroadcastWSUrl);

webrequest.Credentials = new NetworkCredential(QCUserName, QCPassword);

webrequest.PreAuthenticate = true;

webrequest.Method = "POST";

webrequest.ContentType = "application/x-www-form-urlencoded";

webrequest.ContentLength = buffer.Length;

using (Stream postData = webrequest.GetRequestStream())

{

postData.Write(buffer, 0, buffer.Length);

postData.Close();

}

HttpWebResponse webresponse = null;

try

{

webresponse = (HttpWebResponse)webrequest.GetResponse();

sent = (webresponse.StatusCode == HttpStatusCode.OK);

}

catch (Exception ex)

{

string msgErr = "QCSendBroadcastMessage\\n\\n " + ex.ToString();

ReportError(msgErr);

}

finally

{

if (webresponse != null)

webresponse.Close();

}

}

}

catch (Exception ex)

{

string msgErr = "QCSendBroadcastMessage\\n\\n " + ex.ToString();

ReportError(msgErr);

}

return sent;

}

private static bool QCSendHotPopMessage(ItemRequest item)

{

bool sent = false;

try

{

string sendTo = string.Empty;

sendTo = item.EmployeeId;

if (!string.IsNullOrEmpty(sendTo))

{

string message = QCHotPopMessage;

foreach (var property in item.GetType().GetProperties())

{

message = message.Replace("{" + property.Name + "}", (property.GetValue(item) ?? string.Empty).ToString());

}

StringBuilder parameters = new StringBuilder();

parameters.Append("to=");

parameters.Append(sendTo);

parameters.Append("&msg=");

parameters.Append(HttpUtility.UrlEncode(message, ASCIIEncoding.ASCII));

parameters.Append("&sender=");

parameters.Append(QCUserName);

parameters.Append("&queued=Y");

parameters.Append("&pwdResetFlag=T");

byte[] buffer = Encoding.ASCII.GetBytes(parameters.ToString());

HttpWebRequest webrequest = (HttpWebRequest)WebRequest.Create(QCChatWSUrl);

webrequest.Credentials = new NetworkCredential(QCUserName, QCPassword);

webrequest.PreAuthenticate = true;

webrequest.Method = "POST";

webrequest.ContentType = "application/x-www-form-urlencoded";

webrequest.ContentLength = buffer.Length;

using (Stream postData = webrequest.GetRequestStream())

{

postData.Write(buffer, 0, buffer.Length);

postData.Close();

}

HttpWebResponse webresponse = null;

try

{

webresponse = (HttpWebResponse)webrequest.GetResponse();

sent = (webresponse.StatusCode == HttpStatusCode.OK);

log.Info("HotPopMessage sent to: " + sendTo);

}

catch (Exception ex)

{

// string msgErr = "QCSendHotPopMessag\\n\\n " + ex.ToString();

// ReportError(msgErr);

}

finally

{

if (webresponse != null)

webresponse.Close();

}

}

}

catch (Exception ex)

{

string msgErr = "QCSendHotPopMessage 2\\n\\n " + ex.ToString();

//ReportError(msgErr);

}

return sent;

}

private static bool QCResetCredentials(ItemRequest item)

{

bool sent = false;

try

{

//https://quickconnect.convergys.com/ws/passwordvault/send/resetPassword?

//empId =100004452

//&appName=Application1

//&appUserid="+URLEncoder.encode("{DES}7NfWEZOa0W4=","UTF-8")+"

//&appPwd="+URLEncoder.encode("{DES}7NfWEZOa0W4=","UTF-8")+"

StringBuilder parameters = new StringBuilder();

parameters.Append("empId=");

parameters.Append(HttpUtility.UrlEncode(item.EmployeeId));

parameters.Append("&appName=");

parameters.Append(HttpUtility.UrlEncode(item.ResetType));

parameters.Append("&appUserid=");

parameters.Append(HttpUtility.UrlEncode(PWCrypto.encrypt(item.ToolID)));

parameters.Append("&appPwd=");

parameters.Append(HttpUtility.UrlEncode(PWCrypto.encrypt(item.Password)));

parameters.Append("&loginId=");

parameters.Append(HttpUtility.UrlEncode(QCUserName));

parameters.Append("&loginPwd=");

parameters.Append(HttpUtility.UrlEncode(QCPassword));

byte[] buffer = Encoding.ASCII.GetBytes(parameters.ToString());

HttpWebRequest webrequest = (HttpWebRequest)WebRequest.Create(QCResetWSUrl);

webrequest.Credentials = new NetworkCredential(QCUserName, QCPassword);

webrequest.PreAuthenticate = true;

webrequest.Method = "POST";

webrequest.ContentType = "application/x-www-form-urlencoded";

webrequest.ContentLength = buffer.Length;

#if DEBUG

//to bypass invalid certificates on test site

ServicePointManager.ServerCertificateValidationCallback = delegate (

Object obj,

X509Certificate certificate,

X509Chain chain,

SslPolicyErrors errors)

{

return (true);

};

#endif

using (Stream postData = webrequest.GetRequestStream())

{

postData.Write(buffer, 0, buffer.Length);

postData.Close();

}

HttpWebResponse webresponse = null;

try

{

webresponse = (HttpWebResponse)webrequest.GetResponse();

if (webresponse.StatusCode == HttpStatusCode.OK)

{

using (var reader = new StreamReader(webresponse.GetResponseStream()))

{

string responseText = reader.ReadToEnd();

//get the success text every time, to avoid restarting the service

QCResetSuccessText = ConfigurationManager.AppSettings["QCResetSuccessText"];

if (responseText.Contains(QCResetSuccessText))

{

sent = true;

log.Info("Message from service: " + responseText);

}

else

{

//log the values

log.Info("Request: '" + item.EmployeeId + ", '" + item.ResetType + "', '" + item.Name + "'");

log.Warn("Error message from service: " + responseText);

}

}

}

}

catch (Exception ex)

{

string msgErr = "QCResetCredentials \\n\\n" + ex.ToString();

ReportError(msgErr);

}

finally

{

if (webresponse != null)

webresponse.Close();

}

}

catch (Exception ex)

{

string msgErr = "QCResetCredentials 2 \\n\\n" + ex.ToString();

ReportError(msgErr);

}

return sent;

}

private static bool ComposeEmailAndSend(string address, List<ItemRequest> items)

{

bool sent = false;

try

{

if (items.Count > 0)

{

StringBuilder sb = new StringBuilder();

sb.Append("<p style='font-family:Arial, Helvetica, sans-serif'>");

sb.Append("Dear ");

sb.Append(items.First().TMFirstName); //take it from the first item

sb.Append(",");

sb.Append("<br /><br />");

sb.Append("This is an automated message from Notification Robot.");

sb.Append("<br /><br />");

sb.Append("This mailbox is not monitored. In case of questions or concerns, please send them to Comcast.PasswordReset@Convergys.com.");

sb.Append("<br /><br />");

sb.Append("New passwords have been set up for the accounts listed below:");

sb.Append("</p>");

sb.Append("<table border=1 cellpadding=0 cellspacing=0 style='font-family:Arial, Helvetica, sans-serif;'>");

// headers

sb.Append("<tr>");

sb.AppendFormat("<td>{0}</td>", "Reset Type");

sb.AppendFormat("<td>{0}</td>", "Tool ID");

//sb.AppendFormat("<td>{0}</td>", "Name");

sb.AppendFormat("<td>{0}</td>", "First Name");

sb.AppendFormat("<td>{0}</td>", "Last Name");

sb.AppendFormat("<td>{0}</td>", "CVG Reference Number");

sb.AppendFormat("<td>{0}</td>", "EmployeeId");

sb.AppendFormat("<td>{0}</td>", "Password");

sb.Append("</tr>");

// data rows

foreach (var item in items)

{

sb.Append("<tr>");

sb.AppendFormat("<td>{0}</td>", item.ResetType);

sb.AppendFormat("<td>{0}</td>", item.ToolID);

//sb.AppendFormat("<td>{0}</td>", item.Name);

sb.AppendFormat("<td>{0}</td>", item.FirstName);

sb.AppendFormat("<td>{0}</td>", item.LastName);

sb.AppendFormat("<td>{0}</td>", item.CVGReferenceNumber);

sb.AppendFormat("<td>{0}</td>", item.EmployeeId);

sb.AppendFormat("<td>{0}</td>", item.Password);

sb.Append("</tr>");

}

sb.Append("</table>");

sb.Append("<p style='font-family:Arial, Helvetica, sans-serif'>");

sb.Append("<br />");

sb.Append("In case of questions or concerns, please contact the Command Center.");

sb.Append("</p>");

string subject = "Password reset";

string body = sb.ToString();

string[] sendTo;

sendTo = new[] { address };

sent = SendEmail(subject, body, sendTo);

}

}

catch (Exception ex)

{

string msgErr = "ComposeEmailAndSend \\n\\n" + ex.ToString();

ReportError(msgErr);

}

return sent;

}

private static bool SendEmail(string subject, string body, string to)

{

return SendEmail(subject, body, new[] { to });

}

private static bool SendEmail(string subject, string body, string[] to)

{

bool sent = false;

try

{

if (to.Length > 0)

{

EmailMessage message = new EmailMessage(service);

message.Subject = subject;

message.Body = body;

foreach (var address in to)

{

message.ToRecipients.Add(address);

}

message.SendAndSaveCopy();

sent = true;

}

}

catch (Exception ex)

{

string msgErr = "Send Email " + ex.ToString();

ReportError(msgErr);

}

return sent;

}

private static void ProcessExceptionEmail(Item email, string errorMessage, string sender)

{

try

{

try

{

string ErrorEmailToList = string.Empty;

ErrorEmailToList = ConfigurationManager.AppSettings["ErrorEmailToList"];

if (sender != "") ErrorEmailToList = ErrorEmailToList + ";" + sender;

if (!string.IsNullOrEmpty(ErrorEmailToList))

{

string[] sendTo = ErrorEmailToList.Split(new[] { ';' }, StringSplitOptions.RemoveEmptyEntries);

StringBuilder sb = new StringBuilder();

sb.Append("<p style='font-family:Arial, Helvetica, sans-serif'>");

sb.Append("This is an automated message from Notification Robot.");

sb.Append("<br /><br />");

sb.Append("This mailbox is not monitored. In case of questions or concerns, please send them to Comcast.PasswordReset@Convergys.com.");

sb.Append("<br /><br />");

sb.Append("The request could not be completed successfully. Please check details below. ");

sb.Append("<br /><br />");

sb.Append("<ul style='font-family:Arial, Helvetica, sans-serif' type='disc'>");

string[] errors = errorMessage.Split(new string[] { Environment.NewLine }, StringSplitOptions.RemoveEmptyEntries);

foreach (var error in errors)

{

sb.Append("<li>");

sb.Append(error);

sb.Append("</li>");

}

sb.Append("</ul>");

sb.Append("</p>");

string body = sb.ToString();

EmailMessage message = (EmailMessage)email;

ResponseMessage reply = message.CreateReply(true); //reply to all

reply.Subject = "Notification Robot Service Error - " + message.Subject;

reply.BodyPrefix = body;

//cant reply to Workday

var WorkdayEmail = new EmailAddress("workday@convergys.comm");

if (reply.ToRecipients.Contains(WorkdayEmail))

reply.ToRecipients.Remove(WorkdayEmail);

foreach (var address in sendTo)

{

reply.CcRecipients.Add(address);

}

reply.SendAndSaveCopy();

}

}

catch (Exception ex)

{

//string msgErr = "ProcessExceptionEmail \\n\\n" + ex.ToString();

//ReportError(msgErr);

}

email.Move(ExceptionFolderId);

}

catch (Exception ex)

{

//string msgErr = "ProcessExceptionEmail 2 \\n\\n" + ex.ToString();

//ReportError(msgErr);

}

}

}

class ItemRequest

{

public string ResetType { get; set; }

public string ToolID { get; set; }

public string Name { get; set; }

public string CVGReferenceNumber { get; set; }

public string EmployeeId { get; set; }

public string Ticket { get; set; }

public string Password { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public string TMEmailAddress { get; set; }

public string TMFirstName { get; set; }

}

}

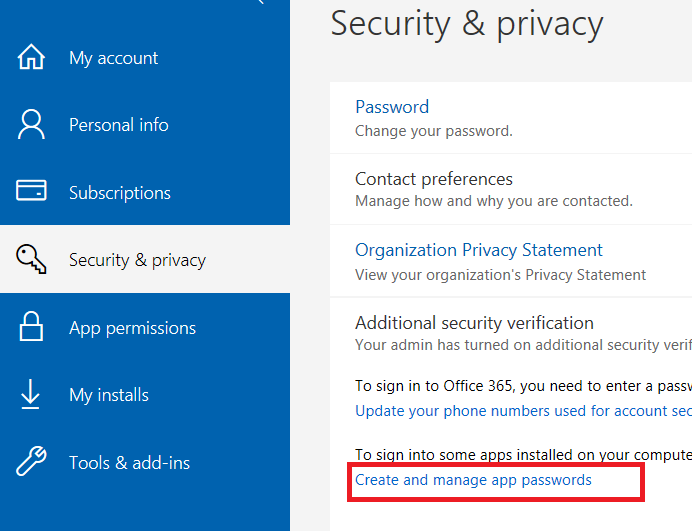
1. Go to <https://portal.office.com/account/>
2. tdal6392@sa.convergys.com

2. Go to Security & privacy

3. Go to Additional security verification

 Or

4. Go to Create and manage app passwords



To include

Install-Package Microsoft.Exchange.WebServices