TMS PassKit
DEVELOPERS GUIDE

August 2019

Web: http://www.tmssoftware.com
Email: info@tmssoftware.com

Index

Getting Started

With TMS PassKit, wallet PassKit files can be generated for use with the iOS / iPhone wallet of all types. Wallet is an application on Apple iOS that manages tickets, vouchers, boarding passes, store cards or other virtual objects. It is a part of the Apple Pay system which supports payment by NFC or Apple Wallet.

To get started with wallet PassKit file generation, add the component TMSPasskitBuilder on the form.

To generate PassKit files, it is required to have an Apple developer account and obtain a certificate from the Apple developer account for the generation of PassKit files.

Features

TMS PassKitBuilder supports the generation of all types of PassKit files.

- Generates tickets, vouchers, boarding passes, store card … for iOS Wallet
- Support for multilanguage PassKit files
- Support for QR code or bar code on PassKit file
- Support for app association / companion app linking
- Control over PassKit file appearance
- Support for PassKit relevance setting including location or beacon based relevance
- Support for PassKit file validity period settings

Getting the certificates

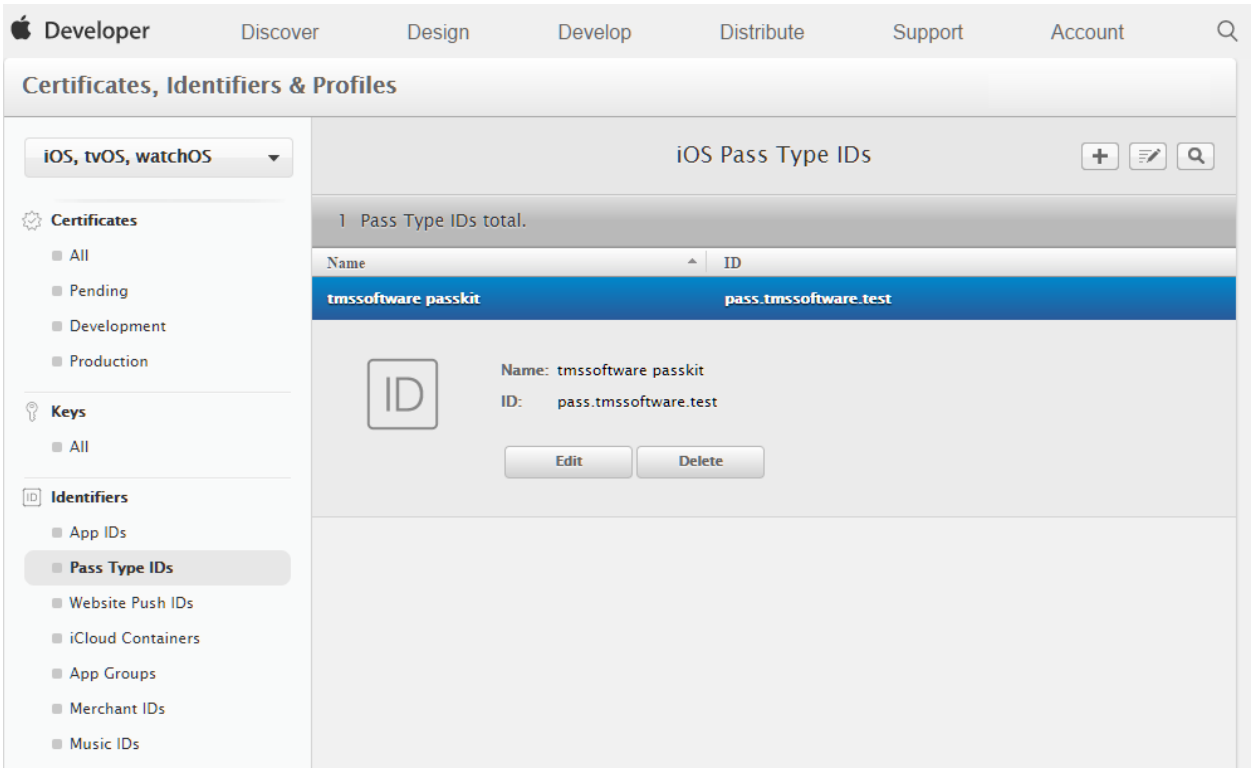
To generate valid PassKit files, it is necessary to obtain & generate the needed certificate files for signing the PassKit file. If the PassKit file is incorrectly signed, the iOS Wallet will refuse to open the file and add it to the iOS Wallet.

Steps:

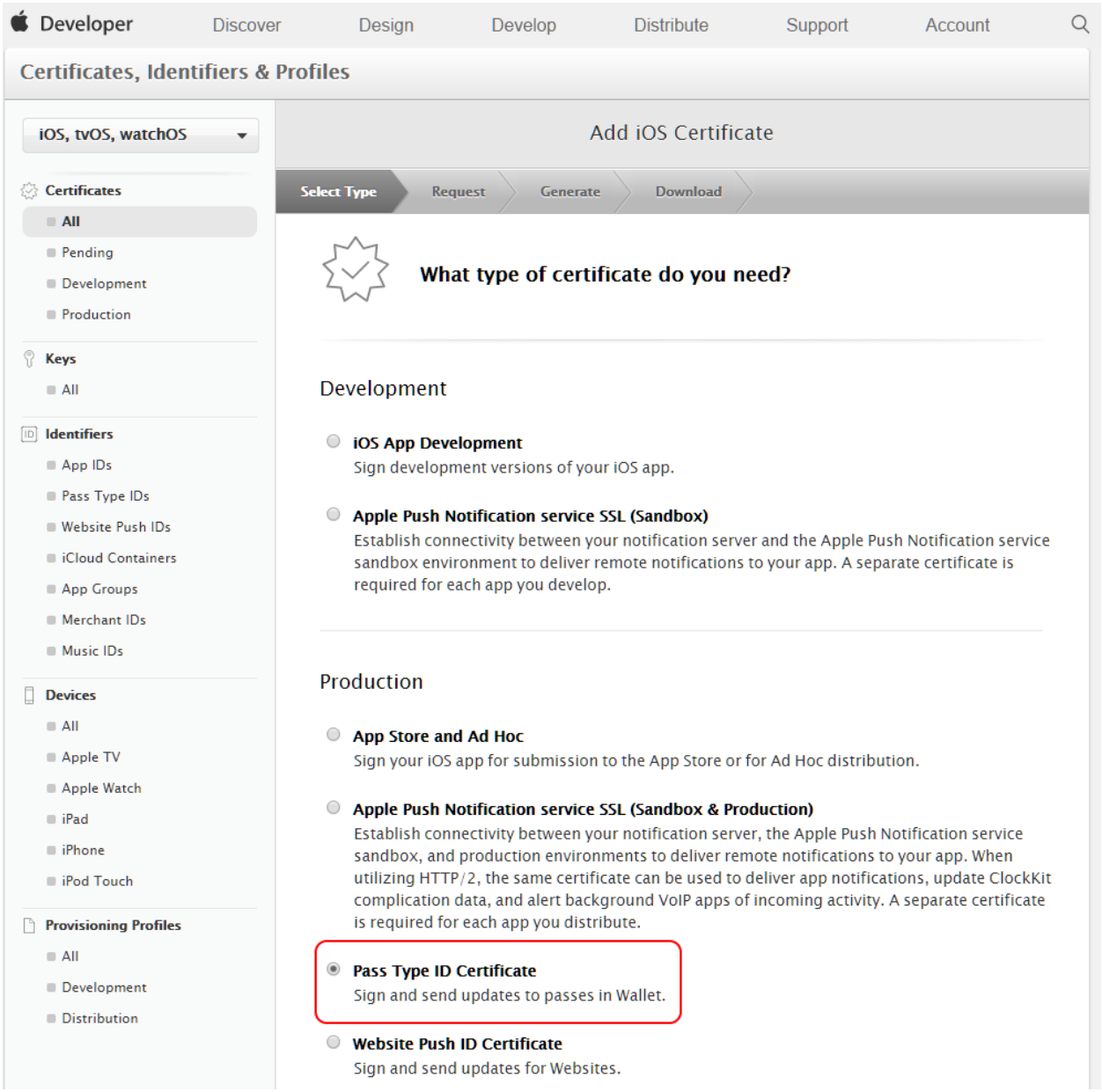
1) Download the Apple World Wide Developer Relations Certificate (WWDR) from

   http://developer.apple.com/certificationauthority/AppleWWDRCA.cer

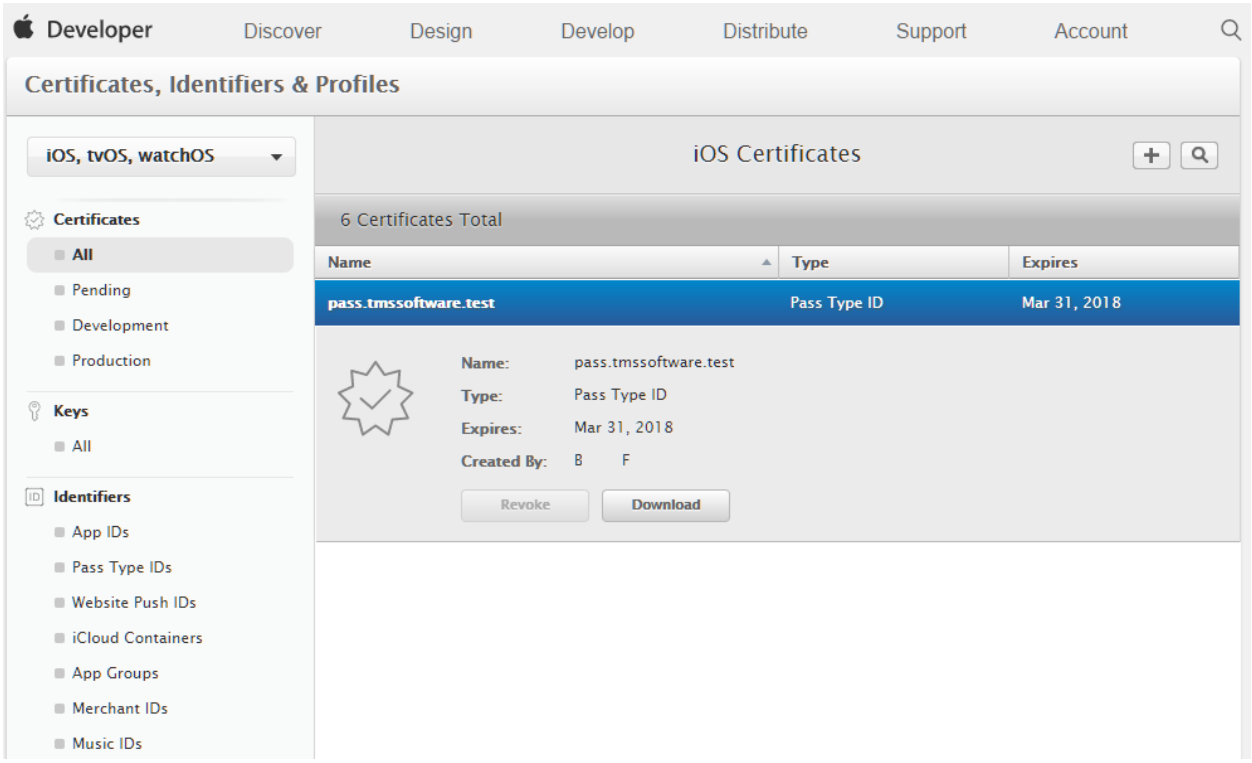
2) Navigate to your Apple developer account and under “Certificates, Identifiers & Profiles”, select Identifiers, “Pass Type IDs” and add a new Pass Type ID. Respect the naming convention from Apple for adding a new Pass Type ID



3) Generate a new PassKit certificate by adding a new certificate of the Pass Type ID type



After the generation of the certificate, it can be downloaded as .CER file:



4) Open the KeyChain Access Manager on OS-X and import the certificate file and let is generate a private key and export it as .p12 type private key file, for example myprivatekey.p12 with the personal password that was asked xxxxx.

5) With these files, you can now convert the certificate .cer file to .pem file for signing PassKit Wallet files. This can be done with the openssl tool.

   If your wallet certificate file is mycompany.wallet.cer

   openssl x509 -inform der -in mycompany.wallet.cer -out mycompany.wallet.pem

6) Convert the private key .p12 file to a .pem file

   openssl pkcs12 -in myprivatekey.p12 -nocerts -nodes -passin pass:xxxxxx | openssl rsa -out myprivatekey.pem

The result is the signing certificate .pem file, the private key .pem file, your private key password and the Apple World Wide developer relations .pem file.

Configuring TTMSPassKitBuilder for signing PassKit files

Set via TMSPassKitBuilder.SigningSetting the files & password for signing:

```
TMSPassKitBuilder.SigningSetting.AppleWwdrCertificate := '.\AppleWWDRCA.pem
TMSPassKitBuilder.SigningSetting.Password := 'xxxxxx';
TMSPassKitBuilder.SigningSetting.PrivateKey := '.\myprivatekey.pem';
TMSPassKitBuilder.SigningSetting.SigningCertificate := '.\mycompany.wallet.pem';
```

The following settings are also needed to be able to generate a valid PassKit wallet file:

Set your company name:

```
TMSPassKitBuilder.OrganizationName := 'your company name';
```

Set a description of the PassKit file

```
TMSPassKitBuilder.Description := 'your PassKit wallet file description';
```

Set the name for the file to be generated:

```
TMSPassKitBuilder.PassName := 'mypasskitfilename';
```

Set the Pass Type ID name as requested from your Apple developer account:

```
TMSPassKitBuilder.PassTypeIdentifier := 'pass.tmssoftware.test';
```

Set your Apple developer account team identifier that is shown on your account details page:

```
TMSPassKitBuilder.TeamIdentifier := 'TEAMIDVALUE';
```

Sets the serial number (can be obtained by inspecting the certificate properties via the KeyChain Access Manager) of your Apple PassKit Type certificate via:

```
TMSPassKitBuilder.SerialNumber := '123.456.789';
```

Note that in order to sign the PassKit file, 2 external DLLs are required. Make sure to deploy the libeay32.dll and ssleay32.dll along with your application executable.

Generating PassKit files with TTMSPassKitBuilder

With all required signing settings completed, the PassKit wallet file can be created by calling:

```
TMSPassKitBuilder1.BuildPass('.\passkitfolder');
```

This will generate in the subfolder .\passkitfolder the PassKit file mypasskitfilename.pkpass and this is the file that can be sent to an iPhone and opened and added to the iOS Wallet.

Defining the appearance and content of the PassKit file

The important elements that define the content and appearance of the PassKit file are:

1) PassKit type

   The type can be set via: TMSPassKitBuilder.PassType

   ptGeneric : not a predefined type
   ptBoardingPass : boarding pass for transport
   ptCoupon : coupon for discount or purchase of something
   ptEventTicket : ticket to participate for an event
   ptStoreCard : card to collect store bonus points for example

2) VisualAppearance

   This holds different settings that control the visual appearance of the wallet file. Note that some visual appearance settings are limited to specific PassKit type files only

   Note that the VisualAppearance images can be language specific.

   See the separate paragraph on VisualAppearance for details about the settings.

3) Fields

   This defines the actual content on the PassKit wallet file. This includes header txt, primary text, secondary text and auxiliary text that go on the front of the wallet file and the fields for the back of the wallet file.

See the separate paragraph on Fields for details about the settings.

Multilanguage support

The languages that are supported in the PassKit wallet file you want to generate can be defined via the stringlist:

TMSPassKitBuilder.Languages: TStringList.

For the languages you wish to support, add the ISO639-1 language code in this stringlist.
See: https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes

To support English and German for example, use

```
TMSPassKitBuilder.Languages.Add('en');
TMSPassKitBuilder.Languages.Add('de');
```

When the supported languages are defined, make sure to add for each language the element in content (Fields) and images (VisualAppearance) by specifying the corresponding language identifier

VisualAppearance

For all images used for the Visual Appearance, it is important to note that a small version and large version of the image file can be specified. The large version is for retina display and is in size thus double width & double height of the normal size image file.

The TMSPassKitBuilder.VisualAppearance class property has following properties:

BackgroundColor: sets the wallet file background color
BackgroundImage: sets the localizable background images (only applicable for the ptEventTicket pass type). This is a collection of images for each supported language. Note that a blur effect will be applied to the background image.

FooterImage: sets the localizable footer images at the bottom of the wallet file. This is a collection of images for each supported language
ForegroundColor: sets the foreground color of the wallet file
GroupingIdentifier:
IconImage: sets the localizable icon images that can optionally be shown at the top of the wallet file
LabelColor: sets the text color of labels on the wallet file
LogoImage: sets the localizable logo images that can be optionally used to add a company logo on the walllet file
LogoText: sets the localizable text that can be used under the company logo
StripImage: sets the localizable images for the optional strip image (only applicable for the ptEventTicket pass type).
SuppressStripShine: when false, the default glow effect applied on the strip image is not used
ThumbnailImage: sets the localizable images for the optional thumbnail image that can be used on the wallet file.

Example:

This adds a different large (for retine displays) strip image for the English and German versions of a wallet file:

```
var
  li: TTMSLocalizablePicture;

  li :=TMSPassKitBuilder1.VisualAppearance.StripImage.Add;
  li.Large.LoadFromFile('.\englishlogo.png');
  li.Language := 'en';

  li :=TMSPassKitBuilder1.VisualAppearance.StripImage.Add;
  li.Large.LoadFromFile('.\germanlogo.png');
  li.Language := 'de';
```

Fields

The different fields define the content on the wallet file.To have localization capabilities, each field is a collection of values used for the different languages supported. A field has a

type and depending on the type, a different value type can be set. The TTMSPassKitField class has following properties:

ChangeMessage: TTMSLocalizableString

Format string for the alert text that is displayed when the pass is updated. The format string must contain the escape %@, which is replaced with the field's new value. For example, "Gate changed to %@."

DataDetectorTypes: TDataDetectorTypes

Specifies what data types the data of a field can be automatically interpreted to. When a data type is detected, this can be used to trigger specific actions when the field is clicked, for example, trigger a phone call when a phone number is detected. By default, all possible data types can be detected: PKDataDetectorTypePhoneNumber, PKDataDetectorTypeLink, PKDataDetectorTypeAddress, PKDataDetectorTypeCalendarEvent

DateField: TTMSPassKitDateField

This has properties:

DateStyle: TDateStyle : specifies how the date is formatted. Values can be: KDateStyleNone, PKDateStyleShort, PKDateStyleMedium, PKDateStyleLong, PKDateStyleFull

IgnoresTimezone: boolean : when true, the timezone is not used in time calculation

IsRelative: boolean : timestamp is defined as relative to current time

TimeStyle: : specifies how the time is formatted

Value: TDateTime : specifies the actual timestamp value

FieldLabel: TTMSLocalizableString

Optional label text that is displayed together with the field value.

FieldType: TTMSPassKitFieldType

This defines the type of the field and can be : TMSpkfString, TMSpkfDate, TMSpkfNumber. When the type is TMSpkfString, the value is set via the StringField property. When the type is

set to TMSpkfNumber, the value is set via the NumberField property and when the type is set to TMSpkfDate, the value is set via the DateField property.

Key: string

Unique key value with which the particular field can be identified

NumberField: TTMSPassKitNumberField

CurrencyCode: string : ISO 4217 currency code for the field's value

NumberStyle: TNumberStyle : specifies the formatting of the number. Settings can be: PKNumberStyleDecimal, PKNumberStylePercent, PKNumberStyleScientific, PKNumberStyleSpellOut

Value: double : actual number value

StringField: TTMSPassKitStringField

This has properties:
- AttributedValue: TTMSLocalizableString : contains a localizable string value with HTML markup that can include hyperlinks. When the AttributedValue is used, the regular Value is ignored.
- Value: TTMSLocalizableString : contains a localizable string value

TextAlignment: TTextAlignment : the values can be: PKTextAlignmentLeft, PKTextAlignmentCenter, PKTextAlignmentRight, PKTextAlignmentNatural

Example:

This inserts a field that contains an expiry date with a localized label text for English and German language:

```
with TMSPassKitBuilder1.Fields.AuxiliaryFields.AddDateField do
begin
  Key := 'expiryDate';
  Value := Now + 30;  // expires within 30 days
  FieldLabel.DefaultValue := 'Valid till'';
  FieldLabel.SetValue('en', 'Valid till');
```

```
  FieldLabel.SetValue('de', 'Gultig bis');
end;
```

Other PassKit file configurations

AssociatedApp

The iTunes Store item identifier(s) can be set via:

TMSPassKitBuilder.AssociatedApp.AssociatedStoreIdentifiers: TList<Integer>

Add the app iTunes Store ID to this list.

The URL that will be passed to the associated app can be set via

TMSPassKitBuilder.AssociatedApp.AppLaunchURL: string;

Barcodes

Collection of possible barcodes to be added to the PassKit wallet file

The TTMSPassKitBarCode collection item has following properties:

AltText: string: the alternative text displayed along with the barcode
BarcodeContent: string : the actual value of the barcode
Encoding: the encoding type used to convert the message from string to data. The default encoding is iso-8859-1
Format: TTMSBarcodeFormat : specifies the possible barcode format. The value can be: PKBarcodeFormatQR, PKBarcodeFormatPDF417, PKBarcodeFormatAztec, PKBarcodeFormatCode128

CompanionApp

UserInfo: string: holds custom information in JSON format that can be passed to the companion app. This is not displayed on the PassKit wallet file

Expiration

ExpirationDate: TDateTime : sets the date and time when the PassKit wallet file is expired.

Voided: boolean : indicates whether the PassKit wallet file is void. For example, when a one-time discount coupon code has been redeemed before expiration, it cannot longer be used and is as such marked as voided.

NFC

Enabled: boolean : Indicates that NFC is enabled for the pass.

EncriptionPublicKey: string : The public encryption key used by the Value Added Services protocol. Use a Base64 encoded X.509 SubjectPublicKeyInfo structure containing a ECDH public key for group P256.

Payload: string : The payload to be transmitted to the Apple Pay terminal. Must be 64 bytes or less. Messages longer than 64 bytes are truncated by the system.

Relevance

Beacons: TTMSPasskitBeaconsCollection : A collection of beacons marking locations where the pass is relevant.

The information per beacon consists of:
- Major, Minor Bluetooth Low Energy location beacon identifier,
- ProximityUUID: string : unique identifier of the beacon
- RelevantText: string : text that can be displayed when the beacon is nearby

Locations: TTMSPasskitLocationsCollection : A collection of geolocations marking locations where the pass is relevant.

The information per location consists of:

Altitidude: double : altitude of the location (in meters)
Longitude: double : longitude in degrees
Latitude: double : latitude in degrees

MaxDistance : Double : Maximum distance in meters from a relevant latitude and longitude that the pass is relevant.

RelevantDate: TDateTime : Date and time when the pass becomes relevant. For example, the start time of a movie.

WebService

Defines URL and token to use to call a webservice when the PassKit wallet file is used.

AuthenticationToken: string : The authentication token to use with the web service. The token must be 16 characters or longer.

URL: string; The URL of a web service that conforms to the API described in PassKit Web Service Reference. The web service must use the HTTPS protocol; the leading https:// is included in the value of this key.

Validation

The TTMSPassKitBuilder component has a built-in method to perform validation and give feedback on the PassKit file settings so it can be ensured that a valid PassKit file is generated from the tool.

To perform validation, the following code can be used:

```
var
  validationResult: TTMSPassKitValidationResult;

begin
  validationResult := TMSPassKitBuilder1.Validate;
  if validationResult.IsValid then
     ShowMessage('Settings for PassKit file are valid');
  FreeAndNil(validationResult);
end;
```

If the result is not valid, the messages in the TTMSPassKitValidationResult object can be retrieved to get detail information about what exact settings are missing or invalid. This can be done with:

```
var
  m: TTMSPasskitValidationMessage;

begin
  for m in validationResult.Messages do
  begin
Memo.Lines.Add(GetEnumName(System.TypeInfo(TTMSValidationMessageSeveri
ty), Ord(m.Severity)) + ': ' + m.Value)
  end;
end;
```

PassKit image file tips

This is the list of possible images that can be added to the PassKit file and notes on where these are used and the recommended image dimensions:

Image Type	Position on the Pass	Other Notes	Dimensions
IconImage	Displayed on the lock screen as part of a notification, and apps like “Mail” when showing that a pass is attached		29 x 29
LogoImage	Displayed in the top left corner of the Pass	The maximum width is 320 pixels. In most cases it should be narrower. Please note, if you use the full pixel width you may not have room to display the Logo Text or Header Fields.	320 x 100
StripImage	Displayed behind the Primary Fields		On iPhone 6 and 6 Plus, the space is 750 x 196 for event tickets, 750 x 288 for gift cards and coupons, and 750 x 246 in all other cases.

			On earlier models, the space is 640 x 168 for event tickets, 640 x 220 for square barcodes on devices with 3.5 inch screens, and 640 x 246 in all other cases.
FooterImage	Displayed directly bove the barcode	Available when using the Transport Pass Type	572 x 30
ThumbnailImage	Displayed next to the fields on the front right of the pass	The aspect ratio should be in the range of 2:3 to 3:2, otherwise the image is cropped	Up to 180 x 180
BackgroundImage	Behind the entire front of the pass	Available when using Event Pass Type. The image is cropped slightly on all sides and blurred	

Example

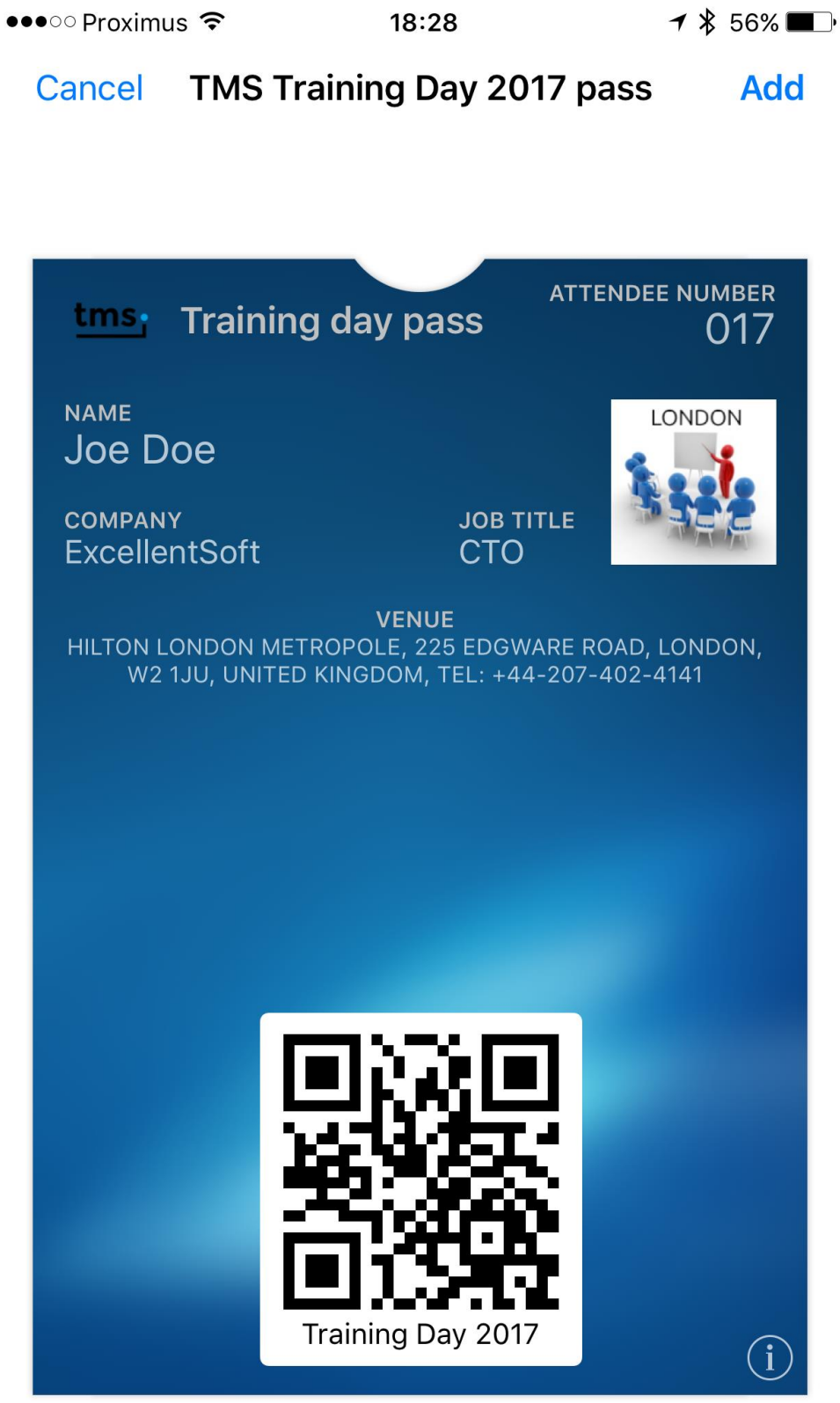
This example code shows how to create an event ticket. The event ticket shows a logo, a logo text, a header field, primary field, secondary field and auxiliary field. On the back of the pass, multiple fields are added.

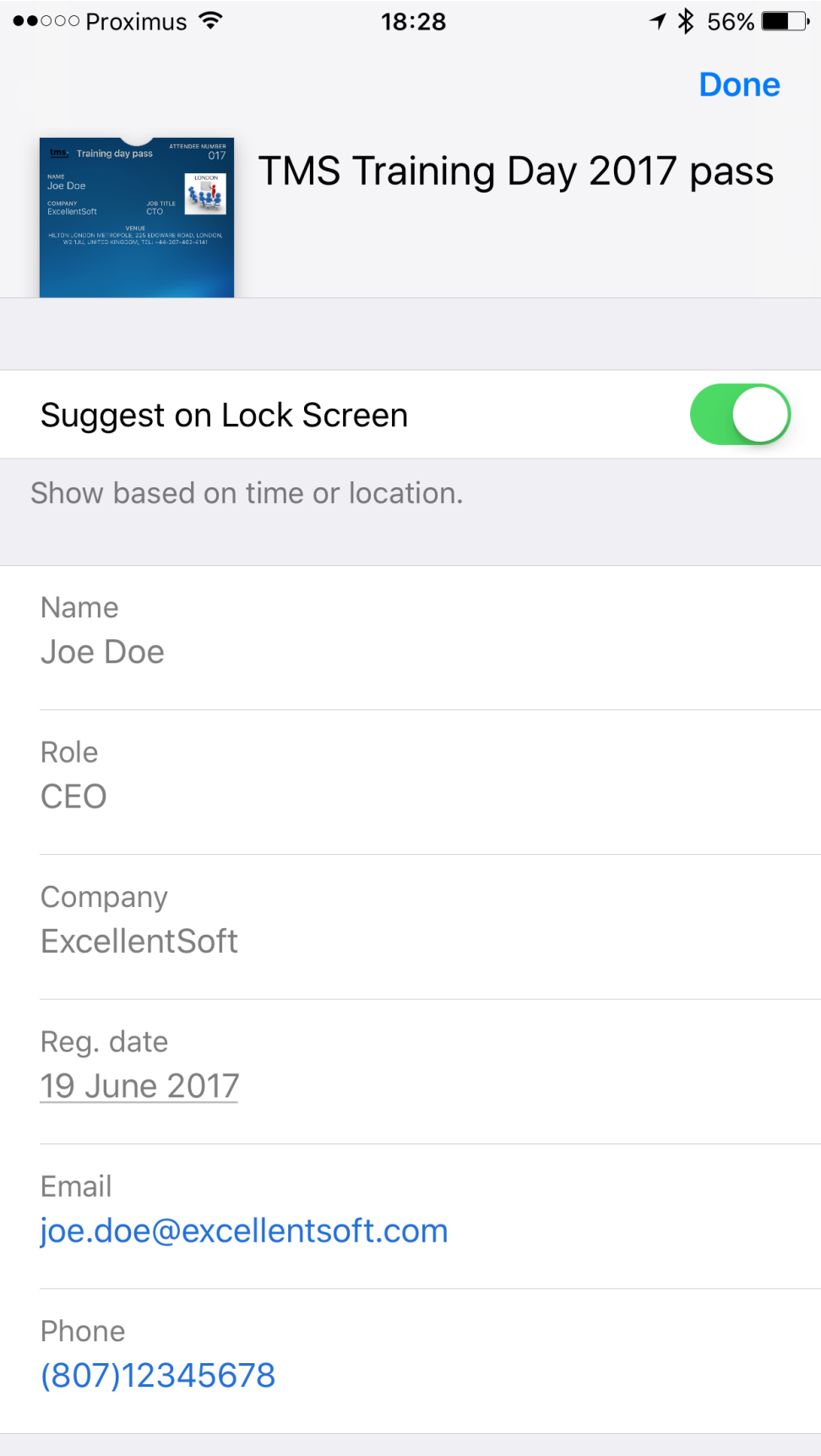
```
var
  m: TMultisizeImage;
  b: TTMSPassKitBarcode;
  s: TTMSTranslation;
begin
  ReportMemoryLeaksOnShutdown := true;

  TMSPassKitBuilder1.PassName := 'tmsday';
  TMSPassKitBuilder1.PassType := ptEventTicket;

  TMSPassKitBuilder1.Description := 'TMS Training Day 2017 pass';
```

```
b := TMSPassKitBuilder1.Barcodes.Add;
b.BarcodeContent := 'TMS Training Day 2017';
b.Format := TTMSBarcodeFormat.PKBarcodeFormatQR;
b.AltText := 'Training Day 2017';

m := TMSPassKitBuilder1.VisualAppearance.BackgroundImage.Add;
m.Language := 'en';
m.Normal.LoadFromFile('.\light-blue-design.jpg');

m := TMSPassKitBuilder1.VisualAppearance.IconImage.Add;
m.Language := 'en';
m.Normal.LoadFromFile('.\tms_logo_100x100.png');
m.Large.LoadFromFile('.\tms_logo_200x200.png');

m := TMSPassKitBuilder1.VisualAppearance.LogoImage.Add;
m.Language := 'en';
m.Normal.LoadFromFile('.\tms_logo_100x100.png');
m.Large.LoadFromFile('.\tms_logo_200x200.png');

s := TMSPassKitBuilder1.VisualAppearance.LogoText.Add;
s.Language := 'en';
s.Value := 'Training day pass';

s := TMSPassKitBuilder1.VisualAppearance.LogoText.Add;
s.Language := 'de';
s.Value := 'Training day Karte';

m := TMSPassKitBuilder1.VisualAppearance.LogoImage.Add;
m.Language := 'de';
m.Normal.LoadFromFile('.\tms_logo_100x100.png');
m.Large.LoadFromFile('.\tms_logo_200x200.png');

m := TMSPassKitBuilder1.VisualAppearance.ThumbnailImage.Add;
m.Language := 'en';
m.Normal.LoadFromFile('.\tms_training_class_en.png');
m.Large.LoadFromFile('.\tms_training_class_en.png');

m := TMSPassKitBuilder1.VisualAppearance.ThumbnailImage.Add;
m.Language := 'de';
m.Normal.LoadFromFile('.\tms_training_class_de.png');
m.Large.LoadFromFile('.\tms_training_class_de.png');

TMSPassKitBuilder1.VisualAppearance.BackgroundColor := clWhite;
TMSPassKitBuilder1.VisualAppearance.ForegroundColor := clBlue;
TMSPassKitBuilder1.VisualAppearance.LabelColor := clSilver;

// Fields on back of pass
//------------------------------------------------------------------------------
with TMSPassKitBuilder1.Fields.BackFields.AddStringField do
begin
```

```
  Key := 'attendeeName';
  StringField.Value.DefaultValue := 'Joe Doe';
  FieldLabel.DefaultValue := 'Name';
end;

with TMSPassKitBuilder1.Fields.BackFields.AddStringField do
begin
  Key := 'attendeeRole';
  StringField.Value.DefaultValue := 'CEO';
  FieldLabel.DefaultValue := 'Role';
end;

with TMSPassKitBuilder1.Fields.BackFields.AddStringField do
begin
  Key := 'attendeeCompanyBG';
  StringField.Value.DefaultValue := 'ExcellentSoft';
  FieldLabel.DefaultValue := 'Company';
end;

with TMSPassKitBuilder1.Fields.BackFields.AddDateField do
begin
  Key := 'regDate';
  DateField.IsRelative := false;
  DateField.IgnoresTimeZone := true;
  DateField.DateStyle := TDateStyle.PKDateStyleLong;
  DateField.Value := EncodeDate(2017, 6 , 19);
  FieldLabel.DefaultValue := 'Reg. date';
end;

with TMSPassKitBuilder1.Fields.BackFields.AddStringField do
begin
  Key := 'attendeeEmail';
  StringField.Value.DefaultValue := 'joe.doe@excellentsoft.com';
  FieldLabel.DefaultValue := 'Email';
end;

with TMSPassKitBuilder1.Fields.BackFields.AddStringField do
begin
  Key := 'attendeePhone';
  StringField.Value.DefaultValue := '(807)12345678';
  FieldLabel.DefaultValue := 'Phone';
end;

// Fields on pass
//------------------------------------------------------------------------------
with TMSPassKitBuilder1.Fields.HeaderFields.AddStringField do
begin
  Key := 'attendeeNumber';
  StringField.Value.DefaultValue := '017';
  TextAlignment := PKTextAlignmentRight;
  FieldLabel.DefaultValue := 'Attendee Number';
```

```
    FieldLabel.AddValue('de', 'Besucher Nummer');
  end;

  with TMSPassKitBuilder1.Fields.PrimaryFields.AddStringField do
  begin
    Key := 'attendee';
    StringField.Value.DefaultValue := 'Joe Doe';
    FieldLabel.DefaultValue := 'Name';
    FieldLabel.AddValue('nl', 'Naam');
  end;

  with TMSPassKitBuilder1.Fields.SecondaryFields.AddStringField do
  begin
    Key := 'attendeeCompanyFG';
    StringField.Value.DefaultValue := 'ExcellentSoft';
    FieldLabel.DefaultValue := 'Company';
    FieldLabel.AddValue('de', 'Gesellschaft');
  end;

  with TMSPassKitBuilder1.Fields.SecondaryFields.AddStringField do
  begin
    Key := 'jobTitle';
    StringField.Value.DefaultValue := 'CTO';
    FieldLabel.DefaultValue := 'Job Title';
  end;

  with TMSPassKitBuilder1.Fields.AuxiliaryFields.AddStringField do
  begin
    Key := 'Address';
    StringField.Value.DefaultValue := 'HILTON LONDON METROPOLE, 225 EDGWARE ROAD,
LONDON, W2 1JU, UNITED KINGDOM, TEL: +44-207-402-4141';
    TextAlignment := TTextAlignment.PKTextAlignmentCenter;
    FieldLabel.DefaultValue := 'Venue';
  end;
```


●●●○○ Proximus
18:28
56%
Cancel
TMS Training Day 2017 pass
Add
tms Training day pass
ATTENDEE NUMBER
017
NAME
Joe Doe
LONDON
COMPANY
ExcellentSoft
JOB TITLE
CTO
VENUE
HILTON LONDON METROPOLE, 225 EDGWARE ROAD, LONDON, W2 1JU, UNITED KINGDOM, TEL: +44-207-402-4141
Training Day 2017


Proximus
18:28
56%
Done
Training day pass
ATTENDEE NUMBER
017
NAME
Joe Doe
COMPANY
ExcellentSoft
JOB TITLE
CTO
LONDON
VENUE
TMS Training Day 2017 pass
Suggest on Lock Screen
Show based on time or location.
Name
Joe Doe
Role
CEO
Company
ExcellentSoft
Reg. date
19 June 2017
Email
joe.doe@excellentsoft.com
Phone
(807)12345678