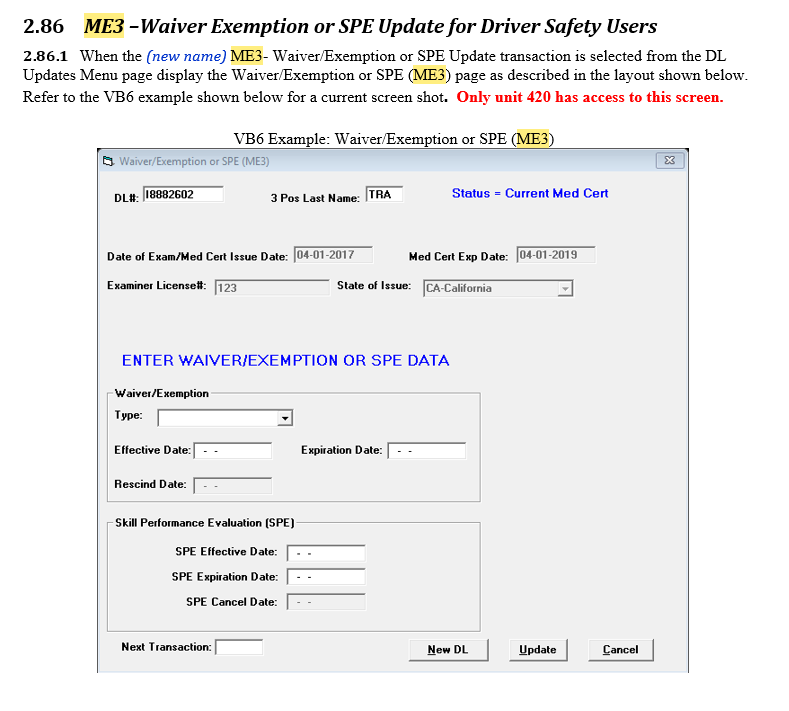
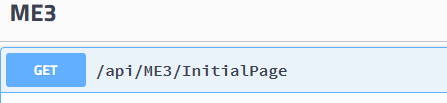
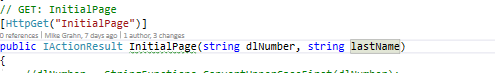
TCODE ME3 – Waiver Exemption or SPE Update for Driver Safety Users







When calling the endpoint InitialPage for ME3, pass in the DL Number and lastname

The endpoint will call the ME1 Tcode in the background to retrieve MED Cert data if it exists.

If ME1 data is not found, the message in the ME3INITDTO will have the verbiage;

Medical and Self Certificate Information: NONE

With a return code of 200. In the ME3INITDTO

ME3INITDTO….



If ME1 data exists, the above DTO will be populated with the resulting data.

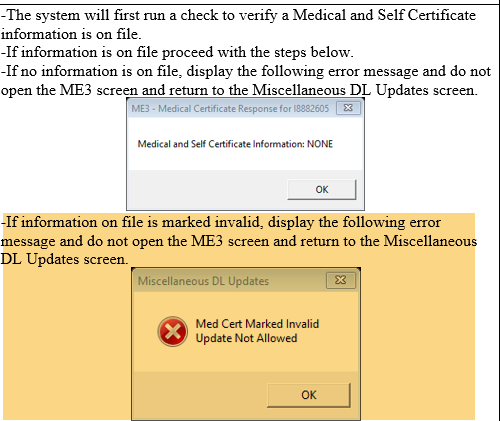
There are two statuses for a MED CERT:

I = Med Cert Marked Invalid

N = Non Current Med Cert

Else

Current Med Cert



The statuses will be populated in the following fields in the ME3INITDTO after ME1 retrieval in the INIT Endpoint:



Status and Status message are populated as follows

if (dto.Status == "I")

{

dto.StatusMessage = "Status - Invalid Med Cert";

}

else if (dto.Status == "N")

{

dto.StatusMessage = "Status - Non Current Med Cert";

}

else

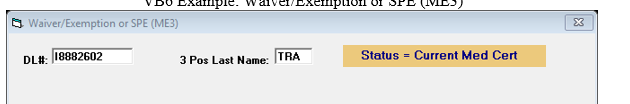
{

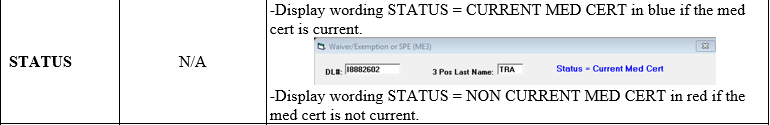
dto.StatusMessage = "Status - Current Med Cert";

}

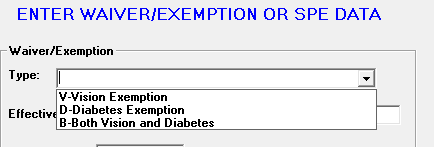
If Status = “I” - invalid, disable Update , NewDL and next Trans buttons, Med Cert cannot be updated.

Display the Status Message if it is populated at:

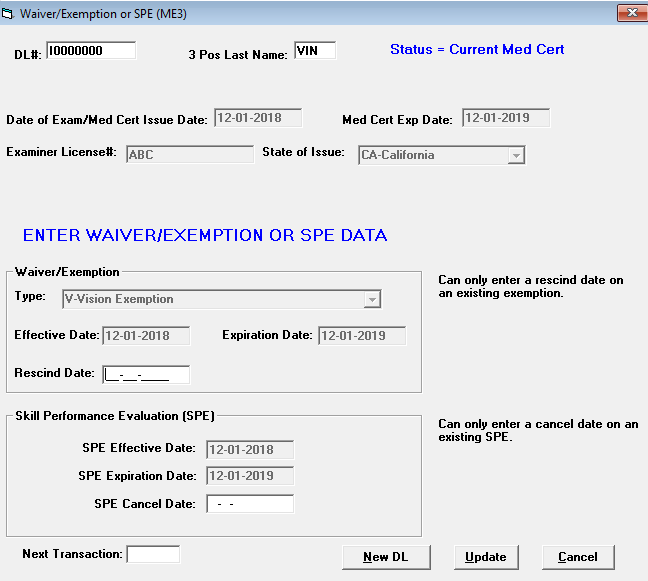


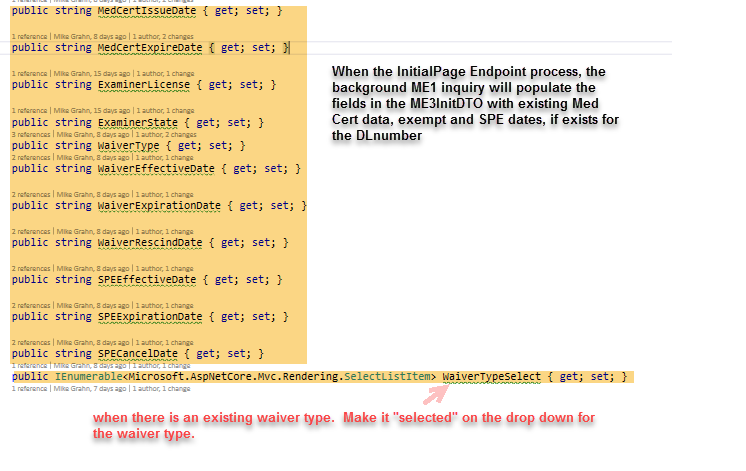


The waiver/exempt type will be populated in the endpoint

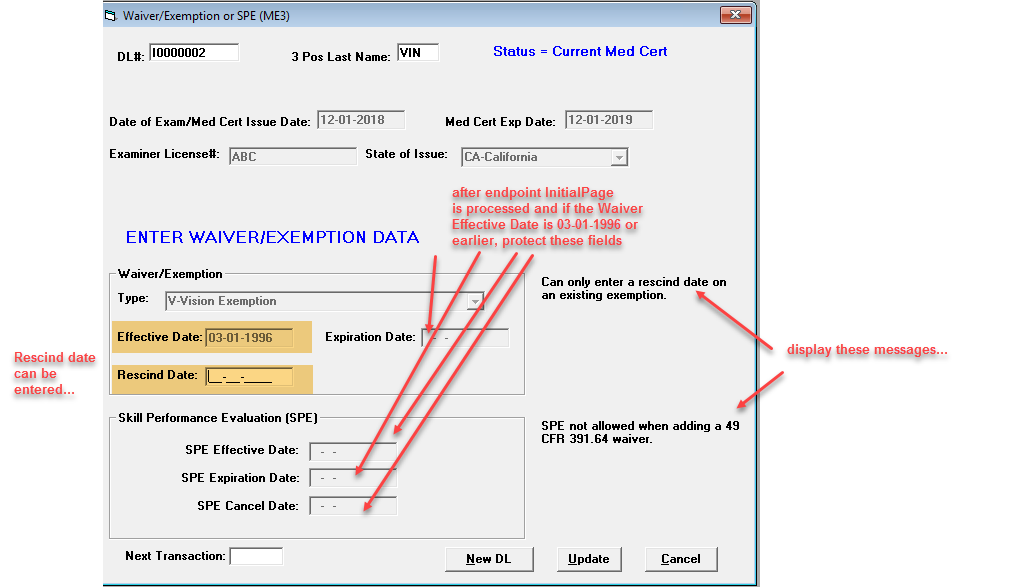


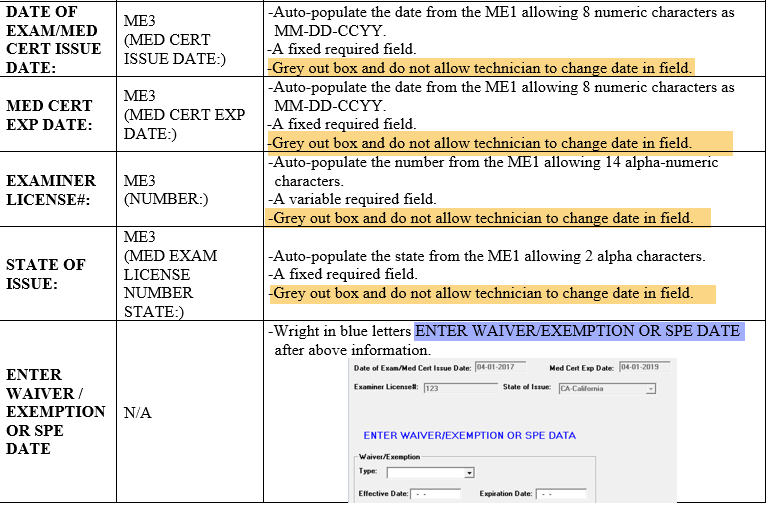
When the InitialPage endpoint is processed it will populate existing data to the screen. All fields that are populated will be protected against input. Cannot be modified.

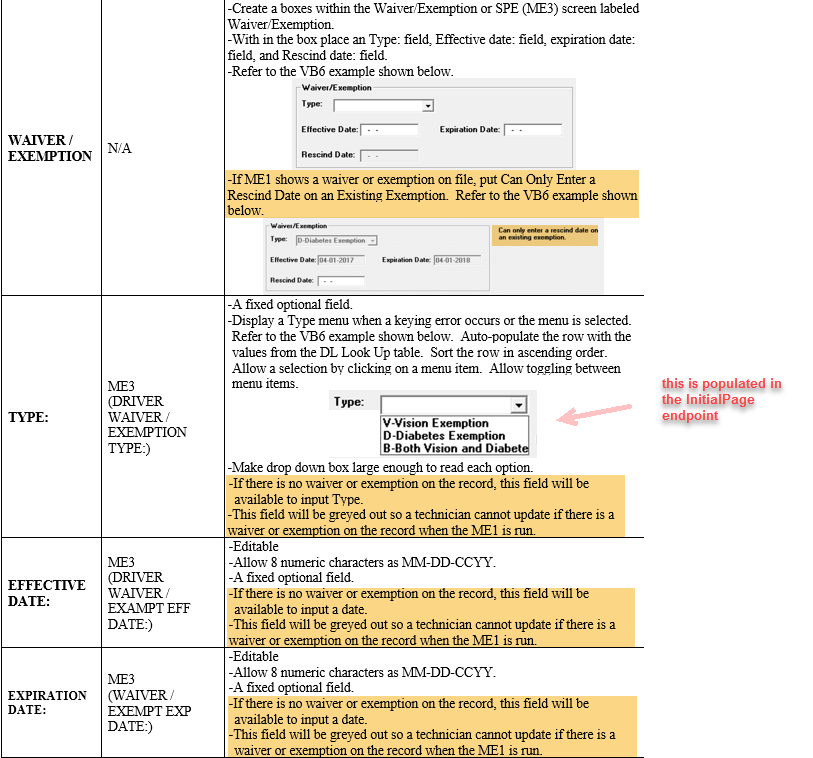


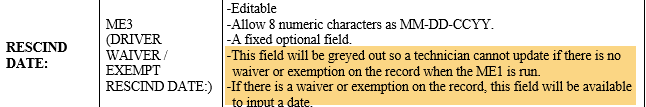


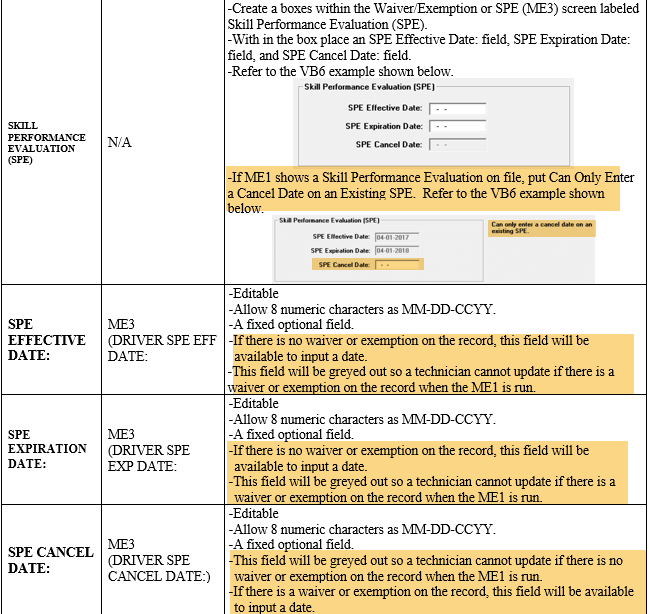
Special Rules:

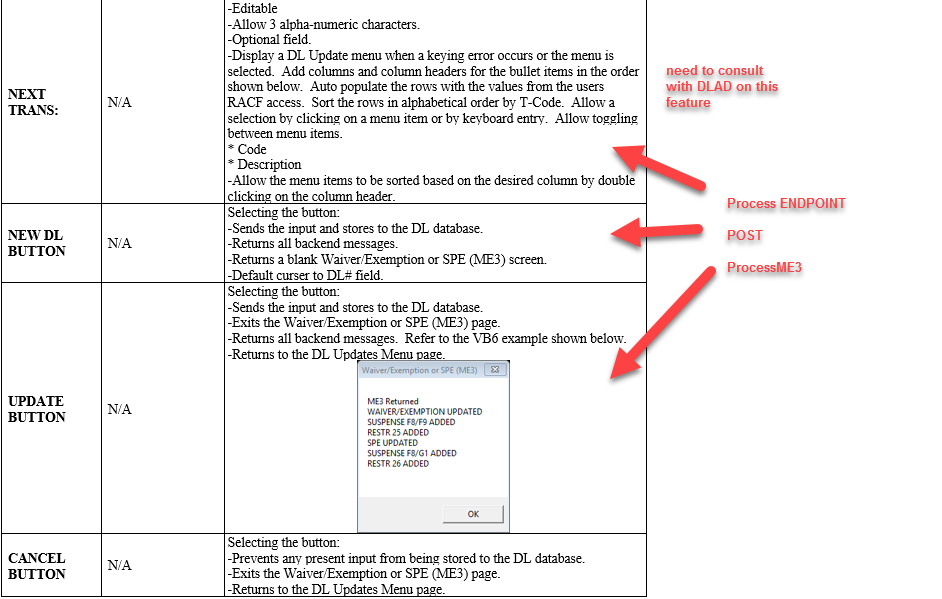




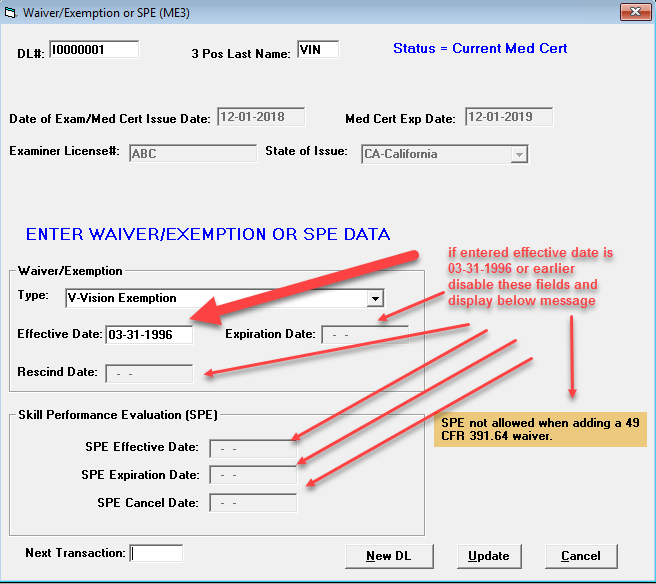








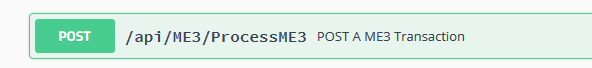
Special rule:

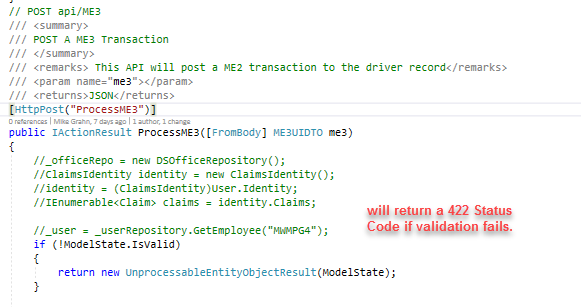


If Status = “I” - invalid, disable Update , NewDL and next Trans buttons, Med Cert cannot be updated.

Upon processing either the Update button, Next trans button or NewDL button,

Process endpoint ProcessME3





The endpoint PROCESSME3 uses the DTO ME3UIDTO

Fields and validations as follows:

public class ME3UIDTO : IValidatableObject

{

public string LoginId { get; set; }

public string NetName { get; set; }

[Required]

public string RequestorCode { get; set; }

[Required]

public string Operator { get; set; }

[Required]

public string DLNumber { get; set; }

[Required]

[StringLength(3)]

public string ThreeCharacterLastName { get; set; }

public string Status { get; set; }

public string StatusMessage { get; set; }

[Required]

[Display(Name = "Med Cert Issue Date")]

[DataType(DataType.Date)]

public DateTime MedCertIssueDate { get; set; }

[Required]

[Display(Name = "Med Cert Expire Date")]

[DataType(DataType.Date)]

public DateTime MedCertExpireDate { get; set; }

[Required]

[StringLength(14)]

[Display(Name = "Examiner License #")]

public string ExaminerLicense { get; set; }

[Required]

[StringLength(2)]

[Display(Name = "State of Issue")]

public string ExaminerState { get; set; }

[StringLength(1)]

[Display(Name = "Type")]

public string WaiverType { get; set; }

[Display(Name = "Effective Date")]

public string WaiverEffectiveDate { get; set; }

[Display(Name = "Expiration Date")]

public string WaiverExpirationDate { get; set; }

[Display(Name = "Rescind Date")]

public string WaiverRescindDate { get; set; }

[Display(Name = "SPE Effective Date")]

public string SPEEffectiveDate { get; set; }

[Display(Name = "SPE Expiration Date")]

public string SPEExpirationDate { get; set; }

[Display(Name = "SPE Cancel Date")]

public string SPECancelDate { get; set; }

public string ME3Response { get; set; }

public string NextTran { get; set; }

private string ErrorMessage { get; set; }

public bool Error { get; set; }

public IEnumerable<ValidationResult> Validate(ValidationContext validationContext)

{

int result;

DateTime tempDate;

string strDate;

DateTime date1;

DateTime date2;

if (DLNumber.Length < 8)

{

yield return new ValidationResult(

$"Please enter a valid DL Number",

new[] { "DLNumber" });

}

if (ThreeCharacterLastName.Length < 3)

{

yield return new ValidationResult(

$"The name is not in the correct format",

new[] { "ThreeCharacterLastName" });

}

if (!string.IsNullOrEmpty(WaiverEffectiveDate))

{

DateFunctions.ConvertStringToDate(WaiverEffectiveDate, "MMddyyyy", out date1);

result = DateFunctions.CompareDates(date1, DateTime.Now);

if (result > 0)

{

yield return new ValidationResult(

"Waiver Effective Date cannot be greater than today",

new[] { "Waiver Effective Date" });

}

}

if (WaiverEffectiveDate != null && WaiverExpirationDate != null)

{

DateFunctions.ConvertStringToDate(WaiverEffectiveDate, "MMddyyyy", out date1);

DateFunctions.ConvertStringToDate(WaiverExpirationDate, "MMddyyyy", out date2);

result = DateFunctions.CompareDates(date1, date2);

if (result > 0)

{

yield return new ValidationResult(

"Waiver Expiration Date cannot be before the Waiver Effective Date",

new[] { "Waiver Expiration Date" });

}

}

if (SPEEffectiveDate != null)

{

DateFunctions.ConvertStringToDate(SPEEffectiveDate, "MMddyyyy", out date1);

result = DateFunctions.CompareDates(date1, DateTime.Now);

if (result > 0)

{

yield return new ValidationResult(

"Driver SPE Effective Date cannot be greater than today",

new[] { "SPE Effective Date" });

}

}

if (SPEEffectiveDate != null && SPEExpirationDate != null)

{

DateFunctions.ConvertStringToDate(SPEEffectiveDate, "MMddyyyy", out date1);

DateFunctions.ConvertStringToDate(SPEExpirationDate, "MMddyyyy", out date2);

result = DateFunctions.CompareDates(date1, date2);

if (result > 0)

{

yield return new ValidationResult(

"SPE Expiration Date cannot be before the SPE Effective Date",

new[] { "SPE Expiration Date" });

}

}

if (Status == "I")

{

yield return new ValidationResult(

"Cannot update an invalid certificate",

new[] { "Invalid Status" });

}

if (!string.IsNullOrEmpty(WaiverType) && WaiverEffectiveDate == null)

{

yield return new ValidationResult(

"Waiver Effective Date must be present",

new[] { "Waiver Effective Date" });

}

}