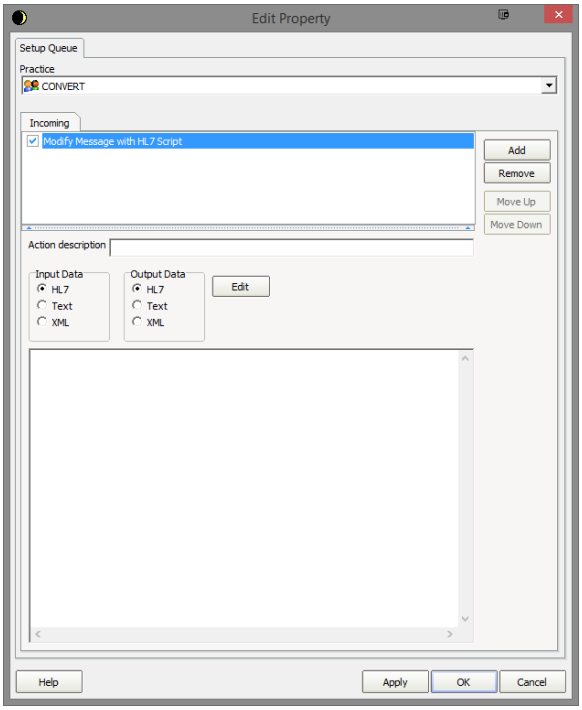
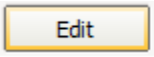
Extra Scripts

Extra Scripts

Add the modifier "**Modify Message with HL7 Script**" in this window you can add Pascal Script to change your messages

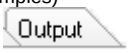


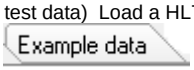
How to use Pascal Script



Medical-Objects Transformation IDE 

write codes(refers to the following examples) 


(open HL7 test data) Load a HL7 test data 
(execute script) 

check Input/Output Message in 

Script Examples

Add TimeZone

```
program AddTimeZone;


begin
   HL7Data['MSH.6.0'] := AddSpecificTimezone(HL7Data['MSH.6.0'], '+0800');
   HL7Data['OBR.6.0'] := AddLocalTimezone(HL7Data['OBR.6.0']);
end.
```

Replace TimeZone

```
program FixTimeZone;


begin
    HL7Data['MSH.6.0'] := HL7TSChangeTimeZoneToLocal(HL7Data['MSH.6.0']);
    HL7Data['OBR.6.0'] := HL7TSChangeTimeZone(HL7Data['OBR.6.0'], '+1200');
    HL7Data['OBR.7.0'] := HL7TSChangeTimeZone(HL7Data['OBR.7.0'], '+1100');
end.
```

Replace GUID

```
program FixGUID;

begin
    if(HL7Data['MSH.3.0'] = 'Noosa Radiology') and (HL7Data['MSH.3.0.2'] =
'649C452B-5EAA-4D8F-A230-F3A54F99A9D4') then
       HL7Data['MSH.3.1'] := 'Practice Name^F2107173-8A7B-4C5D-A055-
CB5C79C76B7E^GUID';

end.
```

This program will find the specific GUID in the Sending Facility and replace it with the GUID of your choice.

Copy OBR16 to PV1-9 if PV1-9 target not set

```
{
  This program copies the provider number from OBR-16 to PV1-9. If PV1
doesnt exist, it will insert it.
  Only copy the components over if a target ID is not already set in PV1.
Try to make it future proof
  incase the input messages contain PV1-9 values.
}


Program AddPV19;
var
  i:integer;
  mshindex: integer;
  pv1index:integer;
  pidindex:integer;
  targetid: string;
  j: integer;

begin
   pv1index :=-1;
   while i < HL7Data.segmentcount -1 do
   begin
       if HL7Data.segmentname[i] = 'MSH' then
       begin
           mshindex := i;
           pv1index :=-1;
           pidindex := -1;
       end
       else if HL7data.segmentname[i] = 'PID' then
           pidindex := i
       else if HL7Data.segmentname[i] = 'PV1'
then
           pv1index :=i
       else if ((HL7Data.segmentname[i] = 'OBR') or (HL7Data.segmentname
[i] = 'ORC')) and (pv1index=-1) then
       begin
           HL7Data.insertsegment (PIDINDEX+1, 'PV1');
           pv1index := PIDINDEX+1;
           inc(i);
       end;

       if (HL7Data.segmentname[i] = 'OBR') and (pv1index<>-1) then
       begin
         if HL7Data.GetFieldvalue(pv1index, 9, 0, 0, 0)=  then
         begin
           targetid := HL7Data.GetFieldvalue(i, 16, 0, 0, 0);
           HL7Data.SetFieldvalue(targetid, pv1index, 9,0,0,0);
           for j := 1 to 6 do
             HL7Data.SetFieldvalue(HL7Data.GetFieldvalue(i, 16, 0, j, 0),
pv1index, 9,0,j,0);
         end;
       end;

       inc(i);
   end
end.
```

Available Pascal Scripts

```
function HL7TSChangeTimeZoneToLocal(const AHL7TS: string): string;
```

Convert input TimeStamp(AHL7TS) to TimeStamp with local TimeZone

```
function HL7TSChangeTimeZone(const AHL7TS: string; const ANewTimeZone:
string): string;
```

Convertinput TimeStamp(AHL7TS) to TimeStamp with input TimeZone(ANewTimeZone)

```
function AddSpecificTimezone(const AHL7TS, ATimeZone: String): string;
```

Add specific TimeZone information(ATimeZone) to HL7 TimeStamp(AHL7TS)

```
function AddLocalTimezone(const AHL7TS: String): string;
```

Add local TimeZone information to HL7 TimeStamp(AHL7TS)

```
function StripTimezone(const AHL7TS: String): string;
```

Get and return HL7 TimeStamp without TimeZone information

```
function HL7TimeZoneToBiasMinutes(const HL7TimeZone: string): Integer;
```

Calculate bias minutes based on input TimeZone(HL7TimeZone)

```
function AgeInYears(ADateOfBirth, ADateOfAge: TDateTime): Integer;
```

Calculate the Age(based on ADateOfBirth and ADateOfAge) in years

```
function AgeToDisplayString(ABirthDate, AAgeAtThisDate: TDateTime): string;
```

Converts directly a Age to a display string

```
function AutoCompleteDate(const AText: string; AddTime: Boolean = False):
string;
```

Adds the next most significant date part with separator

```
function DateStrToHL7DT(const AEditText: String; AAccept2DigitYears:
Boolean = True; AddTimestamp:Boolean = True): string;
```

Convert a user-input date to a HL7 date string

```
procedure ScanDateStr(const ADateStr: string; out ADay, AMonth, AYear:
string);
```

Given a delimited date string, returns the day, month and year parts

```
function HL7DTDecode(const AHL7DT: string; out AYear, AMonth, ADay:
Integer): Boolean;
```

Decode a HL7 date into the year, month and day parts

Returns -1 for parts which are not present. If not even a year is present, returns False otherwise True

```
function HL7DTEncode(AYear, AMonth, ADay: Integer): string;
```

Encode a HL7 date from the input AYear, AMonth and ADay parts

```
function HL7TMDecode(const AHL7TM: string; out AHour, AMinute, ASecond:
Integer; out AFractionalSecond: Double): Boolean;
```

Decode a HL7 time into the hour, minute and second parts (with fractional seconds if it exists)

```
function HL7TMEncode(AHour, AMinute: Integer; ASecond: Integer = -1;
AFractionalSecond: Double = -1): string;
```

Encode a HL7 time from the AHour, AMinuteand and ASecond parts (with fractional seconds if it exists)

```
function HL7TSStripTimezone(const AHL7TS: string): string;
```

Return a HL7 TimeStamp without TimeZone

```
function HL7TMStripTimezone(const AHL7TM: string): string;
```

Return a HL7 Time without TimeZone

```
function HL7TSToLocalTS(const AHL7TS: string; AIncludeTimeZone: Boolean =
True): string;
```

Convert a HL7 TimeStamp to local TimeStamp (with or without TimeZone)

```
function HL7LocalTimeZone: string;
```

Get and return local TimeZone

```
function HL7TimeZoneFromTS(const AHL7TS: string): string;
```

Get and return TimeZone information from the input TimeStamp(AHL7TS)

```
function HL7TimeZoneFromTM(const AHL7TM: string): string;
```

Get and return TimeZone information from the input Time(AHL7TM)

```
function HL7TMGetFractionalSecond(const AHL7TM: string; out
AFractionalSecond: Double): Boolean;
```

Get fractional parts of second and return Boolean value(false or true)

```
function HL7TSEncode(AYear: Integer;
AMonth: Integer = -1;
ADay: Integer = -1;
AHour: Integer = -1;
AMinute: Integer = -1;
ASecond: Integer = -1;
AFractionalSecond: Double = -1.0): string;
```

Encode a HL7 TimeStamp from the input AYear, AMonth, ADay, AHour, AMinute, ASecond and AFractionalSecond parts

```
function HL7TSAddDay(const ADateStr: string; AIncrement: Integer = +1):
string;
```

Add days(AIncrement) to Input(ADateStr) and return the result

Takes a HL7 date (or date/time) input and returns a date (or date/time)

```
function HL7TSAddMonth(const ADateStr: string; AIncrement: Integer = +1):
string;
```

Add months(AIncrement) to Input(ADateStr) and return the result

```
function HL7TSAddYear(const ADateStr: string; AIncrement: Integer = +1):
string;
```

Add years(AIncrement) to Input(ADateStr) and return the result

```
function HL7TSToTDateTime(AHL7DateTime: string): TDateTime;
```

Convert a HL7 TimeStamp(AHL7DateTime) to TDateTime type

```
function HL7DTToTDateTime(AHL7Date: string): TDateTime;
```

Convert a HL7 Date(AHL7Date) to TDateTime type

```
function HL7TMToTDateTime(AHL7Time: string): TDateTime;
```

Convert a HL7 Time(AHL7Time) to TDateTime type

```
function HL7TSToSQLText(const AHL7TS: string): string;
```

Convert a HL7 TimeStamp(AHL7TS) to SQL Text('dd/mm/yyyy')

```
function HL7DTToDisplayString(const AHL7DT: string; const ADisplayFormat:
string = 'd?m?yyyy'): string;
```

Convert directly from a HL7 Date to a display string

```
function HL7TMToDisplayString(const AHL7TM: string; const ADisplayFormat:
string = 'hh:nn'): string;
```

Convert directly from a HL7 Time to a display string

```
function HL7TSToDisplayString(const AHL7TS: string; const ADisplayFormat:
string = 'd.m.yyyy hh:nn'): string;
```

Convert directly from a HL7 TimeStamp to a display string

```
function HL7TSToReverseGregorian(const AHL7TS: string): string;
```

Convert HL7 TimeStamp(AHL7TS) to Reverse Gregorian('yyyymmddhhnnsszzz')

```
function HL7TSToHL7DT(const AHL7TS: string): string;
```

Convert HL7 TimeStamp(AHL7TS) to HL7 Date

```
function HL7TSToHL7TM(const AHL7TS: string): string;
```

Convert HL7 TimeStamp(AHL7TS) to HL7 Time

```
function TDateTimeToHL7DT(AInput:TDateTime): string;
```

Convert TDateTime(AInput) to HL7 Date

```
function HL7TSToDatabaseText(const AHL7TS: string): string;
```

Convert HL7 TimeStamp(AHL7TS) to Database Text('yyyy-mm-dd hh:mm:ss')

```
function HL7TSToTDateTimeDef(AHL7TS: String; Default:TDateTime): TDateTime;
```

Convert a HL7 TimeStamp(AHL7TS) to TDateTime with Default value

```
function MedicalDirectorDateToHL7DT(MD_Date:String): string;
```

Convert Medical Director Date to HL7 Date

```
function LongMonthStringtoMonth(const AMonth:String): Word;
```

Convert month string to month integer

```
function MinutesToHL7TM(const Minutes:Integer): string;
```

Convert minutes(Minutes) to HL7 Time

```
function HL7TMtoMinutes(const HL7TM:String): Integer;
```

Convert HL7 Time(HL7TM) to minutes

```
function TDateTimeToHL7DateTimeAutoPrecision(AInput: TDateTime;
AIncludeTimeZone: Boolean = False): string;
```

Convert TDateTime(AInput) to HL7 DateTime with auto precision step

```
function HL7TMtoOBXDisplayString(const AHL7TM:String): string;
```

Convert directly from a HL7 Time to a OBX display string

```
function XMLDateTimeToHL7TS(XMLDateTime: String): string;
```

Convert XML DateTime to HL7 TimeStamp

```
function DateTimeXMLToString(ADate: TDateTime): string;
```

Convert the TDateTime ADate to a string according to the W3C date/time specification

```
function HL7DateToXMLFormat(ADate: String): string;
```

Convert HL7 Date(ADate) to XML format

```
function HL7DTAddDays(const AHL7DT: string; Days :integer): String;
```

Add days(Days) to Input(AHL7DT) and return the result

```
function IsHL7TSValid(const HL7Date: string): Boolean;
```

Check if HL7 TimeStamp is valid

```
function ActualBiasMinutes: Integer;
```

Calculate an actual Bias Minutes based on the status of TimeZone

var

```
HL7Data: THL7Scriptable
```

```
THL7Scriptable = class(TObject)
public
constructor Create;
destructor Destroy; override;
procedure Clear;
function ComponentCount(const SegmentIndex: Integer; const FieldNo:
Integer;
const RepeatNo: Integer = 0): Integer;
function SubComponentCount(const SegmentIndex: Integer; const FieldNo:
Integer;
const RepeatNo: Integer ; const ComponentNo: Integer): Integer;
procedure DeleteRepeat(const SegmentIndex: Integer; const FieldNo: Integer;
const RepeatNo: Integer );
procedure DeleteSegment(const Index: Integer);
function FieldCount(const SegmentIndex: Integer): Integer;
function GetFieldvalue(const SegmentIndex, FieldNo: Integer; const
RepeatNo:
Integer = 0; const ComponentNo: Integer = 0; const SubComponentNo: Integer
= 0): string;
function IndexofSegment(const SegmentName: String): Integer;
procedure SetFieldvalue(const Value: String; const SegmentIndex, FieldNo:
Integer; const RepeatNo: Integer = 0; const ComponentNo: Integer = 0; const
SubComponentNo: Integer = 0);
procedure InsertSegment(const Index:Integer; const SegmentName: String);
procedure LoadFromFile(const AFilename: String);
procedure LoadFromStream(AStream: TStream);
function RepeatCount(const SegmentIndex: Integer; const FieldNo: Integer):
Integer;
procedure SaveToFile(const AFilename: String);
function SegmentCount: Integer;
property asString: string read GetasString write SetasString;
property Raw: Boolean read GetRaw write SetRaw;
property SegmentName[Index: Integer]: string read GetSegmentName;
property Value[Index: String]: string read GetValue write SetValue;
default;
end;
```

There are more functions. Contact the helpdesk for further assistance.