UFH LAB2ACM interface

**Document Modifications**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Description of Change | Modified By |
| 0.1 | 2011-3-3 | Generate draft | Peng Qiao |
| 1.0 | 2011-3-15 | Interface go live | Peng Qiao |
| 1.1 | 2011-4-26 | Add more features according to ACM req | Peng Qiao |
| 1.2 | 2011-5-16 | Change OBX segment according to ACM req | Peng Qiao |
| 1.2.1 | 2011-7-14 | ACM asked: Pregnancy(code is B4141) with value = “[POSITIVE](http://192.168.0.95:20023/csp/inte_acm/HL7/HL7SchemaSegmentStructure.csp?SS=SS:2.3:OBX#5)” output AbnormalFlags as “H”. | Peng Qiao |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Overview

UFH LAB2ACM interface transfers lab test result of specific patients from LABTrak to ACM.

LABTrak is configured to generate lab test result file in HL7 v2.3 format, Ensemble will be used to read it and do transform and then send it to ACM in HL7 v2.3 format.



## HL7 specification

Input Message specification

See attachment doc: “IN-005-NHS Requirement Specification - Interface.doc”

Output Message specification

See attachment doc: “”

## Source HL7

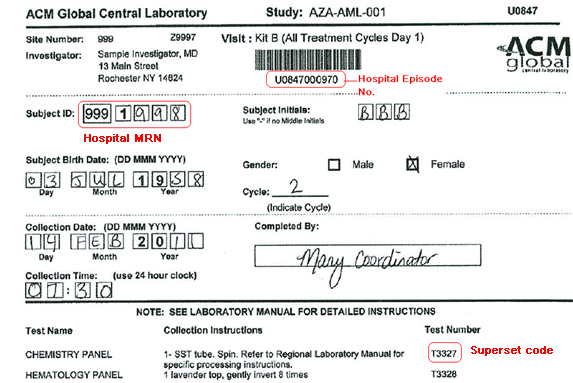
LabTrak was configured to generate HL7 file automatically.

When user authorizes or de-authorizes test result, LabTrak will generate corresponding HL7 message in file. But when user de-authorizes test result, LabTrak will generate ORU\_R01 without OBX segment, it’s not following HL7 standards, so Ensemble will throw error and stop transferring because of validation failure. Confirmed with Leon, we don’t need to transfer de-authorized message to ACM, so we can ignore this kind of error.

## Terms

Following table descript ACM terms and corresponding LABTrak terms:

|  |  |
| --- | --- |
| ACM | LABTrak |
| Visit | Hospital Episode No. |
| Subject ID | Hospital MRN |
| Test Number | Superset code |



# Mismatch analysis

ACM asks to send them test result only when all items ready. LABTrak will send out ORU\_O01 messages only when all necessary test items are filled and user click “Authorise”, after that, the result is not append-able, but can be corrected. So, Ensemble doesn’t need to do message filter here.

There are special requirements on HL7 field value from ACM, as following:

## PID segment

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Field Description | Gap | analysis |
| 18.1 | Patient Account Number. ID | Need to place here: Hospital MRN | In source, PID 4.1.1 is “Hospital MRN”. Copy from PID 4.1.1 |

## ORC segment

LABTrak does not output ORC segment in HL7, so Ensemble need to generate the whole ORC segment.

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Field Description | Gap | analysis |
| 1 | Order control | Need to place here: “RE” | Hardcode “RE” |
| 2.1 | Placer Order Number.entity identifier | Need to place here: Hospital Episode No. | In source, PV1 19.1 is “Hospital Episode No.”. Copy from PV1 19.1 |
| 3 | Filler Order Number |  | Copy from OBR 3 |

## OBR segment

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Field Description | Gap | analysis |
| 2.1 | Placer Order Number.entity identifier | Need to place here: Hospital Episode No. | In source, PV1 19.1 is “Hospital Episode No.”. Copy from PV1 19.1 |
| 4.4 | Universal Service Identifier.alternate identifier | Need to place here: superset code | There are only LABTrak “lab episode no.” and “testset code” in source HL7 (OBR 3.1). so, need to get superset code from LABTrak database using “lab episode no” & “testset code”, and append it to target HL7 |
| 25 | Result Status | Need to be **N** or **NC** | It’s not standard, according ACM: N - final  NC - final(corrected result). When result changed, LABTrak will place change time in 22.1. This can be used to determine if the result is a corrected result. |

## OBX segment (this part is ACM requirements not in their initial requirement document.)

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Field Description | Gap | analysis |
| 8 | AbnormalFlags | TrakLab output Abnormal flag in --,-,+,++. ACM need LL,L,H,HH | Need a lookup table to do the translation. And set target value by search this lookup table.  Add lookup table “ACMAbnormalFlag”. |
| 6.1 | Units.identifier | When observation value type is number (OBX:ValueType=”NM”) and Units is something like 109/L , Traklab will output it in format x10$C(22)9$C(23) /L. ACM want the format is 10\*9/L. | Need loop all NM OBX segment and replace x10$C(22) with 10\*, and $C(23) with null for Units.identifier field. |
| 7 | References Range | For NM OBX, when both ref high and ref low is 0 (or 0.0), TrakLab outputs “Reference Range ” as “0-0”(or 0.0-0.0). ACM wants it to be 0 (or 0.0). | Need loop all NM OBX segment and replace 0-0 with 0 (or 0.0-0.0 with 0.0) for References Range field. |
| 5 | ObservationValue() | When observation value type is text (OBX:ValueType=”TX”), it’s possible this value contains \.br\ to represent Carriage return. ACM want to remove it. | Need loop all TX OBX segment and replace \.br\ with null for ObservationValue() field. |
| 7 | References Range | For TX OBX, TrakLab does not record “Reference Range ” (in TrakLab, it marks it as “label”). But ACM wants some text type test item to have reference range, such as “Negative” for Pregnancy(code is B4141), | Need a lookup table to do the translation.  Add lookup table “ACMHCReferenceRange”, currently, there is only one item “B4141” in it. |
| 2 | Value Type | TrakLab outputs text type observation value in “TX” OBX, while ACM want “ST” OBX. | Need loop all TX OBX segment and replace “Value Type” to “ST”. |
| 8 | AbnormalFlags | For text type observation, TrakLab will not output “Abnormal flag”, even if it’s set in TrakLab.  And forth more, ACM asked Pregnancy(code is B4141) with value = “[POSITIVE](http://192.168.0.95:20023/csp/inte_acm/HL7/HL7SchemaSegmentStructure.csp?SS=SS:2.3:OBX#5)” output AbnormalFlags as “H”. | Need loop all TX OBX segment and get the abnormal flag back from TrakLab by web service.  Hardcode Pregnancy(code is B4141) with value = “[POSITIVE](http://192.168.0.95:20023/csp/inte_acm/HL7/HL7SchemaSegmentStructure.csp?SS=SS:2.3:OBX#5)” output AbnormalFlags as “H”. |
|  | Whole OBX segment | Traklab will output a OBX segment with value type =”FT”. It contains lab report in ObservationValue field with exact report format. ACM want to remove it. | Need loop all FT OBX segment and remove them.  In fact, there should have one FT OBX segment at most in every Traklab HL7 ORU\_R01 message. |

# Design

## LABTrak web service

We need to get superset code from LABTrak database by test set code and lab episode number. To keep loosely coupling between interface and LABTrak, design web service on LABTrak database:

/// TRAKLAB.Interface  
Class TRAKLAB.Interface Extends %SOAP.WebService [ ProcedureBlock ]  
{  
  
/// Name of the WebService.  
Parameter SERVICENAME = "Interface";  
  
/// TODO: change this to actual SOAP namespace.  
/// SOAP Namespace for the WebService  
Parameter NAMESPACE = "http://intersystems.com";  
  
Method GetSupSymByTestset(  
 pLabEpisodeNo As %String,  
 pTestCode As %String) As %String [ WebMethod ]  
{  
 Set tRet = ""  
 &sql(select VISTS\_SuperSet\_DR->CTSS\_Synonym into :tRet from SQLUser.EP\_VisitTestSet where VISTS\_ParRef=:pLabEpisodeNo and VISTS\_TestSet\_DR=:pTestCode)  
   
 Quit tRet  
}  
  
/// Check if the item is reportable. notice: if return value is "N", then this item is not reportable  
Method GetItemReportableByCode(  
 pTestSetCode As %String,  
 pTestItemCode As %String) As %String [ WebMethod ]  
{  
 Set tRet = ""  
 //&sql(select VISTD\_SupressReport into :tRet from SQLUser.EP\_VisitTestSetData where VISTD\_TestCode\_DR=:pTestItemCode and VISTD\_ParRef->VISTS\_ParRef=:pLabEpisodeNo)  
 &sql(select isnull(CTTSF\_Reportable,'') into :tRet from SQLUser.CT\_TestSetFields where CTTSF\_ParRef=:pTestSetCode and CTTSF\_DataItem\_DR=:pTestItemCode)  
 Quit tRet  
}  
  
/// Get item reference range  
Method GetItemReferenceRange(  
 pTestItemCode As %String,  
 pDOB As %String,  
 pGender As %String,  
 pLocation As %String = "ACM") As %String [ WebMethod ]  
{  
 Set tRet = ""   
 Set tRS=##class(%ScrollableResultSet).%New("%DynamicQuery:SQL")  
 Set sc=tRS.Prepare("SELECT CTTCR\_Age,CTTCR\_Species\_DR,CTTCR\_LowRange,CTTCR\_HighRange FROM SQLUser.CT\_TestCodeRanges WHERE CTTCR\_ParRef=? AND CTTCR\_PatientLocation\_DR=? ")  
 Do tRS.Execute(pTestItemCode,pLocation)  
 If (tRS.Count() = 0)  
 {  
 Set tRet = ""  
 }   
 ElseIf (tRS.Count() = 1)  
 {  
 D tRS.Next()  
 Set tLowRange = tRS.Data("CTTCR\_LowRange"),tHighRange = tRS.Data("CTTCR\_HighRange")  
 Set tRet = ..GetRange(tLowRange,tHighRange)  
 }  
 Else  
 {  
 Set tPatientCOSDays = $H-$ZDH(pDOB,5), tPatientCOSDays = +((tPatientCOSDays\365)+((tPatientCOSDays#365)/1000))  
 While tRS.Next()  
 {  
 Set tGender = tRS.Data("CTTCR\_Species\_DR"), tAge = tRS.Data("CTTCR\_Age"), tLowRange = tRS.Data("CTTCR\_LowRange"), tHighRange = tRS.Data("CTTCR\_HighRange")  
 Set tAgeStart = +$ZStrip($P(tAge,"-"), "<>W"), tAgeEnd = +$ZStrip($P(tAge,"-",2), "<>W")  
 if ((tPatientCOSDays>=tAgeStart)&&(tPatientCOSDays<=tAgeEnd)&&((tGender=pGender)||(tGender="")))  
 {  
 Set tRet = ..GetRange(tLowRange,tHighRange)  
 Quit   
 }  
 }  
 }  
  
Quit tRet  
}  
  
ClassMethod GetRange(  
 pLowRange As %String,  
 pHighRange As %String) As %String  
{  
 Set tRet=""  
 If ((pLowRange="")&&(pHighRange=""))  
 {  
 Set tRet = ""  
 }  
 Elseif ((pLowRange'="")&&(pHighRange=""))  
 {  
 Set tRet = ">"\_pLowRange  
 }  
 Elseif ((pLowRange="")&&(pHighRange'=""))  
 {  
 Set tRet = "<"\_pHighRange  
 }  
 Else  
 {  
 Set tRet = pLowRange\_"-"\_pHighRange  
 }  
 Quit tRet  
}  
  
/// get abnormal flag of text type test item by its code and test value.  
Method GetAbnormalFlagByCodeValue(  
 pTestItemCode As %String,  
 pTextValue As %String) As %String [ WebMethod ]  
{  
 Set tTestItemCode = pTestItemCode, tValue=pTextValue ,tRet=""  
 Set tId=""  
 Set tId = $O(^TTAB("TC",tTestItemCode,2,tId))  
 while (tId '="")  
 {  
 Set tRec = $G(^TTAB("TC",tTestItemCode,2,tId,"T",1))  
 if tRec = tValue  
 {  
 Set tRet=$P($G(^TTAB("TC",tTestItemCode,2,tId)), "\")  
 Quit  
 }  
 Set tId = $O(^TTAB("TC",tTestItemCode,2,tId))  
 }  
 Quit tRet  
}

}

## Interface (Ensemble) Side

### Web service client

Generate web service client using SOAP client wizard. No need to change any setting at import stage.

### Data transformation

Following transformation is needed:

|  |  |
| --- | --- |
| Segment/Field | Transformation |
| PID 18.1 | <assign property='target.{PIDgrpgrp(1).PIDgrp.PID:PatientAccountNumber.ID}' value='source.{PIDgrpgrp(1).PIDgrp.PID:AlternatePatientID.ID}' action='set'/> |
| ORC 1 | <assign property='target.{PIDgrpgrp(1).ORCgrp(1).ORC:OrderControl}' value='"RE"' action='set'/> |
| ORC 2.1 | <assign property='target.{PIDgrpgrp(1).ORCgrp(1).ORC:PlacerOrderNumber(1).entityidentifier}' value='source.{PIDgrpgrp(1).PIDgrp.PV1grp.PV1:VisitNumber.ID}' action='set'/> |
| ORC 3 | <assign property='target.{PIDgrpgrp(1).ORCgrp(1).ORC:FillerOrderNumber}' value='source.{PIDgrpgrp(1).ORCgrp(1).OBR:FillerOrderNumber}' action='set'/> |
| ORC 7 | <assign property='target.{PIDgrpgrp(1).ORCgrp(1).ORC:QuantityTiming}' value='source.{PIDgrpgrp(1).ORCgrp(1).OBR:QuantityTiming}' action='set'/> |
| OBR 2.1 | <assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBR:PlacerOrderNumber(1).entityidentifier}' value='source.{PIDgrpgrp(1).PIDgrp.PV1grp.PV1:VisitNumber.ID}' action='set'/> |
| OBR 25 | <if condition='source.{PIDgrpgrp(1).ORCgrp(1).OBR:ResultsRptStatusChngDateTime.timeofanevent}=""'> <true> <assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBR:ResultStatus}' value='"N"' action='set'/> </true> <false> <assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBR:ResultStatus}' value='"NC"' action='set'/> </false> </if> |
| OBR 4.4 | <assign property='setcode' value='source.{PIDgrpgrp(1).ORCgrp(1).OBR:FillerOrderNumber.entityidentifier}' action='set'/> <code> <![CDATA[            try         {          set tLabTrakWSUsr=##class(Ens.Config.Credentials).GetValue("LabTrakWS","Username")          set tLabTrakWSPwd=##class(Ens.Config.Credentials).GetValue("LabTrakWS","Password")          throw:(tLabTrakWSUsr["<N/A") ##class(%Exception.SystemException).%New("Credential Error","10001","Credential: LabTrakWS is not exist","")  set labepisodeno=$P(setcode," ",1), setcode=$P(setcode," ",2)  set soapClient = ##class(Interface.InterfaceSoap).%New()  d soapClient.WSSecurityLogin(tLabTrakWSUsr,tLabTrakWSPwd)  set supersetcode=soapClient.GetSupSymByTestset(labepisodeno,setcode)  set soapClient=""         }         catch err         {          $$$LOGERROR("Error on getting superset code with "\_err.Code\_", "\_err.Name)         } ]]> </code> <assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBR:UniversalServiceIdentifier.alternateidentifier}' value='supersetcode' action='set'/> |
| OBX | <foreach property='source.{PIDgrpgrp(1).ORCgrp(1).OBXgrp()}' key='kOBX'> <assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:AbnormalFlags()}' value='..Lookup("ACMAbnormalFlag",source.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:AbnormalFlags()})' action='set'/> <assign property='tValueType' value='source.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:ValueType}' action='set'/> <if condition='tValueType="FT"'> <true> <assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX}' value='' action='remove'/> </true> </if> <if condition='tValueType="TX"'> <true> <assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:ObservationValue()}' value='..ReplaceStr(source.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:ObservationValue()},"\.br\","")' action='set'/> </true> </if>  <if condition='tValueType="NM"'> <true> <!-- <if condition='..Matches(source.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:Units.identifier},@tPattern)'>  <true>  -->  <assign property='^QPLog(kOBX)' value='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:Units.identifier}' action='set'/>  <assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:Units.identifier}' value='..ReplaceStr(target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:Units.identifier},"x10"\_$C(22),"10\*")' action='set'/>  <assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:Units.identifier}' value='..ReplaceStr(target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:Units.identifier},$C(23),"")' action='set'/> </true> </if> </foreach> |

Summary:

Class LABINT.DT.TrakLab2ACM Extends Ens.DataTransformDTL  
{  
  
XData DTL [ XMLNamespace = "http://www.intersystems.com/dtl" ]  
{  
<?xml version="1.0" ?>  
<transform targetClass='EnsLib.HL7.Message' targetDocType='2.3:ORU\_R01' sourceClass='EnsLib.HL7.Message' sourceDocType='2.3:ORU\_R01' create='copy' language='objectscript'>  
<code>  
<![CDATA[   
 set supersetcode="", setcode="", labepisodeno="", tValueType="", tPattern=".E1""x""1.N1"""\_$C(22)\_"""1.N1"""\_$C(23)\_""".E"  
 set tItemCode="", tGender="", tDOB="", tRange="",tReportable="",tReferenceRange="", tAbnormalFlag="", tTextValue=""  
]]>  
</code>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).ORC:OrderControl}' value='"RE"' action='set'/>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).ORC:PlacerOrderNumber(1).entityidentifier}' value='source.{PIDgrpgrp(1).PIDgrp.PV1grp.PV1:VisitNumber.ID}' action='set'/>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBR:PlacerOrderNumber(1).entityidentifier}' value='source.{PIDgrpgrp(1).PIDgrp.PV1grp.PV1:VisitNumber.ID}' action='set'/>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).ORC:FillerOrderNumber}' value='source.{PIDgrpgrp(1).ORCgrp(1).OBR:FillerOrderNumber}' action='set'/>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).ORC:QuantityTiming}' value='source.{PIDgrpgrp(1).ORCgrp(1).OBR:QuantityTiming}' action='set'/>  
<if condition='source.{PIDgrpgrp(1).ORCgrp(1).OBR:ResultsRptStatusChngDateTime.timeofanevent}=""'>  
<true>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBR:ResultStatus}' value='"N"' action='set'/>  
</true>  
<false>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBR:ResultStatus}' value='"NC"' action='set'/>  
</false>  
</if>  
<assign property='target.{PIDgrpgrp(1).PIDgrp.PID:PatientAccountNumber.ID}' value='source.{PIDgrpgrp(1).PIDgrp.PID:AlternatePatientID.ID}' action='set'/>  
<assign property='setcode' value='source.{PIDgrpgrp(1).ORCgrp(1).OBR:FillerOrderNumber.entityidentifier}' action='set'/>  
<code>  
<![CDATA[   
        try  
        {  
         set tLabTrakWSUsr=##class(Ens.Config.Credentials).GetValue("LabTrakWS","Username")  
         set tLabTrakWSPwd=##class(Ens.Config.Credentials).GetValue("LabTrakWS","Password")  
         throw:(tLabTrakWSUsr["<N/A") ##class(%Exception.SystemException).%New("Credential Error","10001","Credential: LabTrakWS is not exist","")  
 set labepisodeno=$P(setcode," ",1), setcode=$P(setcode," ",2)  
 set soapClient = ##class(Interface.InterfaceSoap).%New()  
 d soapClient.WSSecurityLogin(tLabTrakWSUsr,tLabTrakWSPwd)  
 set supersetcode=soapClient.GetSupSymByTestset(labepisodeno,setcode)  
 set soapClient=""  
        }  
        catch err  
        {  
         $$$LOGERROR("Error on getting superset code with "\_err.Code\_", "\_err.Name)  
        }  
]]>  
</code>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBR:UniversalServiceIdentifier.alternateidentifier}' value='supersetcode' action='set'/>  
<foreach property='source.{PIDgrpgrp(1).ORCgrp(1).OBXgrp()}' key='kOBX'>  
<assign property='tItemCode' value='source.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:ObservationIdentifier.identifier}' action='set'/>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:AbnormalFlags()}' value='..Lookup("ACMAbnormalFlag",source.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:AbnormalFlags()})' action='set'/>  
<assign property='tValueType' value='source.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:ValueType}' action='set'/>  
<if condition='tValueType="FT"'>  
<true>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX}' value='' action='remove'/>  
</true>  
</if>  
<if condition='tValueType="TX"'>  
<true>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:ObservationValue()}' value='..ReplaceStr(source.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:ObservationValue()},"\.br\","")' action='set'/>  
<assign property='tTextValue' value='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:ObservationValue()}' action='set'/>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:ValueType}' value='"ST"' action='set'/>  
<code>  
<![CDATA[   
        try  
        {  
         set tReferenceRange = $G(^Ens.LookupTable("ACMHCReferenceRange",tItemCode))  
        }  
        catch err  
        {  
         $$$LOGERROR("Error on fill Reference Range with "\_err.Code\_", "\_err.Name)  
        }  
]]>  
</code>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:ReferencesRange}' value='tReferenceRange' action='set'/>  
<code>  
<![CDATA[   
        try  
        {  
         set tLabTrakWSUsr=##class(Ens.Config.Credentials).GetValue("LabTrakWS","Username")  
         set tLabTrakWSPwd=##class(Ens.Config.Credentials).GetValue("LabTrakWS","Password")  
         throw:(tLabTrakWSUsr["<N/A") ##class(%Exception.SystemException).%New("Credential Error","10001","Credential: LabTrakWS is not exist","")  
 set labepisodeno=$P(setcode," ",1), setcode=$P(setcode," ",2)  
 set soapClient = ##class(Interface.InterfaceSoap).%New()  
 d soapClient.WSSecurityLogin(tLabTrakWSUsr,tLabTrakWSPwd)  
 set tAbnormalFlag=soapClient.GetAbnormalFlagByCodeValue(tItemCode,tTextValue)  
 set soapClient=""

// hard coded: if the test item code is B4141 and tTextValue is POSITIVE, then abnormal flag will be H  
         set:((tItemCode="B4141")&&($ZCVT(tTextValue,"U")="POSITIVE")) tAbnormalFlag = "H"  
        }  
        catch err  
        {  
         $$$LOGERROR("Error on get abnormal flag for item:"\_tItemCode\_" with "\_err.Code\_", "\_err.Name)  
        }  
]]>  
</code>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:AbnormalFlags()}' value='tAbnormalFlag' action='set'/>  
</true>  
</if>  
<if condition='tValueType="NM"'>  
<true>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:Units.identifier}' value='..ReplaceStr(target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:Units.identifier},"x10"\_$C(22),"10\*")' action='set'/>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:Units.identifier}' value='..ReplaceStr(target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:Units.identifier},$C(23),"")' action='set'/>  
<if condition='source.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:ReferencesRange}="0-0"'>  
<true>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:ReferencesRange}' value='0' action='set'/>  
</true>  
</if>  
<if condition='source.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:ReferencesRange}="0.0-0.0"'>  
<true>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:ReferencesRange}' value='"0.0"' action='set'/>  
</true>  
</if>  
</true>  
</if>  
</foreach>  
</transform>

}

### Production

#### Routing rules

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Rules** | **Source** | **Message Class** | **Document Name** | **Schema Category** | **Schema DocType** | **Conditions** | **Action** | | **Transform** | **Target** |
|  | [Base](javascript:%20editRule('base');) | HL7FileService | EnsLib.HL7.Message |  |  |  |  |  |  |  |  |
|  | [Rule 1](javascript:%20editRule('0');) | *HL7FileService* | *EnsLib.HL7.Message* |  | 2.3 | ORU\_R01 | |  | | --- | | [Add](javascript:%20addCondition(0);) | | [X](javascript:%20removeAction(0,0);)  [Add](javascript:%20addAction(0);) | [Send](javascript:%20editAction(0,0);) | LABINT.DT.TrakLab2ACM | HL7FileOperation |
|  |  | | | | | | | | | | |

#### Credentials setting

|  |  |  |
| --- | --- | --- |
| Name | Description | Account Infomation |
| LABTRIALFTP | Account for UFH internal FTP | ufhftp/\*\*\*\* |
| LabTrakWS | Account for access web service of LabTrak | \_system/\*\*\*\* |

#### Adaptor setting

|  |  |  |
| --- | --- | --- |
| Adaptor | setting | value |
| File inbound adaptor | Message Schema Category | 2.3 |
| DefCharEncoding | !GB18030 |
| File Path | d:\LABInt\TrakLab\in |
| Archive Path | d:\LABInt\TrakLab\archive |
| Charset | GB18030 |
| FTP outbound adaptor | File Name | %f\_%Q.hl7 |
| FTP Server |  |
| FTP Port | 21 |
| Credentials | LABTRIALFTP |
| DefCharEncoding | !GB18030 |
| Charset | GB18030 |
| Stay Connected | 0 \*to avoid a bug of FTP adaptor |
| ~~File outbound adaptor~~ | ~~File Name~~ | ~~%f\_%Q.hl7~~ |
| ~~File Path~~ | ~~d:\LABInt\ACM\in~~ |
| ~~DefCharEncoding~~ | ~~!GB18030~~ |
| ~~Charset~~ | ~~GB18030~~ |
|  |  |  |

Backup: Khatha changed Traklab routine, so, reference range and non-reportable part are not need anymore, remove them.

<?xml version="1.0" ?>  
<transform targetClass='EnsLib.HL7.Message' targetDocType='2.3:ORU\_R01' sourceClass='EnsLib.HL7.Message' sourceDocType='2.3:ORU\_R01' create='copy' language='objectscript'>  
<code>  
<![CDATA[   
 set supersetcode="", setcode="", labepisodeno="", tValueType="", tPattern=".E1""x""1.N1"""\_$C(22)\_"""1.N1"""\_$C(23)\_""".E"  
 set tItemCode="", tGender="", tDOB="", tRange="",tReportable=""  
]]>  
</code>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).ORC:OrderControl}' value='"RE"' action='set'/>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).ORC:PlacerOrderNumber(1).entityidentifier}' value='source.{PIDgrpgrp(1).PIDgrp.PV1grp.PV1:VisitNumber.ID}' action='set'/>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBR:PlacerOrderNumber(1).entityidentifier}' value='source.{PIDgrpgrp(1).PIDgrp.PV1grp.PV1:VisitNumber.ID}' action='set'/>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).ORC:FillerOrderNumber}' value='source.{PIDgrpgrp(1).ORCgrp(1).OBR:FillerOrderNumber}' action='set'/>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).ORC:QuantityTiming}' value='source.{PIDgrpgrp(1).ORCgrp(1).OBR:QuantityTiming}' action='set'/>  
<if condition='source.{PIDgrpgrp(1).ORCgrp(1).OBR:ResultsRptStatusChngDateTime.timeofanevent}=""'>  
<true>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBR:ResultStatus}' value='"N"' action='set'/>  
</true>  
<false>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBR:ResultStatus}' value='"NC"' action='set'/>  
</false>  
</if>  
<assign property='target.{PIDgrpgrp(1).PIDgrp.PID:PatientAccountNumber.ID}' value='source.{PIDgrpgrp(1).PIDgrp.PID:AlternatePatientID.ID}' action='set'/>  
<assign property='setcode' value='source.{PIDgrpgrp(1).ORCgrp(1).OBR:FillerOrderNumber.entityidentifier}' action='set'/>  
<code>  
<![CDATA[   
        try  
        {  
         set tLabTrakWSUsr=##class(Ens.Config.Credentials).GetValue("LabTrakWS","Username")  
         set tLabTrakWSPwd=##class(Ens.Config.Credentials).GetValue("LabTrakWS","Password")  
         throw:(tLabTrakWSUsr["<N/A") ##class(%Exception.SystemException).%New("Credential Error","10001","Credential: LabTrakWS is not exist","")  
 set labepisodeno=$P(setcode," ",1), setcode=$P(setcode," ",2)  
 set soapClient = ##class(Interface.InterfaceSoap).%New()  
 d soapClient.WSSecurityLogin(tLabTrakWSUsr,tLabTrakWSPwd)  
 set supersetcode=soapClient.GetSupSymByTestset(labepisodeno,setcode)  
 set soapClient=""  
        }  
        catch err  
        {  
         $$$LOGERROR("Error on getting superset code with "\_err.Code\_", "\_err.Name)  
        }  
]]>  
</code>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBR:UniversalServiceIdentifier.alternateidentifier}' value='supersetcode' action='set'/>  
<foreach property='source.{PIDgrpgrp(1).ORCgrp(1).OBXgrp()}' key='kOBX'>  
<assign property='tItemCode' value='source.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:ObservationIdentifier.identifier}' action='set'/>  
<!--<code>  
<![CDATA[   
        set tRange="",tReportable=""  
]]>  
</code>  
<code>  
<![CDATA[   
        try  
        {  
         set tLabTrakWSUsr=##class(Ens.Config.Credentials).GetValue("LabTrakWS","Username")  
         set tLabTrakWSPwd=##class(Ens.Config.Credentials).GetValue("LabTrakWS","Password")  
         throw:(tLabTrakWSUsr["<N/A") ##class(%Exception.SystemException).%New("Credential Error","10001","Credential: LabTrakWS is not exist","")  
 set soapClient = ##class(Interface.InterfaceSoap).%New()  
 d soapClient.WSSecurityLogin(tLabTrakWSUsr,tLabTrakWSPwd)  
 set tReportable=soapClient.GetItemReportableByCode(setcode,tItemCode)  
 set soapClient=""  
        }  
        catch err  
        {  
         $$$LOGERROR("Error on getting isReportable with "\_err.Code\_", "\_err.Name)  
        }  
]]>  
</code>  
<if condition='tReportable="N"'>  
<true>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX}' value='' action='remove'/>  
</true>  
<false>-->  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:AbnormalFlags()}' value='..Lookup("ACMAbnormalFlag",source.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:AbnormalFlags()})' action='set'/>  
<assign property='tValueType' value='source.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:ValueType}' action='set'/>  
<if condition='tValueType="FT"'>  
<true>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX}' value='' action='remove'/>  
</true>  
</if>  
<if condition='tValueType="TX"'>  
<true>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:ObservationValue()}' value='..ReplaceStr(source.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:ObservationValue()},"\.br\","")' action='set'/>  
</true>  
</if>  
<if condition='tValueType="NM"'>  
<true>  
<!--  
<assign property='tGender' value='source.{PIDgrpgrp(1).PIDgrp.PID:Sex}' action='set'/>  
<assign property='tDOB' value='source.{PIDgrpgrp(1).PIDgrp.PID:DateofBirth.timeofanevent}' action='set'/>  
<code>  
<![CDATA[   
         try  
         {  
         set tLabTrakWSUsr=##class(Ens.Config.Credentials).GetValue("LabTrakWS","Username")  
         set tLabTrakWSPwd=##class(Ens.Config.Credentials).GetValue("LabTrakWS","Password")  
         throw:(tLabTrakWSUsr["<N/A") ##class(%Exception.SystemException).%New("Credential Error","10001","Credential: LabTrakWS is not exist","")  
 set soapClient = ##class(Interface.InterfaceSoap).%New()  
 d soapClient.WSSecurityLogin(tLabTrakWSUsr,tLabTrakWSPwd)  
 set tRange=soapClient.GetItemReferenceRange(tItemCode,tDOB,tGender,"ACM")  
 set soapClient=""  
         }  
         catch err  
         {  
         $$$LOGERROR("Error on getting reference range with "\_err.Code\_", "\_err.Name)  
         }  
]]>  
</code>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:ReferencesRange}' value='tRange' action='set'/>-->  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:Units.identifier}' value='..ReplaceStr(target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:Units.identifier},"x10"\_$C(22),"10\*")' action='set'/>  
<assign property='target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:Units.identifier}' value='..ReplaceStr(target.{PIDgrpgrp(1).ORCgrp(1).OBXgrp(kOBX).OBX:Units.identifier},$C(23),"")' action='set'/>  
</true>  
</if>  
<!--</false>  
</if>-->  
</foreach>  
</transform>