DXC Assure Claims

Managing Workflow with Web Based Script Editor

DXC Technology, 1775 Tysons Blvd, Tysons, VA 22102, USA. All rights reserved. Printed in U.S.A.

All questions regarding this documentation should be routed through customer assistance, Blythewood, SC, on Phone: 800-420-6007 or via Email: risksupp@dxc.com

Table of Contents

Managing Workflow with Web Based Script Editor

Introduction

What DXC Assure Claims Scripting Offers

DXC Assure Claims Scripting offers the following:

1. Custom rules can be added to organization workflow in the claim administration management process.
2. After Save Routine Example:
    - Send email notifications to users/groups about a claim.
    - Display an on-screen reminder (warning message) to review a document/file or perform actions
3. Assure Claims custom scripts are all in VB .Net.

Scripting Routines

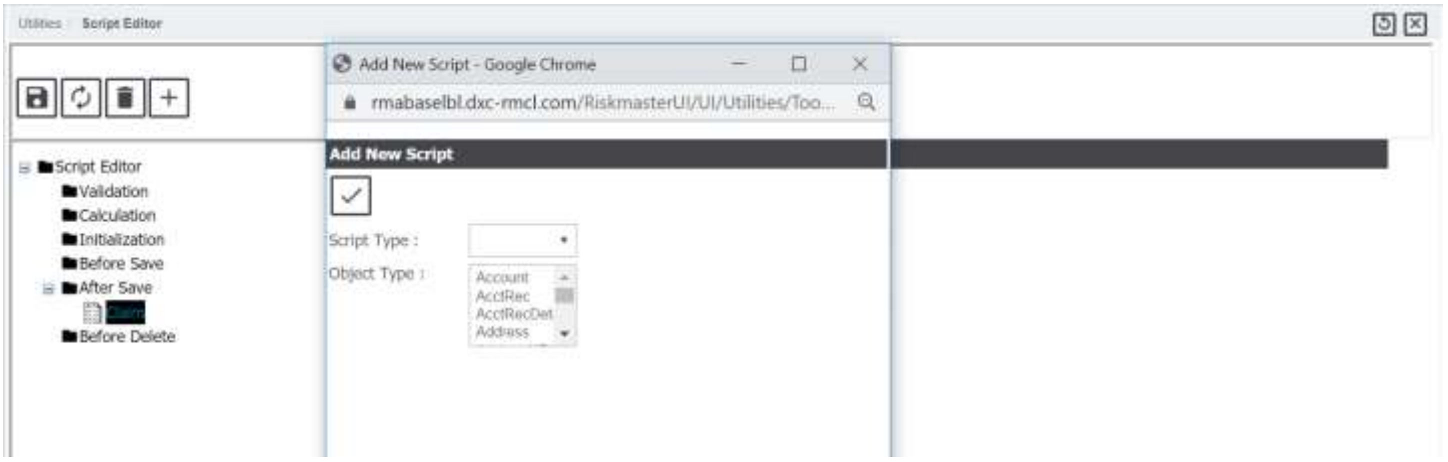
The object lifecycle in Assure Claims consists of the following processes:

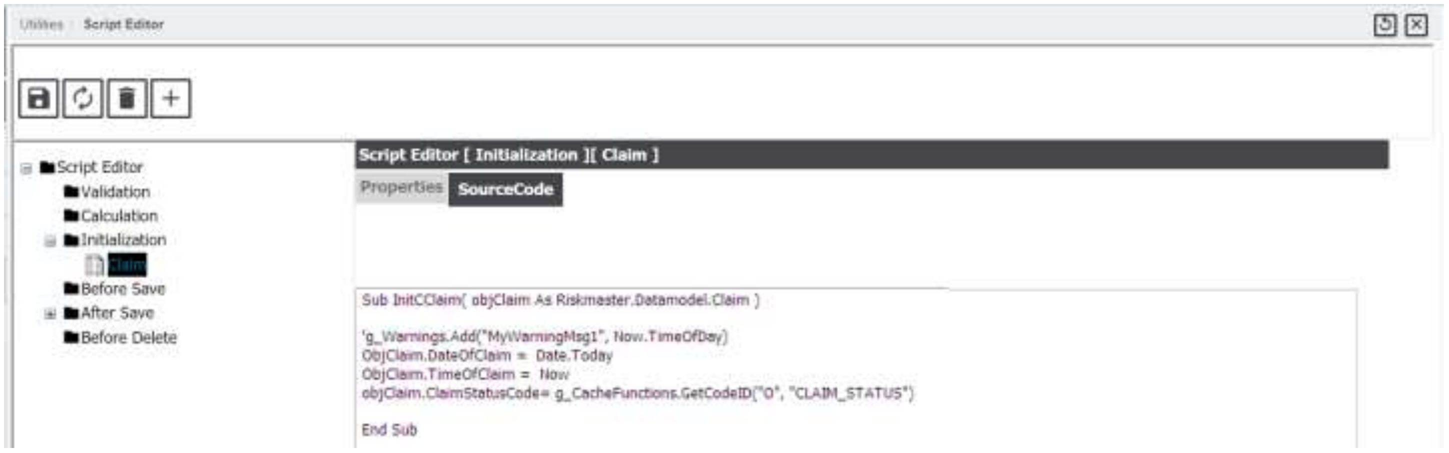
1. **Initialization**
    - **Triggers** on requests for new forms.
    - **Initializes** any custom default values.
      i.e., default claim date to today's date.
2. **Calculation**
    - **Triggers** on Save requests.
    - **Calculates** current object values.
      i.e. supplemental fields are populated with data calculated from one or more fields.
3. **Validation**
    - **Triggers** after Calculation routine is done.
    - **Process** your custom business rule validations.
    - **Cancels** the Save process, if validation rules are not met.
      i.e., if specific claim type, then it requires reserves to be set.
4. **Before Save**
    - **Triggers** after successful Validation.
    - **Processes** data and make necessary changes related to the current object.
    - **All changes** will be committed by Assure Claims after this routine.
5. **After Save**
    - **Triggers** after all changes are successfully saved by Assure Claims.
    - **Processes** workflow management (emails, warning reminders, etc.)
6. **Before Delete**
    - **Triggers** on Delete requests.
    - **Processes** your workflow management and custom rules.
    - **Cancels** the Delete process if custom policies are not met.

Important note about After Save scripting routine

- Fired after all changes are successfully saved by Assure Claims.
- Main purpose is to send emails and display informative messages and/or reminders!
- New Data/existing data must be created/updated in prior scripting routines (e.g. Before Save) NOT in After Save scripts.
- New data may be created in After Save scripts but only under certain circumstances. ClaimID, Control Number, etc. may not be available in prior routines for a new save.
  - For example, if Client wants to create some sort of collections with the same control number when a new payment is made.
  - The payment transaction is not rolled back, if something goes wrong while creating the collections in an After-Save script.
  - Try and Catch block must be used in After Save scripts so in case something goes wrong at least we can throw a Warning message to inform User as the payment was made but collections did not get created!

Script Editor User Interface





```
Sub InitCClaim( objClaim As Riskmaster.Datamodel.Claim )

'g_Warnings.Add("MyWarningMsg1", Now.TimeOfDay)
ObjClaim.DateOfClaim =  Date.Today
ObjClaim.TimeOfClaim =  Now
objClaim.ClaimStatusCode= g_CacheFunctions.GetCodeID("O", "CLAIM_STATUS")

End Sub
```

Technology Overview

DXC Assure Claims Solution vs. Legacy Solution

DXC Assure Claims Scripting

What happens to legacy scripts?	Unless someone converts them, nothing
Can products run in parallel if script technology is "incompatible"?	Yes. New and old scripts are stored in separate DB tables. They can live in the same database and are each invoked from only the appropriate product(s).
Where do most Assure Claims scripting operations occur in Assure Claims?	DataModel is the object representation of the database through which most Assure Claims scripting operations should occur.

Compiled Version of Assure Claims Scripts Already Available at Run Time

- Once a new script is saved or an existing one is edited, a compiled version (DLL file) is stored in database.
- Compiling errors are received at the time scripts are created, saved or edited.
- Each time the script gets fired, there is no need to compile the script; hence better performance.

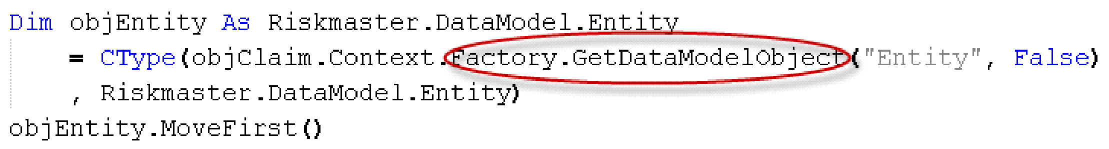
Important Notes about DXC Assure Claims Scripting Module

- The script engine's behavior is screen based.
- Claim validation script does not get fired while an event is being saved or vice versa.
- A separate claim validation script for claim screen and a separate event validation script for event screen can be created with no conflicts.

Creating a DataModel Object

Create a DataModel object

- Get to the Factory property by looking at the current object (let say it is objClaim):



```
Dim objEntity As Riskmaster.DataModel.Entity
    = CType(objClaim.Context.Factory.GetDataModelObject("Entity", False)
    , Riskmaster.DataModel.Entity)
objEntity.MoveFirst()
```

- After creating and initializing our DataModel object, we can move to an existing record (like above) or we can populate properties and save the object as a new record.

Navigation

Walk the hierarchy

- Up to Parents
  - A parent must be loaded first (no need to use .Save() method if we update a parent and we are in Cal/Val/BeforeSave routines):

```
objClaim.LoadParent()
Dim objEvent As Riskmaster.DataModel.Event = CType(objClaim.Parent
    , Riskmaster.DataModel.Event)
g_Warnings.Add("MyWarningMsg1"
    , "Event Status: " & g_CacheFunctions.GetCodeDesc(objEvent.EventStatusCode))
```

- Down to children
  - Let say we are in one of Claim Cal/Val/BeforeSave routines.
  - Loop through a list of children:

```
For Each objClaimant As Riskmaster.DataModel.Claimant In objClaim.ClaimantList
    If (objClaimant.ClaimantEid = 200) Then
        objClaimant.PrimaryClmntFlag = True
        Exit For
    End If
Next
```

Although we updated objClaimant, we don't need to call .Save() method as it is a claim child and RMX automatically saves it after the BeforeSave routine is successfully done.

Fetch and update a Record

- Let say we are in one of Claim Cal/Val/BeforeSave routines:

```
Dim objEntity As Riskmaster.DataModel.Entity = Nothing
If (Not objClaim.PrimaryClaimant Is Nothing
    And objClaim.PrimaryClaimant.AttorneyEid <> 0) Then
    objEntity = CType(objClaim.Context.Factory.GetDataModelObject("Entity", False)
        , Riskmaster.DataModel.Entity)
    objEntity.MoveTo(objClaim.PrimaryClaimant.AttorneyEid)
    objEntity.City = "Los Angeles"
End If
If (objEntity Is Nothing) Then
    g_ValErrors.Add("MyErrorMsg1", "There is no primary claimant attorney!")
Else
    objEntity.Save()
End If
```

We need to save objEntity since we updated it and it is not a claim child.

Access to Supplemental fields

- Let say we are in Claim Calculation routine:

```
objClaim.LoadParent()
Dim objEvent As Riskmaster.DataModel.Event = CType(objClaim.Parent
    , Riskmaster.DataModel.Event)
If (objClaim.IsNew) Then
    If (objEvent.DateReported < objClaim.DateOfClaim) Then
        objClaim.Supplementals("TEST_TEXT").Value = "0"
    Else
        objClaim.Supplementals("TEST_TEXT").Value = "1"
    End If
End If
```

All letters in upper-case!

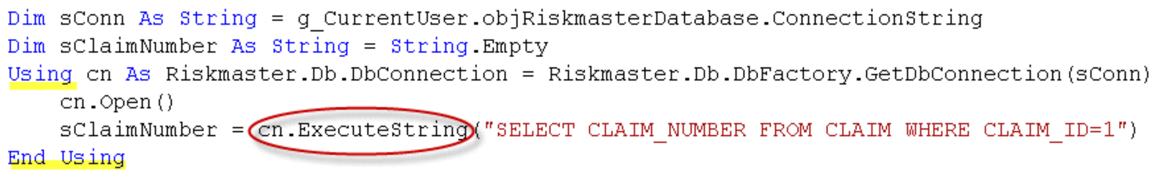
- Assure Claims saves this Supplemental field after the BeforeSave routine is successfully done.

Direct Access to Database

(NOT recommended)

Direct Access to Database

- Always stay away from db calls which can be avoided by using DataModel.
- Reading a single Data Value

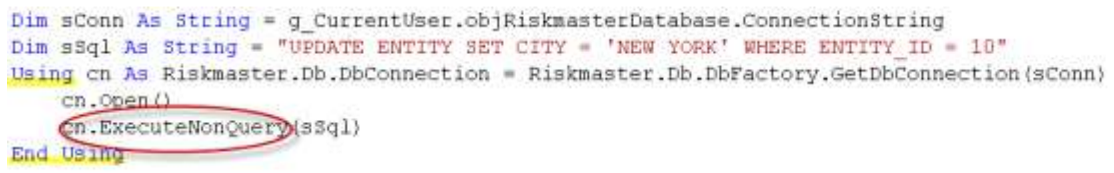


```
Dim sConn As String = g_CurrentUser.objRiskmasterDatabase.ConnectionString
Dim sClaimNumber As String = String.Empty
Using cn As Riskmaster.Db.DbConnection = Riskmaster.Db.DbFactory.GetDbConnection(sConn)
    cn.Open()
    sClaimNumber = cn.ExecuteString("SELECT CLAIM_NUMBER FROM CLAIM WHERE CLAIM_ID=1")
End Using
```

- Reading a record set:

Direct Access to Database

- The DataModel .Save() methods can prevent damaging data, system parameters and history information.
- Creating/Updating data through db calls are not recommended at all.
- Non-query statements:



```
Dim sConn As String = g_CurrentUser.objRiskmasterDatabase.ConnectionString
Dim sSql As String = "UPDATE ENTITY SET CITY = 'NEW YORK' WHERE ENTITY_ID = 10"
Using cn As Riskmaster.Db.DbConnection = Riskmaster.Db.DbFactory.GetDbConnection(sConn)
    cn.Open()
    cn.ExecuteNonQuery(sSql)
End Using
```

Safety, Performance Considerations and Best Practices

Data Model Vs Direct Access

- Use DataModel instead of direct access to database:
- Data, System Parameters and History Information can be damaged!

Data Model Roll Back

- If something goes wrong, DataModel will roll back all changes in db.
- This does not happen in case of direct access to database

No Additional Changes

- It is NOT recommended to make further changes in AfterSave routines.
- Changes in prior routines will not be rolled back if something goes wrong.
- Also, AfterSave routines do not throw custom error messages; they can only throw warnings.

Using Blocks

- In case of having direct access to database, always use the USING blocks as shown before.

Loops and Performance

- Loops can affect the performance! Loops inside another loop can dramatically change time complexity.

Loop Termination

- Loop termination conditions must be always satisfied; otherwise we face endless loops in special cases.

.Save( )

- No needs to call .Save() method for the current, parent and children objects in Cal/Val/BeforeSave routines.
- Can cause problems and performance issues. Assure Claims calls .Save() methods and all changes will be saved.

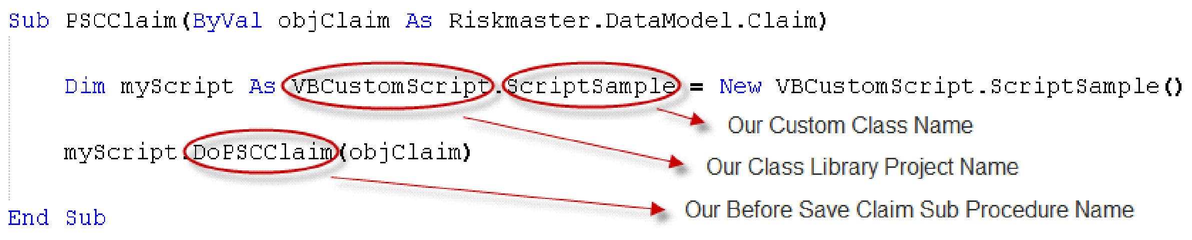
Debugging

Logging

- Logging messages is the primary method for debugging during script development.
- The default log file is serviceerror.log
    - For example: DXC Assure Claims.Common.Log.Write("Hello Scripting World")
- Categorized messages can be configured in “loggingConfiguration.config” file under WCFService folder so that they can be written into separate files at run-time.
    - For example: DXC Assure Claims.Common.Log.Write(“My Categorized Message Here”,”Scripting”)

Advanced Debugging using Assure Claims assemblies

- This enables us to step through our scripts for line by line debugging.
- Visual Studio with the "Attach to Process" feature (Standard Edition or higher) need to be installed on a test server.
- We compile our class library (Referencing to some Assure Claims assemblies are required) and add our dll filename in “ScriptingAssemblies” area in Web.config under WCFService folder.
- In Script Editor, our script simply becomes a call out to the custom assembly. For example:



```
Sub PSCClaim(ByVal objClaim As Riskmaster.DataModel.Claim)

    Dim myScript As VBCustomScript.ScriptSample = New VBCustomScript.ScriptSample()

    myScript.DoPSCClaim(objClaim)

End Sub
```

Our Custom Class Name
Our Class Library Project Name
Our Before Save Claim Sub Procedure Name

Scripts

Claim Adjuster Before Delete script to display a warning message to the end user to add a Current Adjuster to the Claim if not already present.

```
Sub PDCClaimAdjuster( objClaimAdjuster As Riskmaster.Datamodel.ClaimAdjuster )

Dim iCount As Integer = 0
```

```
        Try


            iCount = objClaimAdjuster.Context.DbConnLookup.ExecuteInt(String.Format("SELECT COUNT(*) FROM 
CLAIM_ADJUSTER WHERE CLAIM_ID = {0} AND CURRENT_ADJ_FLAG = -1 AND ADJ_ROW_ID <> {1}", 
objClaimAdjuster.ClaimId, objClaimAdjuster.AdjRowId))

            If (iCount < 1) Then

                g_Warnings.Add("AdjusterWarning", "Please remember to add an adjuster record to the claim with the 
current adjuster checkbox marked.")

            End If


        Catch ex As Exception

            Log.Write(ex.ToString)

        End Try

End Sub
```

Claimant After Save script to add Claimant’s Attorney, Insurer and Attorney Firm as Person Involved of type Others at Event level.

```
Sub ASCClaimant(objClaimant As Riskmaster.DataModel.Claimant)


        Dim objClaimantMMSEA As ClaimantMmsea

        Dim objClmtMmseaTpoc As ClmtMmseaTpoc

        Dim sDateTimeNow As String

        Dim iCount As Integer = 0

        Dim bUpdateRequired As Boolean

        bUpdateRequired = False

        Try

            sDateTimeNow = Riskmaster.Common.Conversion.ToDbDate(Date.Now)

            Log.Write("1")



            'start - Add Claimant attorney ,insurer and attorney firm as person involved
            Dim objPersonInvolved As PersonInvolved
            Dim iEventId As Integer = 0
            Dim iPiRowId As Integer = 0

            'FOR ADDING Claimant AS PERSON INVOLVED
            iEventId = objClaimant.Context.DbConnLookup.ExecuteInt(String.Format("SELECT EVENT_ID FROM
CLAIM WHERE CLAIM_ID={0}", objClaimant.ClaimId))



            'FOR ADDING Claimant ATTORNEY AS PERSON INVOLVED
            Log.Write("Saving Claimant attorney start")
            If objClaimant.AttorneyEid > 0 Then
                objPersonInvolved = CType(objClaimant.Context.Factory.GetDataModelObject("PersonInvolved", False),
PersonInvolved)
                iPiRowId = CType(g_Storage("ClaimantAtt_PiRowId_" & objClaimant.ClaimId), Integer)

                If iPiRowId > 0 Then
                    Log.Write("Saving Claimant attorney update")
                    objPersonInvolved.MoveTo(iPiRowId)
                End If
                objPersonInvolved.PiEid = objClaimant.AttorneyEid
                objPersonInvolved.PiTypeCode = objClaimant.Context.LocalCache.GetCodeId("O",
"PERSON_INV_TYPE")
                objPersonInvolved.EventId = iEventId
                objPersonInvolved.ParentTableName = "EVENT"
                objPersonInvolved.ParentRowId = iEventId
                objPersonInvolved.Save()
                objPersonInvolved = Nothing
            End If
            g_Storage.Delete("ClaimantAtt_PiRowId_" & objClaimant.ClaimId)
            'FOR ADDING Claimant INSURER AS PERSON INVOLVED
            Log.Write("Saving Claimant ins start")
            If objClaimant.InsurerEid > 0 Then
                objPersonInvolved = CType(objClaimant.Context.Factory.GetDataModelObject("PersonInvolved", False),
PersonInvolved)
```

```
            iPiRowId = CType(g_Storage("ClaimantIns_PiRowId_" & objClaimant.ClaimId), Integer)
            If iPiRowId > 0 Then
                Log.Write("Saving Claimant ins update")
                objPersonInvolved.MoveTo(iPiRowId)
            End If
            objPersonInvolved.PiEid = objClaimant.InsurerEid
            objPersonInvolved.PiTypeCode = objClaimant.Context.LocalCache.GetCodeId("O",
"PERSON_INV_TYPE")
            objPersonInvolved.EventId = iEventId
            objPersonInvolved.ParentTableName = "EVENT"
            objPersonInvolved.ParentRowId = iEventId
            objPersonInvolved.Save()
            objPersonInvolved = Nothing
        End If
        g_Storage.Delete("ClaimantIns_PiRowId_" & objClaimant.ClaimId)


        'FOR ADDING Claimant ATTORNEY FIRM AS PERSON INVOLVED
        Log.Write("Saving Claimant firm start")
        If objClaimant.AttorneyEntity.ParentEid > 0 Then
            objPersonInvolved = CType(objClaimant.Context.Factory.GetDataModelObject("PersonInvolved", False),
PersonInvolved)
            iPiRowId = CType(g_Storage("ClaimantAttFirm_PiRowId_" & objClaimant.ClaimId), Integer)
            If iPiRowId > 0 Then
                Log.Write("Saving Claimant firm update")
                objPersonInvolved.MoveTo(iPiRowId)
            End If
            objPersonInvolved.PiEid = objClaimant.AttorneyEntity.ParentEid
            objPersonInvolved.PiTypeCode = objClaimant.Context.LocalCache.GetCodeId("O",
"PERSON_INV_TYPE")
            objPersonInvolved.EventId = iEventId
            objPersonInvolved.ParentTableName = "EVENT"
            objPersonInvolved.ParentRowId = iEventId
            objPersonInvolved.Save()
            objPersonInvolved = Nothing
        End If
        g_Storage.Delete("ClaimantAttFirm_PiRowId_" & objClaimant.ClaimId)
        'end - Add Claimant attorney ,insurer and attorney firm as person involved


    Catch ex As Exception
        '  g_ValErrors.Add("ASCClaimant", ex.Message.ToString)
        Log.Write(ex.Message.ToString)
    Finally


        If Not objClaimant Is Nothing Then
            objClaimant = Nothing
        End If
    End Try
End Sub
```

Claim Adjuster After Save script to display a warning message to end user to add Current Adjuster to the Claim if not already present.

```
Sub ASCClaimAdjuster(objClaimAdjuster As Riskmaster.DataModel.ClaimAdjuster)


        Dim iCount As Integer = 0
        Try
            If Not objClaimAdjuster.CurrentAdjFlag Then
                iCount = objClaimAdjuster.Context.DbConnLookup.ExecuteInt(String.Format("SELECT COUNT(*) FROM
CLAIM_ADJUSTER WHERE CLAIM_ID = {0} AND CURRENT_ADJ_FLAG = -1", objClaimAdjuster.ClaimId))
                If (iCount < 1) Then
                    g_Warnings.Add("AdjusterWarning", "Please remember to add an adjuster record to the claim with the
current adjuster checkbox marked.")
                End If
            End If
        Catch ex As Exception
            Log.Write(ex.ToString)
        End Try


    End Sub
```

Claim After Save script to create an initial Medical type Reserve with open status of amount 1 of Workers’ Compensation claim if Reserves are not present on the Claim

```
Sub ASCClaim( objClaim As Riskmaster.Datamodel.Claim )


   Dim objReserveCurrent As Riskmaster.DataModel.ReserveCurrent

       Try

            If(objClaim.ReserveCurrentList.Count = 0) then

                              If(objClaim.LineOfBusCode = objClaim.Context.LocalCache.GetCodeId("WC",
"LINE_OF_BUSINESS")) then


                          objReserveCurrent = CType(objClaim.Context.Factory.GetDataModelObject("ReserveCurrent",
False), Riskmaster.DataModel.ReserveCurrent)

                  objReserveCurrent.ClaimId = objClaim.ClaimId

                                  objReserveCurrent.ClaimantEid = objClaim.PrimaryClaimant.ClaimantEid

                          objReserveCurrent.ReserveAmount = 1

                                  objReserveCurrent.Reason = "Initial Reserve"

                                  objReserveCurrent.ReserveTypeCode =
objReserveCurrent.Context.LocalCache.GetCodeId("MED", "RESERVE_TYPE")

                                  objReserveCurrent.ResStatusCode = objReserveCurrent.Context.LocalCache.GetCodeId("O",
"RESERVE_STATUS")

                                  objReserveCurrent.ClaimCurrencyCode = objClaim.CurrencyType

                                  objReserveCurrent.PolicyCurrencyCode = objClaim.CurrencyType

                                  objReserveCurrent.CurrencyConversionDate
=Riskmaster.Common.Conversion.ToDbDate(Now)


                          'objReserveCurrent.Save()

               End If

           End If

       Catch ex As Exception

           Log.Write(" ***ERROR*** " + ex.Message + " ***ERROR*** in (ASCClaim: ) " & objclaim.ClaimId, "Scripting",
0)

       Finally

           If Not (objReserveCurrent Is Nothing) Then

               objReserveCurrent.Dispose()

           End If


       End Try

End Sub
```

Claimant Before Save script to retrieve and store Claimant ‘s Attorney, Insurer and Attorney Firm Person Involved Row ID in global storage which are attached as Person Involved of type Others at Event Level.

```
Sub PSCClaimant(objClaimant As Riskmaster.DataModel.Claimant)


        Dim iPiRowId As Integer = 0

        Dim iEventId As Integer = 0

        Dim sLienHolderName As String

        Dim sLastName As String

        Dim sMiddleName As String

        Dim sFirstName As String

        Dim sSQL As String

        Dim arrName() As String

        Dim iEntityId As Integer

        Try

            iEventId = objClaimant.Context.DbConnLookup.ExecuteInt(String.Format("SELECT EVENT_ID FROM
CLAIM WHERE CLAIM_ID={0}", objClaimant.ClaimId))


            g_Storage("ClaimantAtt_PiRowId_" & objClaimant.ClaimId) = "0"

            g_Storage("ClaimantIns_PiRowId_" & objClaimant.ClaimId) = "0"

            g_Storage("ClaimantAttFirm_PiRowId_" & objClaimant.ClaimId) = "0"

            g_Storage("ClaimantLienHolder_PiRowId_" & objClaimant.ClaimId) = "0"

            g_Storage("ClaimantMortgagee_PiRowId_" & objClaimant.ClaimId) = "0"

            g_Storage("ClaimantPubAdjuster_PiRowId_" & objClaimant.ClaimId) = "0"

            If objClaimant.ClaimantRowId > 0 Then


                iPiRowId = objClaimant.Context.DbConnLookup.ExecuteInt(String.Format("SELECT PINLV.PI_ROW_ID
FROM PERSON_INVOLVED PINLV INNER JOIN CLAIMANT CLM ON PINLV.PI_EID = CLM.ATTORNEY_EID
WHERE PINLV.EVENT_ID = {0} AND CLM.CLAIMANT_ROW_ID = {1} AND PINLV.PI_TYPE_CODE = {2} AND
CLM.ATTORNEY_EID > 0", iEventId, objClaimant.ClaimantRowId, objClaimant.Context.LocalCache.GetCodeId("O",
"PERSON_INV_TYPE")))

                g_Storage("ClaimantAtt_PiRowId_" & objClaimant.ClaimId) = iPiRowId.ToString

                Log.Write("ClaimantAtt_PiRowId_  - " & iPiRowId.ToString)

                iPiRowId = 0
                iPiRowId = objClaimant.Context.DbConnLookup.ExecuteInt(String.Format("SELECT PINLV.PI_ROW_ID FROM PERSON_INVOLVED PINLV INNER JOIN CLAIMANT CLM ON PINLV.PI_EID = CLM.INSURER_EID WHERE PINLV.EVENT_ID = {0} AND CLM.CLAIMANT_ROW_ID = {1} AND PINLV.PI_TYPE_CODE = {2} AND CLM.INSURER_EID > 0", iEventId, objClaimant.ClaimantRowId, objClaimant.Context.LocalCache.GetCodeId("O", "PERSON_INV_TYPE")))

                g_Storage("ClaimantIns_PiRowId_" & objClaimant.ClaimId) = iPiRowId.ToString

                Log.Write("ClaimantIns_PiRowId_  - " & iPiRowId.ToString)



                iPiRowId = 0

                iPiRowId = objClaimant.Context.DbConnLookup.ExecuteInt(String.Format("SELECT PINLV.PI_ROW_ID FROM PERSON_INVOLVED PINLV INNER JOIN ENTITY ENT  ON PINLV.PI_EID = ENT.PARENT_EID WHERE PINLV.EVENT_ID = {0} And PINLV.PI_TYPE_CODE = {2} AND ENT.ENTITY_ID = (SELECT ATTORNEY_EID FROM CLAIMANT WHERE CLAIMANT_ROW_ID = {1} AND ATTORNEY_EID > 0)", iEventId, objClaimant.ClaimantRowId, objClaimant.Context.LocalCache.GetCodeId("O", "PERSON_INV_TYPE")))

                g_Storage("ClaimantAttFirm_PiRowId_" & objClaimant.ClaimId) = iPiRowId.ToString

                Log.Write("ClaimantAttFirm_PiRowId_  - " & iPiRowId.ToString)



                iPiRowId = 0



            End If

        Catch ex As Exception

            '  g_ValErrors.Add("ASCDefendant", ex.Message.ToString)

            Log.Write(ex.ToString)

        End Try


    End Sub
```

Event Initialization script to set Date of Event, Time of Event, Date Reported and Time Reported as Current Date and Time for new Event. It also sets the Event status as Open.

```
Sub InitCEvent( objEvent As Riskmaster.Datamodel.Event )

                ' Write your code here.

objEvent.dateofevent = now

objEvent.timereported = now
objEvent.timeofevent = now
objEvent.datereported= now

objEvent.eventstatuscode = g_CacheFunctions.GetCodeID("O", "EVENT_STATUS")

End Sub
```

Claim Initialization script to set Date of Claim and Time of Claim as Current Date and Time for new Event. It also sets the Claim status as Open.

```
Sub InitCClaim( objClaim As Riskmaster.Datamodel.Claim )

        ' Write your code here.
objclaim.dateofclaim = now
objclaim.timeofclaim = now

objclaim.claimstatuscode = g_CacheFunctions.GetCodeID("O", "CLAIM_STATUS")

End Sub
```

Reserve Current Validation script to throw Validation error if user try to add Indemnity Reserve on Medical claim types (MED and MO).

```
Sub ValCReserveCurrent(objReserveCurrent As Riskmaster.Datamodel.ReserveCurrent)

        ' Good idea is to reset error collection
         g_ValErrors.Clear()

     Dim objClaim As Riskmaster.DataModel.Claim
     objClaim = CType(objReserveCurrent.Context.Factory.GetDataModelObject("Claim", False),
Riskmaster.DataModel.Claim)
```

```
        If (objReserveCurrent.ClaimId > 0) Then
            objClaim.MoveTo(objReserveCurrent.ClaimId)
        End If


        ' Indemnity Reserve validation for medical claim types
        If (objReserveCurrent.Context.LocalCache.GetRelatedCodeId(objReserveCurrent.ReserveTypeCode) =
objReserveCurrent.Context.LocalCache.GetCodeId("I", "MASTER_RESERVE")) Then
                  ' MED - medical claim type
                  If (objClaim.ClaimTypeCode = objReserveCurrent.Context.LocalCache.GetCodeId("MED",
"CLAIM_TYPE")) Then
                         g_ValErrors.Add("IndemnityValForMedClaim", "Indemnity reserve is not allowed on Medical
claim type")
                  End If
                 ' MO - medical only claim type
                  If (objClaim.ClaimTypeCode = objReserveCurrent.Context.LocalCache.GetCodeId("MO",
"CLAIM_TYPE")) Then
                         g_ValErrors.Add("IndemnityValForMedClaim", "Indemnity reserve is not allowed on Medical
claim type")
                  End If


        End If


End Sub
```

DXC.technology

Follow DXC Technology on social Media

Get the insights that matter.

Keep up to date with technology and
innovation, now and in the future.