Rational. software

CRM6: Superscripts – Advanced ClearQuest Scripting Techniques

Alan Murphy
IT Specialist - IBM Rational Brand Services
IBM Certified Rational ClearQuest Administrator

IBM Rational Software Development Conference 2007

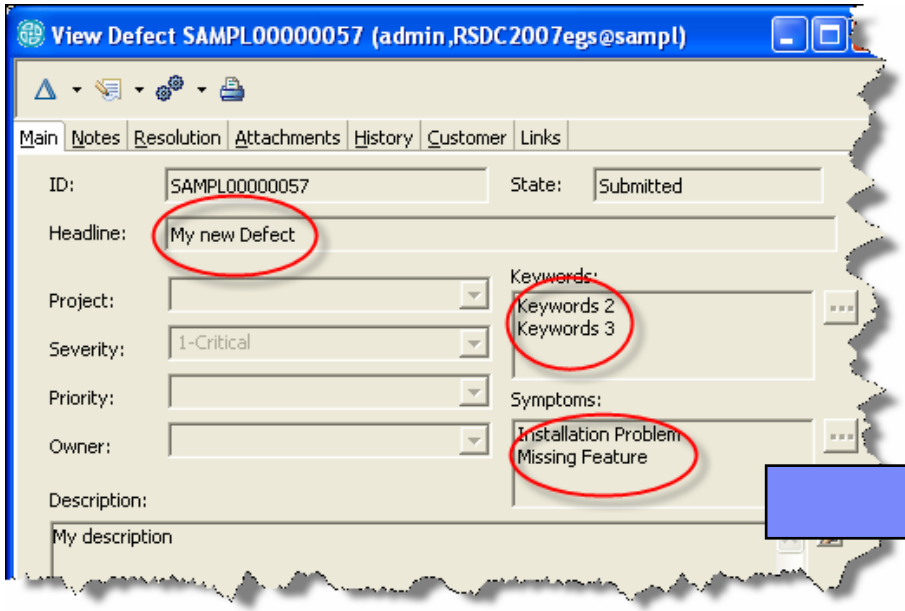
What keeps me **Rational**?

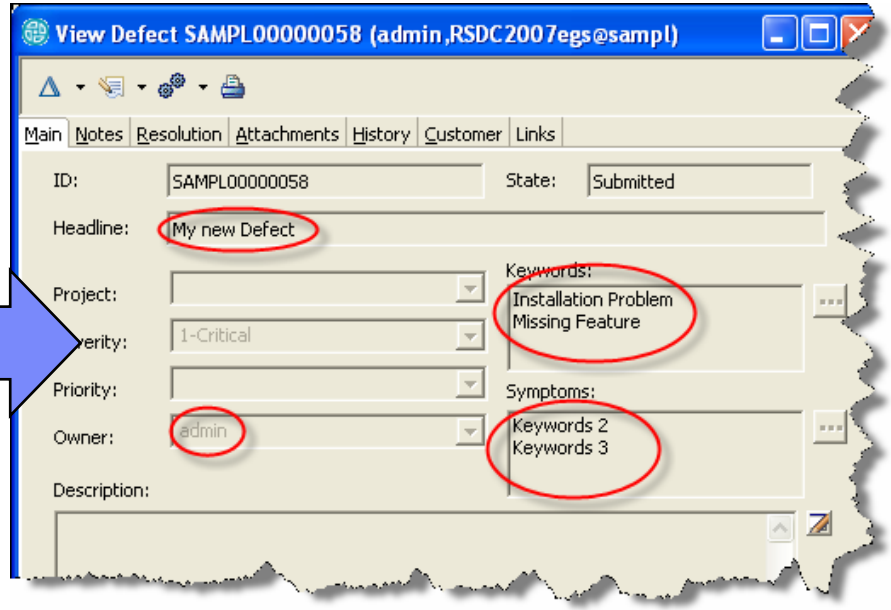
Agenda

- Understanding Script Engine Lifecycle
  - How to Copy Data to a Child Record created via a Parent/Child Control
- Adding Items to Parent Child Lists
- ClearQuest Designer Hidden Features
- Script Debugging
- Using ClearQuest MetaData
  - AdminEdit Example
- Anatomy of a Query
- Performance
  - Choice Lists
  - Data Caching
  - Fetching Record Data
  - Attachments
  - Some Tips

Copy Field Values to a New Record

- When using Parent Child relationships
  - Often it's desirable to copy some data to the newly created child
    - Child record not necessarily the Same Type as the Parent





IBM Rational ClearQuest

File Edit View Tools Help

ClearQuest Navigator

admin,RSDC2007egs@sampl
- Personal Queries
- Public Queries
- Recently Submitted
- Find Record History

ClearQuest Query Results

Query Results

Page: 1 Total Pages: 1 Total: 0

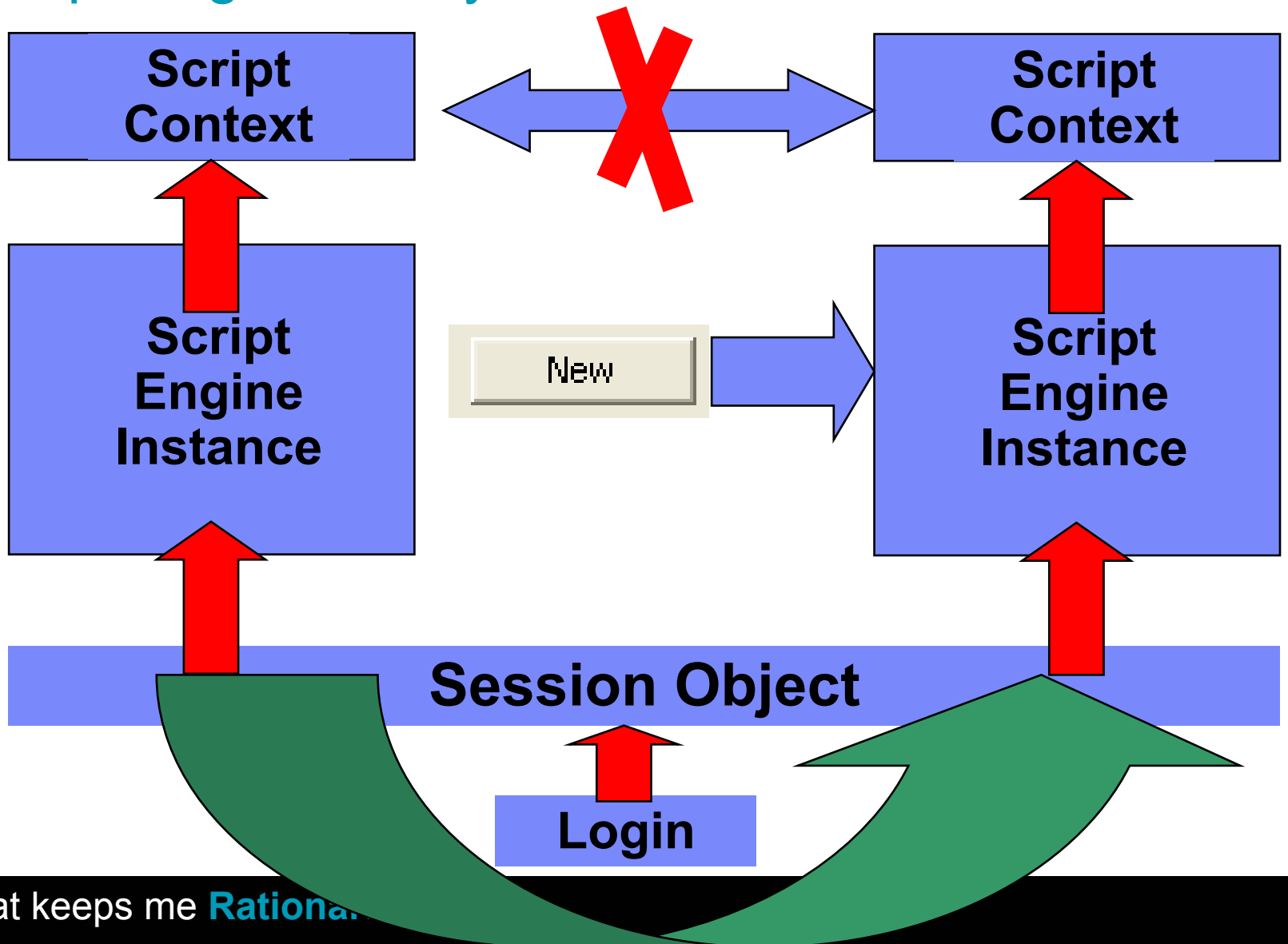
Console | Properties

Property | Value

ClearQuest Record Details | Test Results

ClearQuest Record Details

VBScript Engine Lifecycle – Observed Behaviour



So How Do We Copy Data Between Records - 1?

- We've Seen the only Common Context is the Session Object
  - Name Values
    - Collection that Behaves Like an Associative Array
      - List of Name Value Pairs
    - Can Store anything in a NameValue e.g. string, int, array, Object
      - **But** more complex items like arrays and Objects are really references to data in the script engine's context so **can't** be passed between instances. (We get a reference to nothing)
      - So we have to use a simple form like a String in this particular example
    - Two API's
      NameValue – Sets / Gets a named value
      HasValue – Tests if a particular named value exists

So How Do We Copy Data Between Records - 2?

- Break Down into a Number of Simple Steps
  1. Copy the Data We Need into the Session Object in Engine Instance 1
  2. Copy the Data We Need Back from the Session Object in Engine Instance 2
- But How ?
  1. We need some Code with the Intelligence – A Global Script to do the saving and retrieving
     - Needs it to Be Flexible – Copy Field A -> Field A or A->B
     - Reusable for any record type(s)

  2. We need to get the Data into the session object – Triggered by the push of the **New** Button – So We Need a Record Script
  3. We need to get the Data out of the session object and into the new record – Triggered by the **Submit** action – So We Need an Action Initialisation Script

So How Do We Copy Data Between Records - 3?

```
' This global script contains code to assist in the 'cloneing' or copying of data
' from one record to another

sub CacheData(CacheName, FieldToCopyList, FieldToOutputList)

    Dim MySession
    set MySession = GetSession

    ' Save the values of the specified fields into a session name value since
    ' these are passed between engine instances.  Although you can store objects in
    ' namevalues, you cant pass objects between engine instances (this includes arrays)
    ' We're going to convert the array to a string and use a series of bell characters as
    ' to delimit them
    MySession.NameValue(CacheName) = Join(GetFieldStringValues(FieldToCopyList), string(5,7))

    ' Now we'll store the map of field names for the destination of these values
    MySession.NameValue(CacheName & "_MAP") = Join(FieldToOutputList, string(5,7))
end sub

sub ReplayCachedData(CacheName)

    Dim MySession, ValueList, FieldList
    set MySession = GetSession

    ' First get the saved values and convert back to an array
    ValueList = Split(MySession.NameValue(CacheName), string(5,7))

    ' Now get the list of field names to output to
    FieldList = Split(MySession.NameValue(CacheName & "_MAP"), string(5,7))

    ' Finally set the field values
    SetFieldValues FieldList, ValueList

end sub
```

So How Do We Copy Data Between Records - 4?

Record Scripts
- Basic
  - CopyToChild

Hook Name: CopyToChild Unicode Aware

```
Function Defect_CopyToChild(param)
  ' param As Variant
  ' record type name is Defect

    ' This will be our new button pre-click hook
    ' It will setup the cache to copy fields from the current defect to the new one
    call CacheData("DEFECTDATACACHE", _
        array("keywords","symptoms","Headline","Submitter","Version"), _
        array("symptoms","keywords","Headline","Owner","note_entry"))

    ' Now say the cache is active
    GetSession.NameValue("CacheActive")= "TRUE"
End Function
```

Keywords -> Symptoms
Symptoms-> Keywords
Headline -> Headline
Submitter -> Owner
Version -> Note_Entry

So How Do We Copy Data Between Records - 5?

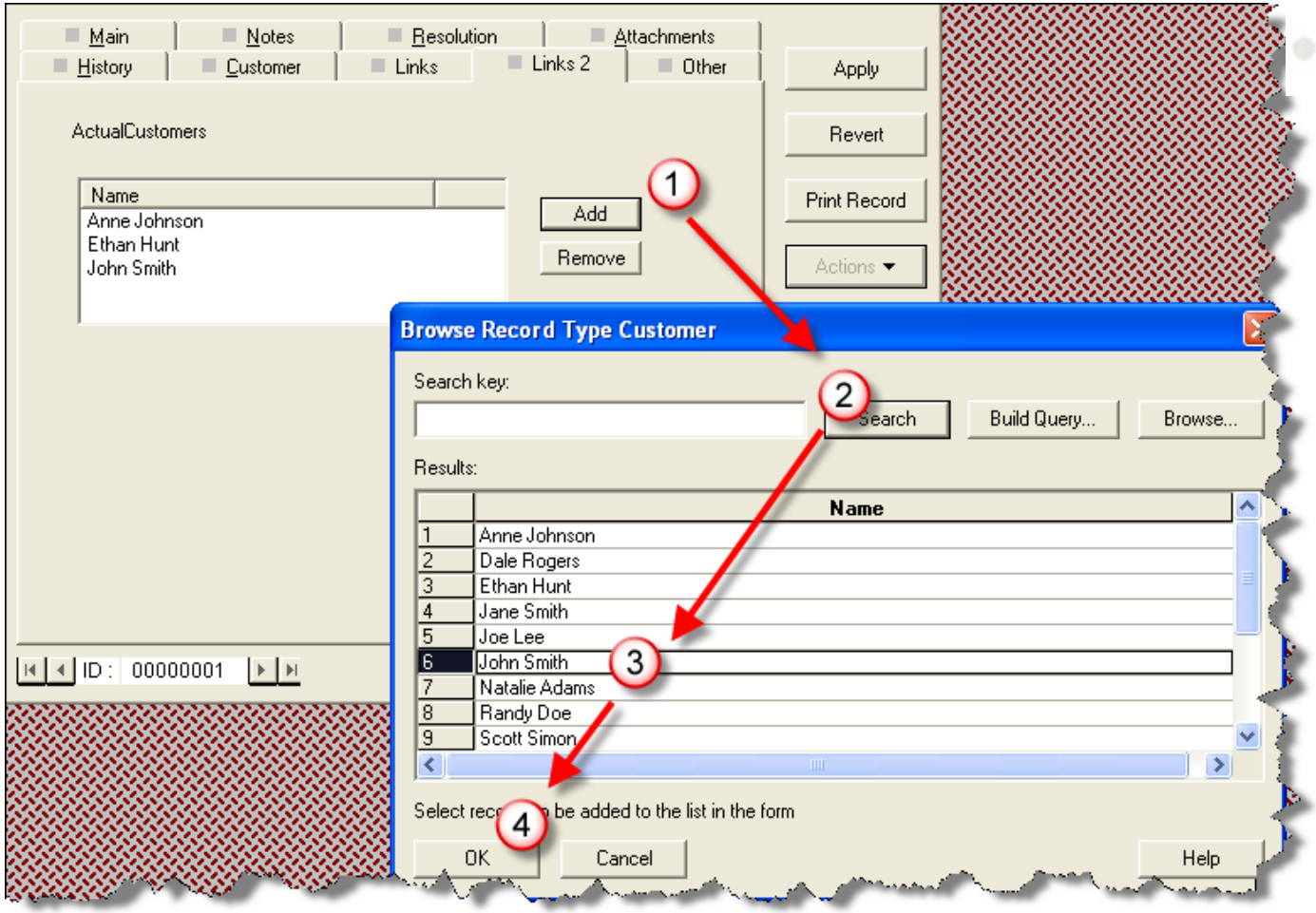
Action Name	Type	Access Control	Initialization
Submit	SUBMIT	All Users	BASIC
Assign	CHANGE_STATE	All Users	

Hook Types: ACTION_INITIALIZATION ☐ Unicode Aware

```
Sub Defect_Initialization(actionname, actiontype)
  ' actionname As String
  ' actiontype As Long
  ' action is Submit
  ' record type name is Defect

    ' This script looks to see if there are any cached values to initilaise with
    if GetSession.NameValue("CacheActive")= "TRUE" then
        call ReplayCachedData("DEFECTDATACACHE")
    end if
    GetSession.NameValue("CacheActive")=""
End Sub
```

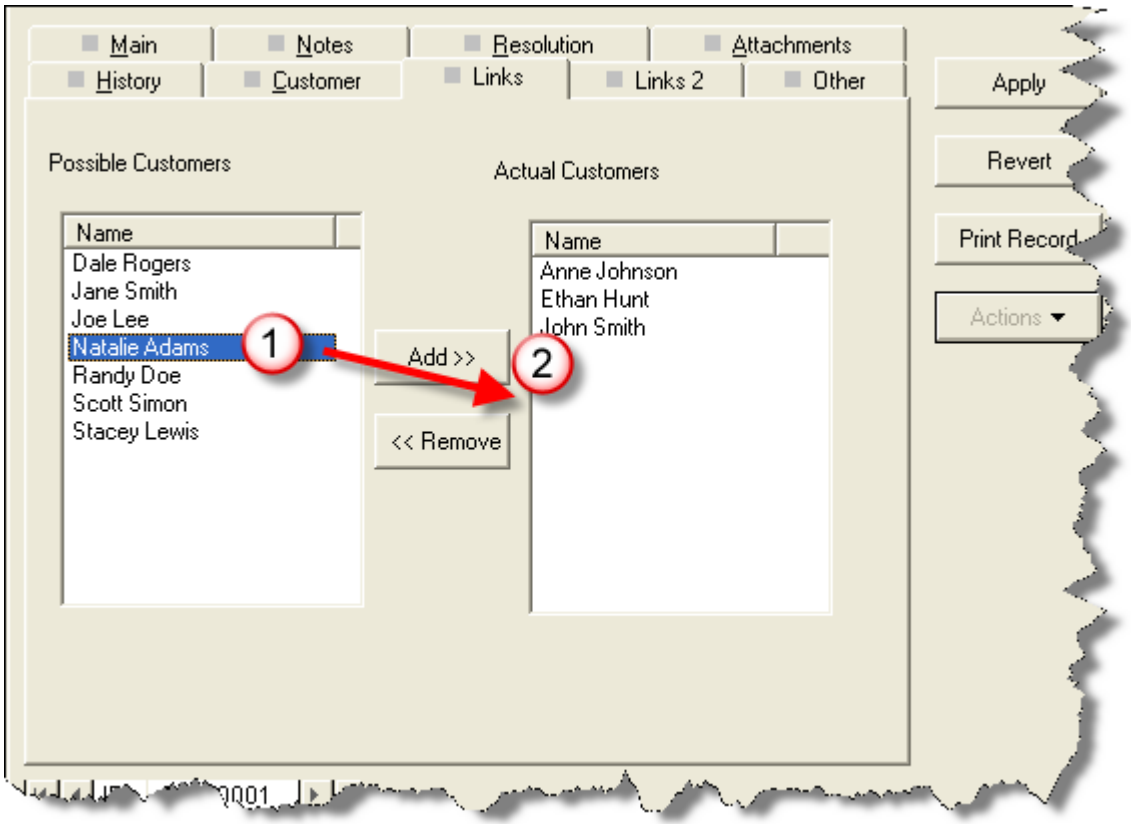
Adding Items to Parent Child Lists - 1

- Do You Sometimes wish there was another way of do this



Adding Items to Parent Child Lists - 2

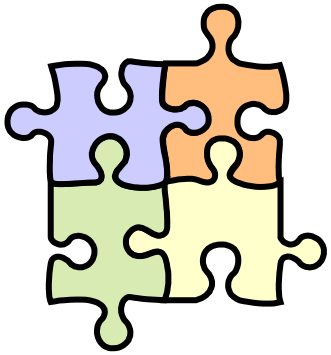
- Well There is !!



VB ONLY

Adding Items to Parent Child Lists - 3

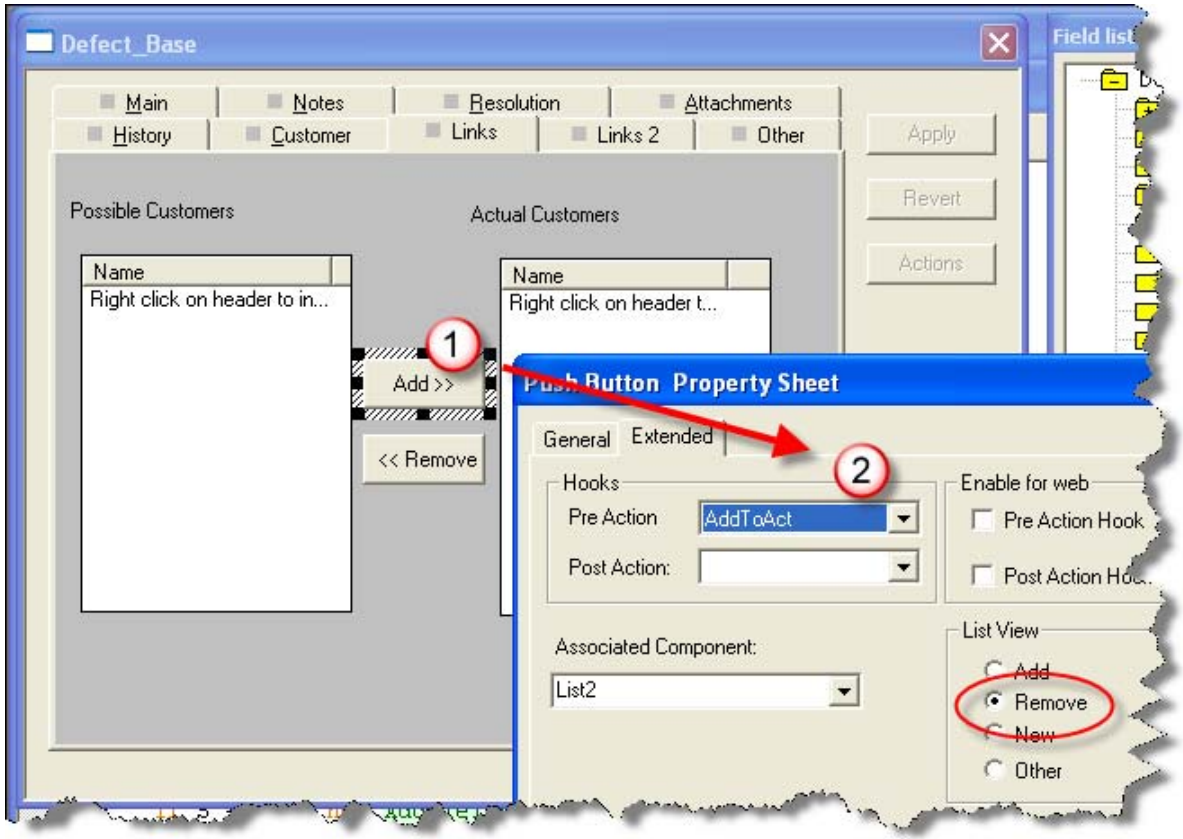
- How ??
- Need Two Parent Child Fields
  - One is the actual values you want
  - The Second is the list of Possible other Values you **Could** Add
    - This is Just the list of All **Possible** Items **Less** those Already Selected
    - Not stored in the DB
      - Filled On demand
      - Cleared before commit to the DB



- What Automation do we need
  - A couple of record scripts
  - A Global Script
  - An Action Initialisation Hook or some other event to fill the "Possibles" List

Adding Items to Parent Child Lists - 4

- Add the Parent Child Controls for the Reference Lists to the Form
  - Remove all the buttons except for the **Remove** Ones and Rename appropriately
  - The Add>> Button is the Remove button for the LH Control
  - Add two records scripts
  - Attach the record scripts to the **PreAction** hooks for the buttons
  - Each hook finds the selected record and adds it to the other list **Before** the remove happens



Adding Items to Parent Child Lists - 5

- The Code

- Record Scripts

```
Function Defect_AddToAct(param)
  ' param As Variant
  ' record type name is Defect
    S=""
    On Error Resume Next
    Sel = Param.ListSelection
    S = Sel(0)
    if S <> "" then AddFieldValue "ActualCustomers", S
End Function

Function Defect_AddToPos(param)
  ' param As Variant
  ' record type name is Defect
    S=""
    On Error Resume Next
    Sel = Param.ListSelection
    S = Sel(0)
    if S <> "" then AddFieldValue "PossibleCustomers", S
End Function
```

- Global Scripts

```
sub PopulatePossibles(PosFieldName, ActFieldName, KeyField)
    Set MySession = GetSession
    RefRecType = MySession.GetEntityDef(GetEntityDefName). _
                                        GetFieldReferenceEntityDef(ActFieldName).GetName
    set MyQuery =MySession.BuildQuery(RefRecType)
    set MyFilter = MyQuery.BuildFilterOperator(AD_BOOL_OP_AND)
    MyQuery.BuildField(KeyField)
    MyFilter.BuildFilter KeyField, AD_COMP_OP_NOT_IN, GetFieldValue(ActFieldName).GetValueAsList
    set MyResults = MySession.BuildResultSet(MyQuery)
    MyResults.Execute
    while MyResults.MoveNext = AD_SUCCESS
        AddFieldValue PosFieldName, MyResults.GetColumnValue(1)
    wend
end sub
```

ClearQuest Designer Hidden Features - 1

- Did you know the Script editor is customisable?
  - It can do a simple form of Syntax Highlighting by keywords
    - For example

```
VB Statements & keywords look like this -> for
VB Functions Looks like this -> Split
Comments -> ' This is a comment
Strings -> "Sample text"
CQMethods -> GetEntity
CQConstants -> AD_SUCCESS
CQ Properties -> fieldname
VB Built-in Objects -> Dictionary
```

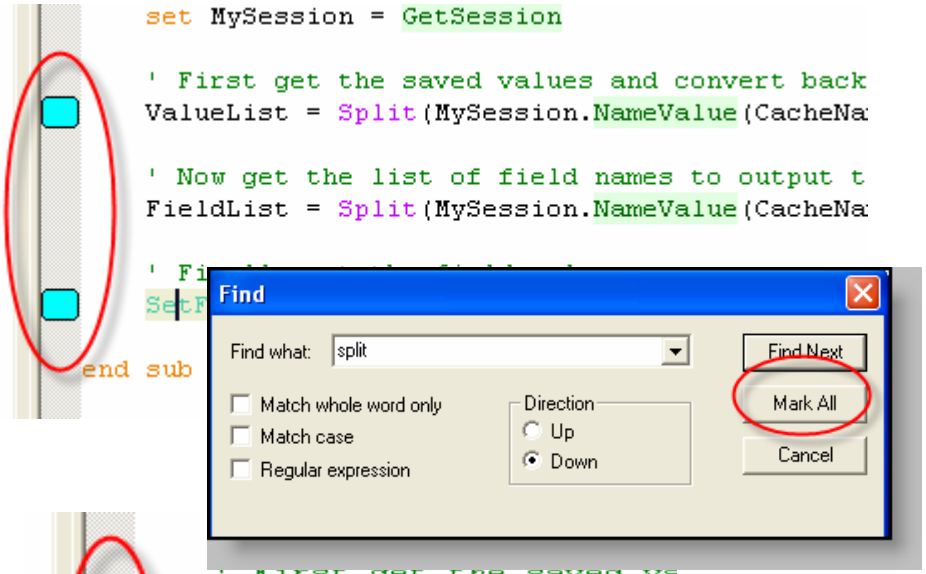
- How is it Done?
  - In ClearQuest Install Directory there are two files:
    - Vbscript.ini -> Can be edited to customise VB Script Editor
    - Perl.ini -> Can be edited to customise the Perl Script editor

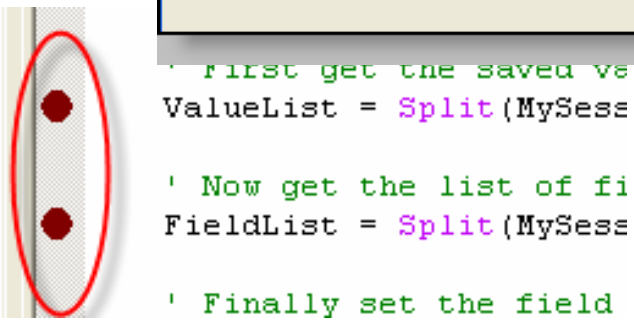
ClearQuest Designer Hidden Features - 2

- Where Do I Get One ?
  - Customisations are available for VB and Perl which include all constants & CQ API's up to CQ V7.0.1
- Why is this useful?
  - Since all the CQ API's and Constant names are included, if it isn't coloured its spelt incorrectly
  - Reduces Code / Test Cycle times
- What Doesn't it do?
  - Since it works off a word list it can't
    - Tell you if you're using an API / Constant in the wrong place

ClearQuest Designer Hidden Features - 2

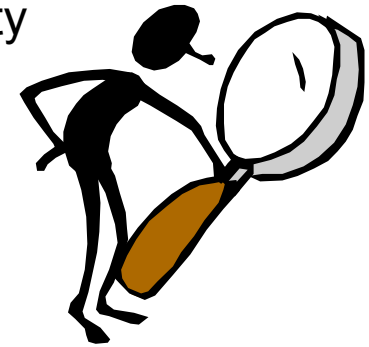
- Function Keys
  - The Script editor responds to a number of function keys, some of which are useful
    - Ctrl-F2 Toggles a Bookmark
    - F2 Jumps between Bookmarks
      - Useful with Mark All in Find…
    - F3 Jumps to next instance of string last used in Find… and highlights it
    - F9 Toggles 'Breakpoints'
      - Not used as breakpoints though
      - Really another marker





Script Debugging - 1

- How can I Debug My ClearQuest Schema?
  - You could instrument the Code with:
    - MSGBOX calls or equivalent
      - Interrupts execution – Requires Manual Intervention to Continue
      - If left in can have adverse effects e.g. on Web Interface
      - GetSession.OutputDebugString
    - Danger of Breaking Code when Removed reducing stability
  - You Could Make Use of ClearQuest Diagnostic Output
    - ClearQuest has Almost 60 Different Kinds of Trace you can choose from



- The Important thing to Remember is ClearQuest Hooks & Scripts are **interpreted** – So unless you test rigorously you may leave latent bugs.

Script Debugging - 2

- ClearQuest Provides a Wealth of Debugging Information
  - If you know where to look for it
- Control By a Set of Windows Registry Key Values
  - Key is:
    - HKEY_CURRENT_USER\Software\Rational Software\ClearQuest\Diagnostic]
  - Values are:
    - Trace
      - A list of what to Collect
    - Report
      - What info to add to the output , e.g times, sequence numbers etc
    - Output
      - Where to Send it
    - There are others that we're not Concerned with here

- NOTE: ClearQuest tools only read these keys when they are started.

Script Debugging – 3 - Values of Report Parameter

- "Report"="DIAG_FLAGS=-1;MESSAGE_INFO=0xC01"

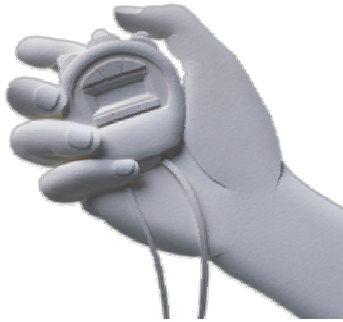
Makes ClearQuest Output the current setting of all debug flags

0x000001 – message number
0x000002 – PID
0x000004 – PPID on Unix, unused on Win32
0x000008 – Thread ID
0x000010 – machine name (Unix only)
0x000020 – PGID on Unix, unused on Win32
0x000100 – time (sortable format)
0x000200 – date (sortable format)
0x000400 – seconds since initial debug message
0x000800 – seconds since previous debug message
0x001000 – add label for each item in prefix
0x010000 – thread stack size (Unix only)

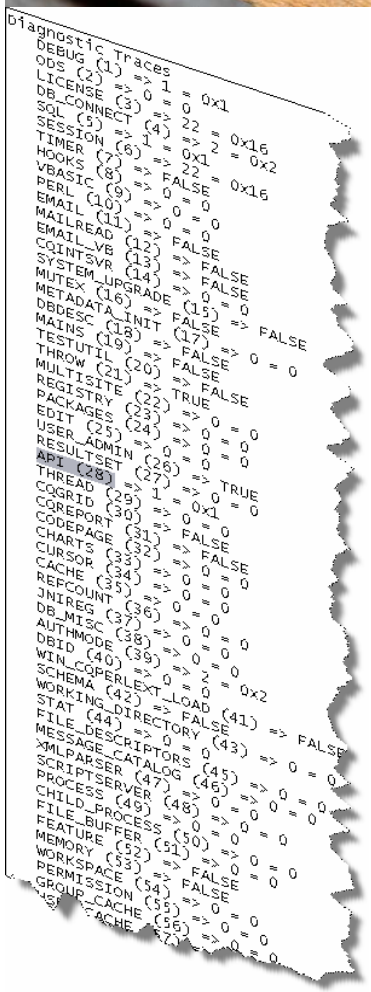
```
720: [CQ 4.031, +0.000, 487] session admin(PRIVATE_SESSION,soph,720,
720: [CQ 4.031, +0.000, 488] SQL: select count(*) from ratl_replicas
720: [CQ 4.031, +0.000, 489] SQL: select dbid, is_active, version,
720: [CQ 4.047, +0.016, 490] session admin(PRIVATE_SESSION,soph,720
720: [CQ 4.063, +0.016, 491] CQLicenseMgr: feature ClearQuest, versi
720: [CQ 4.078, +0.015, 492] CQLicenseMgr: setUserName admin
720: [CQ 4.078, +0.000, 493] CQLicenseMgr: asking for a license
720: [CQ 9.109, +5.031, 494] CQLicenseMgr: obtained a license: Succe
720: CQResourceDLL::LoadLangDLL: Trying C:\Program Files\Rational\C
720: CQResourceDLL::LoadLangDLL: Trying C:\Program Files\Rational\C
720: CQResourceDLL::LoadLangDLL: Trying C:\Program Files\Rational\C
720: [CQ 9.141, +0.032, 495] SQL: select dbid,name,type,subtype,par
720: [CQ 9.156, +0.015, 496] session admin(PRIVATE_SESSION,soph,72
```

Elapsed Time Since start

Time Since Last Trace



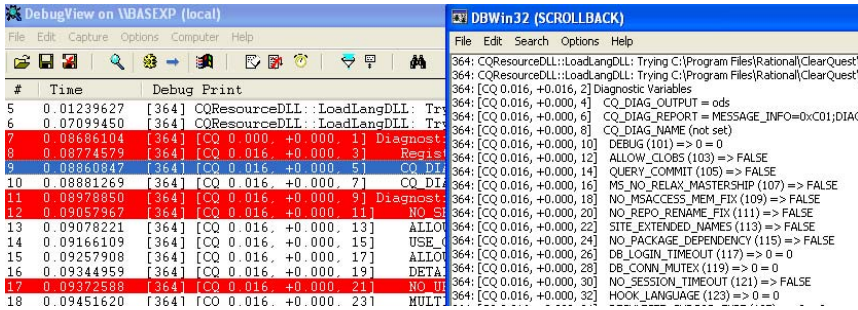
Script Debugging - 4



- Of the Almost 60 Different Kinds of Trace you can choose from we'll Concern ourselves with these
  - HOOKS
    - Annotates the output with which hook is firing
  - THROW
    - Reports Exceptions thrown by the ClearQuest Core
  - API
    - Includes all ClearQuest API calls in the trace showing their parameters and return values – Very Useful
  - SQL
    - Shows all Calls made to the Database

Script Debugging - 5

- How do I collect the Trace Output?
  - The **Output** Registry value can be set to:
    - ODS – OutputDebugString – which can be collected with a suitable monitor utility such as:
      - DBWin32[1] – Provided with ClearQuest; or
      - Debugview – A much more capable free Utility from
        - http://www.sysinternals.com/utilities/debugview.mspx
        - Output can be filtered and highlighted
        - Keeps up with the Trace Output better than DBWin32
        - Timestamps Output
    - A Filename to collect the trace in



[1]NOTE: You must have administrator rights to run dbwin32 as it needs access to a system object.

Script Debugging – 5.5

Debug Flags can be set on either platform via Environment Variables

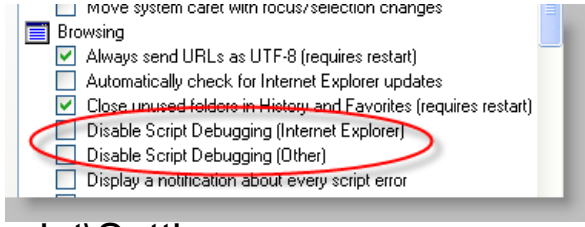
On Unix – Debug flags are set like this:

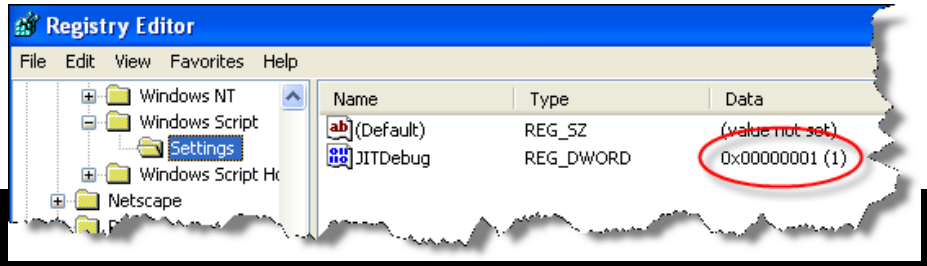
- **setenv CQ_DIAG_TRACE Throw;Db_Connect=2;SQL=2;API**
- **setenv CQ_DIAG_REPORT MESSAGE_INFO=0x70B**
- **setenv CQ_DIAG_OUTPUT trace.log**

- **On Windows like this:**
  - **set CQ_DIAG_TRACE=Throw;Db_Connect=2;SQL=2;API**
  - **set CQ_DIAG_REPORT=MESSAGE_INFO=0x70B**
  - **set CQ_DIAG_OUTPUT=c:\trace.log**

Script Debugging - 6

- Scripting Language is VBScript ?
  - You can use the Microsoft Script Debugger or
  - Use the Debugger from Visual Studio 6 Visual Interdev Component
- To Use the Microsoft Script Debugger
  - Download the Script Debugger engine and Install
    - http://www.microsoft.com/downloads/details.aspx?FamilyID=2f465be0-94fd-4569-b3c4-dffdf19ccd99&displaylang=en
  - Enable The Debugger
    - First Enable in Internet Explorer – Tools -> Internet Options .. -> Advanced
    - Ensure the Disable Script debugging options are unticked
    - Enable Just-In-Time debugging
    - Run RegEdit and edit this key
      - HKEY_CURRENT_USER\Software\Microsoft\Windows Script\Settings
      - Set JITDebug = 1

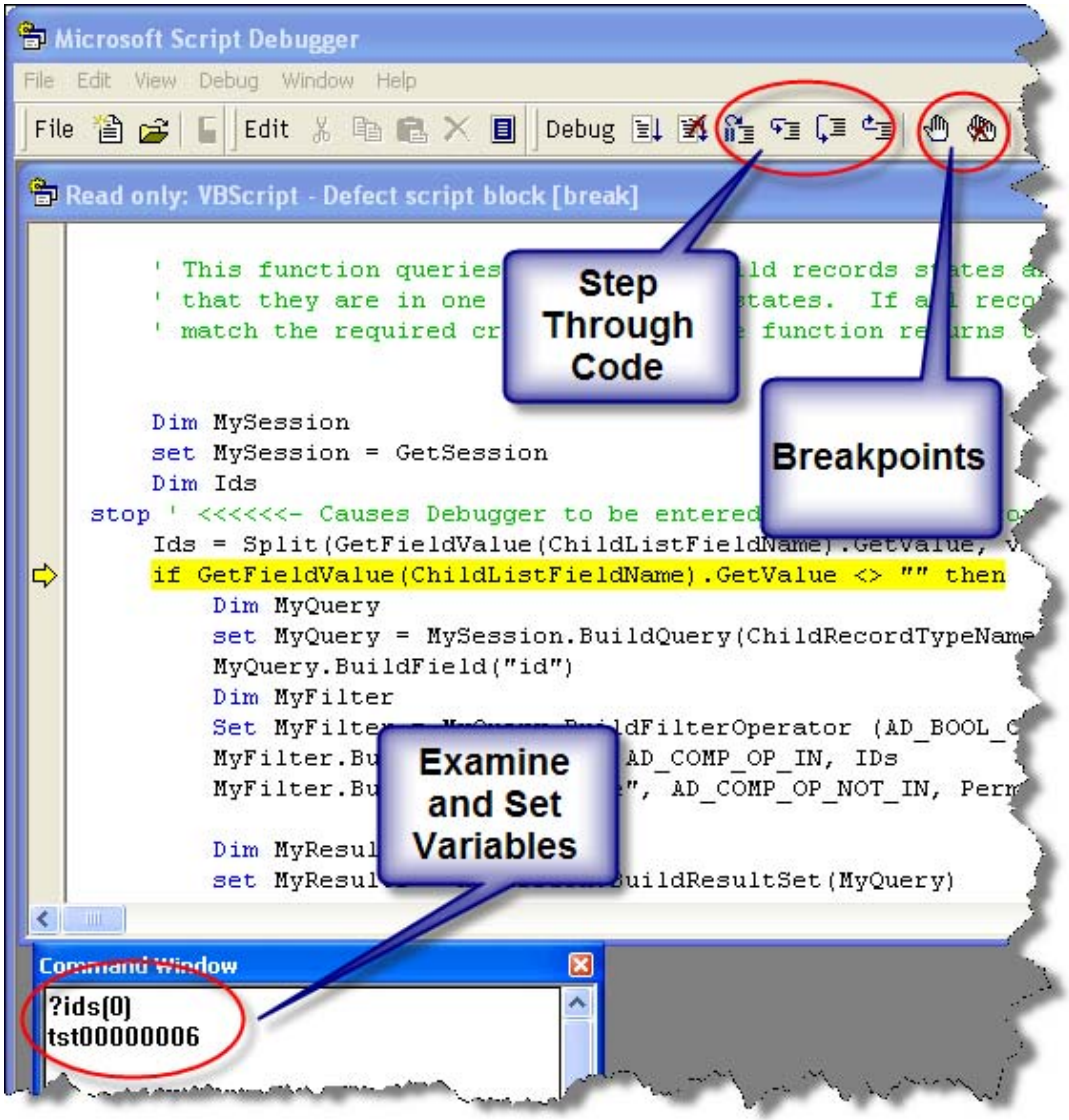




Script Debugging - 7

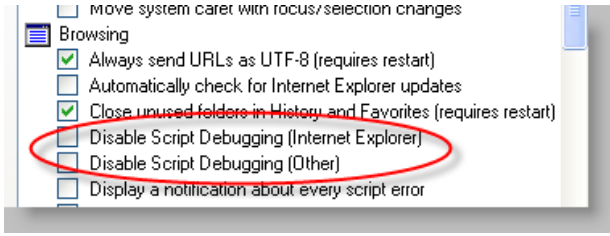
- Get in and Examine live code
- Set Breakpoints!!
- Single Step!!

- STOP Statement causes debugger to be entered as do any ClearQuest or other VBScript Errors.

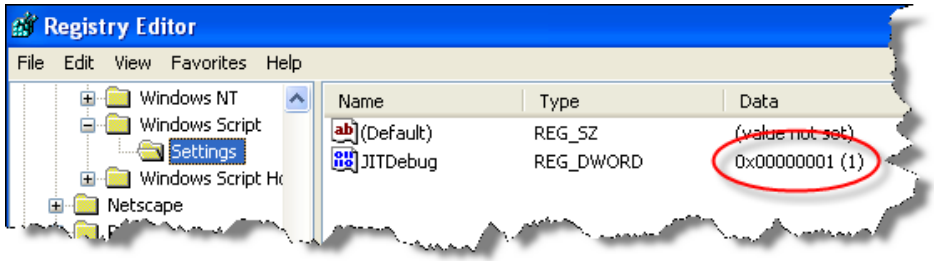


Script Debugging - 8

- Lucky Enough to have Visual Interdev from Visual Studio 6?
  - Instead of Installing Microsoft Script Debugger
  - Install Visual Interdev instead
  - Enable The Debugger
    - First Enable in Internet Explorer – Tools -> Internet Options .. -> Advanced
    - Ensure the Disable Script debugging options are un-ticked

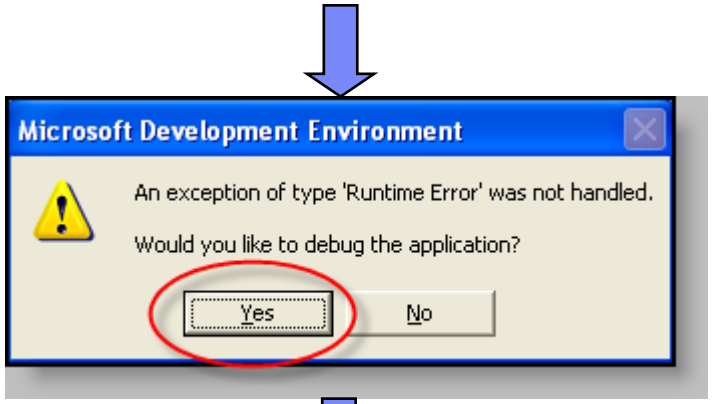


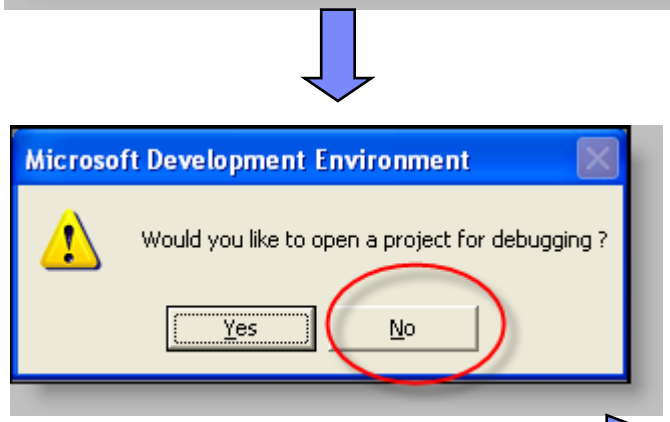
  - Enable Just-In-Time debugging
    - Run RegEdit and edit this key
      - HKEY_CURRENT_USER\Software\Microsoft\Windows Script\Settings
      - Set JITDebug = 1

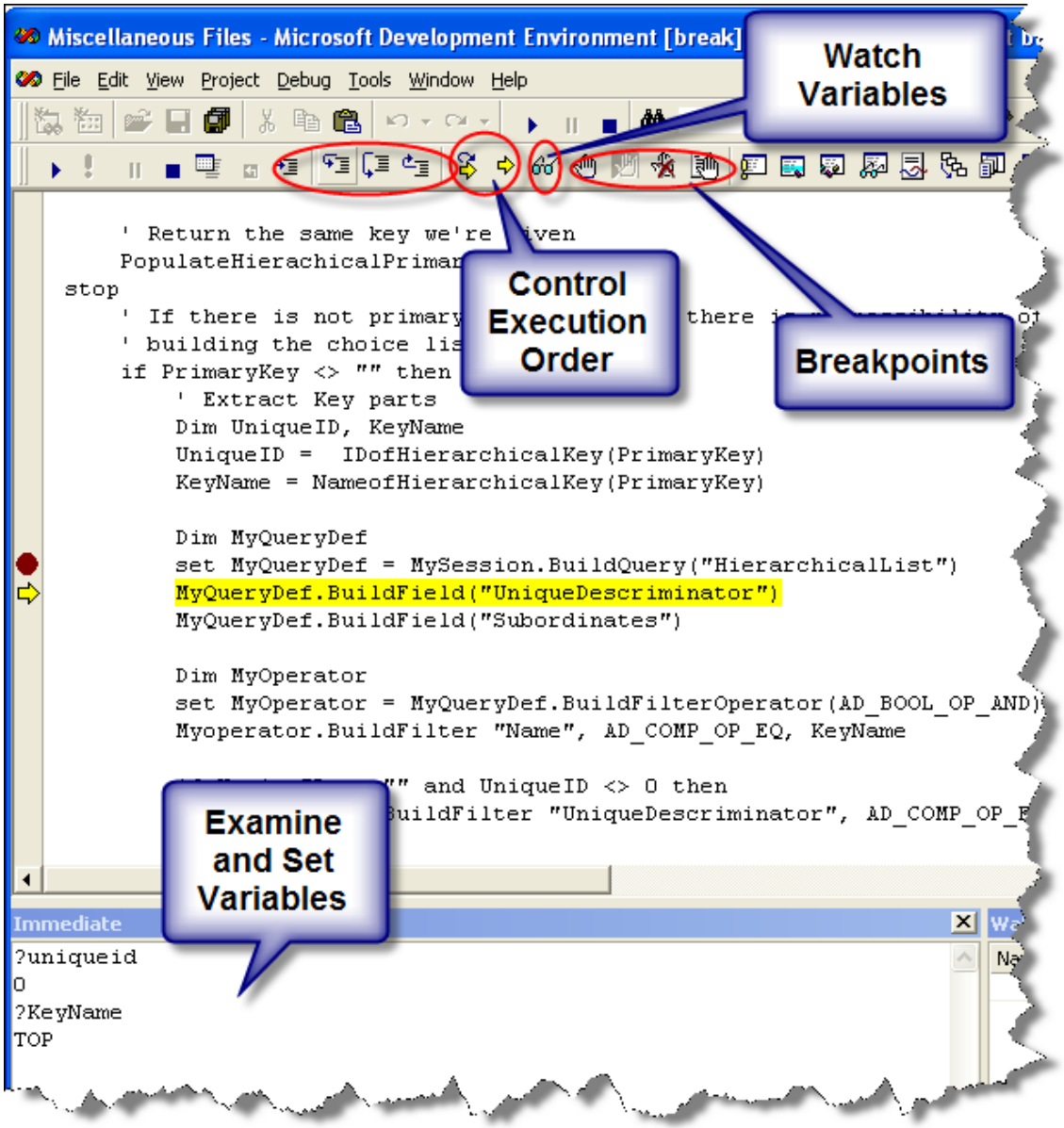


Script Debugging - 9

- When an Error Occurs

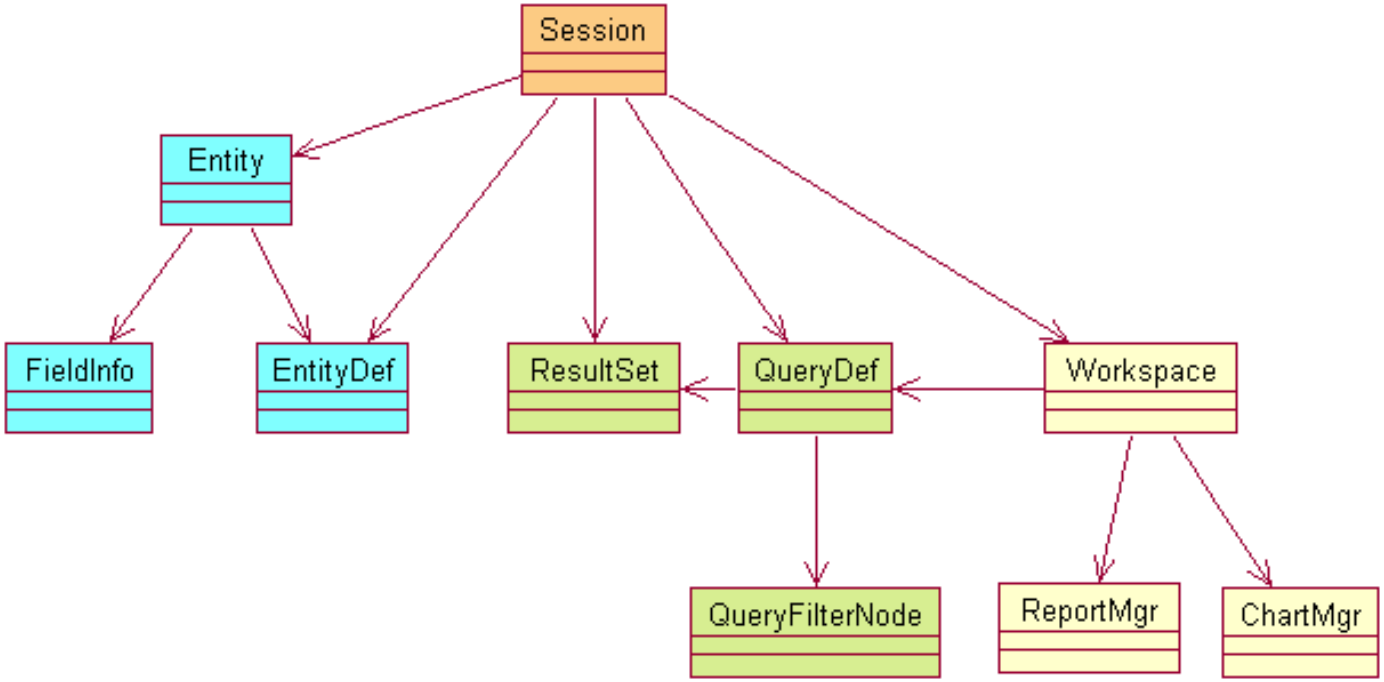


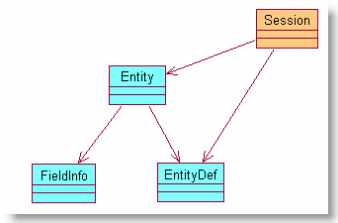




Using ClearQuest MetaData - 1

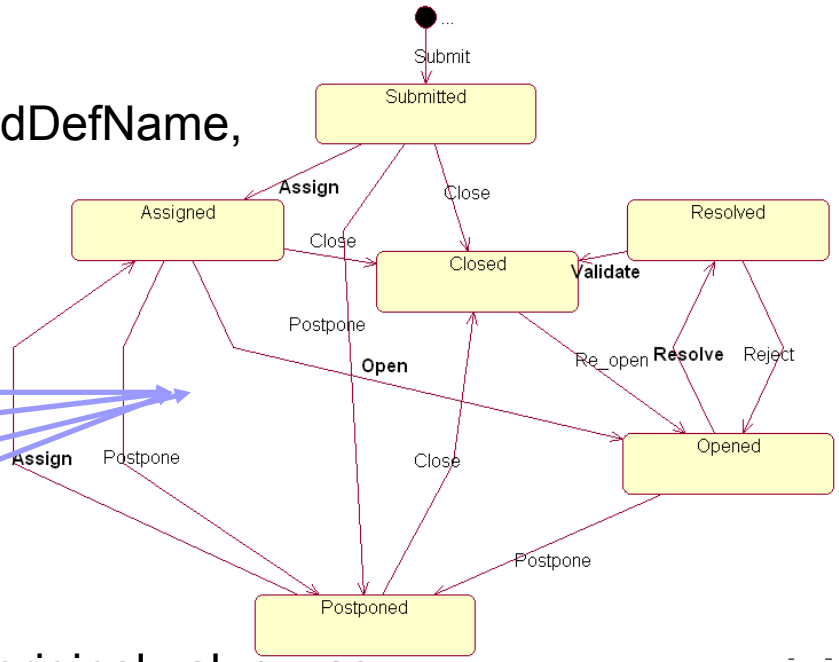
- The ClearQuest API Provides a Wealth of Data about your Schema





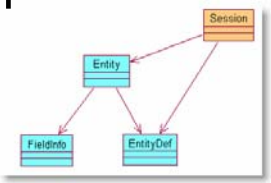
Using ClearQuest MetaData - 2

- The EntityDef Object provides APIs to discover things such as
  - Names of all fields in the entity type
    - GetFieldDefNames
  - Type of a field
    - GetFieldDefType, IsSystemOwnedFieldDefName, GetFieldReferenceEntityDef
  - Discovering the Workflow using
    - GetActionDestStateName, GetActionSourceStateNames, GetDefaultActionName
  - Discovering States in the Workflow
    - GetStateDefNames
  - If a field has been modified and what its original value was
    - ValueChangedThisAction, GetFieldOriginalValue



Using ClearQuest MetaData - 3

- Common Problem in Schemas
  - Especially Complex ones or
  - Where Data Import has been performed
- Fields can have Missing / Incorrect Data In them
- The Workflow won't allow errors to be corrected
  - Usually because the Fields are read-only
- Result:
  - Can't commit updates to affected Records
- Solution:
  - Add a new action for administrators to be able to edit any (or almost any) Field
- How do we Script this in a reusable manner that won't break when we add, remove or rename fields?



AdminEdit Example - 1

Action Name	Type	Access Control	Initialization	Validation	Commit	Notificatio
Submit	SUBMIT	All Users				
Import	IMPORT	All Users				
AdminEdit	MODIFY	User Groups	BASIC			

User Groups

Select User Groups:

- [ ] EmailPlusAdmins
- [x] ProjectRoleAdmins

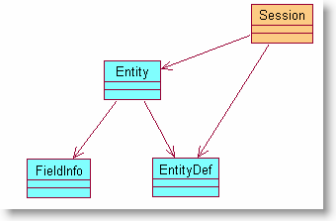
Ok Cancel Help

Hook Types: ACTION_INITIALIZATION [ ] Unicode Aware

```
Sub placeholder_Initialization(actionname, actiontype)
  ' actionname As String
  ' actiontype As Long
  ' action is AdminEdit
  ' record type name is placeholder

    'Make all fields editable except those in the array passed
    Call AE_AdminEdit(array("Notes_Log", "Priority"))
End Sub
```

AdminEdit Example - 2

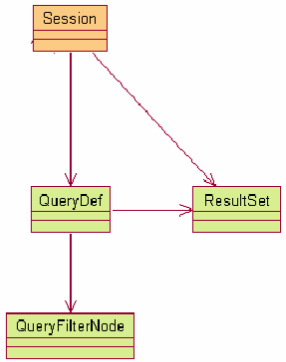
```
sub AE_AdminEdit(ExceptionList)


    Dim MySession
    Set MySession = GetSession()

    ' get the list of fields for the current record type
    Dim EntityName
    EntityName = GetEntityDefName

    Dim MyEntityDef
    Set MyEntityDef = MySession.GetEntityDef(EntityName)

    Dim FieldList
    FieldList = MyEntityDef.GetFieldDefNames()

    ' Now check each field to see if its in the exception list
    ' and mark those that aren't as AD_OPTIONAL.
    Dim FL, EL
    For Each FL in FieldList
        Dim FoundException
        FoundException = False
        For Each EL in ExceptionList
            if FL = EL then
                FoundException = True
                exit for
            end if
        Next
        if not FoundException then SetFieldRequiredNessForCurrentAction FL, AD_OPTIONAL
    Next
end Sub
```



Anatomy of a Query - 1 – Simple Query

- A Simple Query Consists of
  - A List of fields you want values for
  - A Results Set
    - Returns the requested fields for ***All*** records of a given type
    - You can then iterate over the results sets and process the data

```
Set MyQuery = GetSession.BuildQuery("Users")
MyQuery.BuildField("Login_name") ' Column 1
MyQuery.BuildField("fullname") ' Column 2
set MyResults = GetSession.BuildResultSet(MyQuery)
MyResults.Execute
While MyResults.MoveNext = AD_SUCCESS
    Line = MyResults.GetColumnValue(1) & " : " & _
                MyResults.GetColumnValue(2) & VBCrlf
wend
```



Anatomy of a Query - 2 – Query with Filter

- More Useful
  - Focused Question
  - Useful for Choice List Hooks
  - Add a Filter to Restrict Results – Lets look at Members of a Group Testers

```
Set MyQuery = GetSession.BuildQuery("Users")
MyQuery.BuildField("Login_name") ' Column 1
MyQuery.BuildField("fullname") ' Column 2
set FilterOp = MyQuery.BuildFilterOperator(AD_COMP_OP_AND)
FilterOp.BuildFilter "Groups.Name", AD_COMP_OP_EQ, "Testers"
set MyResults = GetSession.BuildResultSet(MyQuery)
MyResults.Execute
While MyResults.MoveNext = AD_SUCCESS
    Line = MyResults.GetColumnValue(1) & " : " & _
                MyResults.GetColumnValue(2) & VBCrlf
wend
```

Anatomy of a Query - 3 – Sorting by Field(s)

- How Do we Sort the results set?

```
Set MyQuery = GetSession.BuildQuery("Users")
MyQuery.BuildField("Login_name") ' Column 1
MyQuery.BuildField("fullname") ' Column 2
set MyField = MyQuery.QueryFieldDefs.Item("Login_Name")
MyField.SortType AD_SORT_ASC
MyField.Sortorder 1
set FilterOp = MyQuery.BuildFilterOperator(AD_COMP_OP_AND)
FilterOp.BuildFilter "Groups.Name", AD_COMP_OP_EQ, "Testers"
set MyResults = GetSession.BuildResultSet(MyQuery)
MyResults.Execute
While MyResults.MoveNext = AD_SUCCESS
    Line = Line & MyResults.GetColumnValue(1) & " : " & _
                  MyResults.GetColumnValue(2) & VBCrlf
wend
```

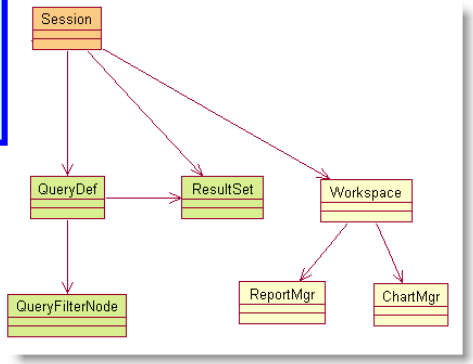
Anatomy of a Query - 4

- How Do I Run a Query from the Workspace?

```
Set MySession = GetSession
Set MyWksp = MySession.GetWorkSpace
set MyQuery = MyWksp.GetQueryDef("Public Queries/All Defects")
set MyResults = MySession.BuildResultSet(MyQuery)
MyResults.Execute
.....
```

- How Many Records did My Query Return?

```
Set MyQuery = GetSession.BuildQuery("Users")
MyQuery.BuildField("Login_name") ' Column 1
set MyResults = GetSession.BuildResultSet(MyQuery)
MyResults.EnableRecordCount
MyResults.Execute
RecCount = MyResults.RecordCount
```



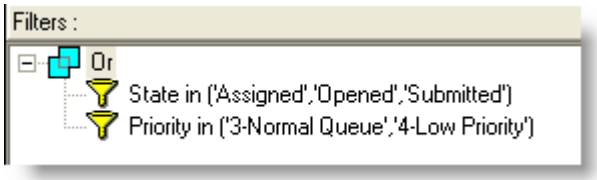
How Do I Modify an Existing Workspace Query?

- Add a New Filter Node and Save the Modified Query

```
Set MySession = GetSession
Set MyWksp = MySession.GetWorkSpace
set MyQuery = MyWksp.GetQueryDef("Public Queries/Unresolved Defects")
set NewNode = MyQuery.CreateTopNode (AD_BOOL_OP_OR)
NewNode.BuildFilter "Priority", AD_COMP_OP_IN, _
                      array("3-Normal Queue","4-Low Priority")
MyWksp.Savequerydef "Unresolved Defects", "Public Queries", _
                     MyQuery, true
```

Before

After



Anatomy of a Query – 5 – Stateless Records

- Building a Query using a records Unique Key
- Gotcha
  - Normally all Fields added to a Query are returned
  - Particularly Useful where the record has a multipart key
  - BuildUniqueKey Isn't Automatically Included
    - Need to use isShown Method to cause it to be returned

```
Set MyQuery = GetSession.BuildQuery("Project")
set UniqueKeyField = MyQuery.BuildUniqueKeyField
UniqueKeyField.IsShown = True
MyQuery.BuildField("Manager")
set MyResults = GetSession.BuildResultSet(MyQuery)
MyResults.Execute
While MyResults.MoveNext = AD_SUCCESS
    Line = MyResults.GetColumnValue(1) & " : " & _
                MyResults.GetColumnValue(2) & VBCrlf
wend
```

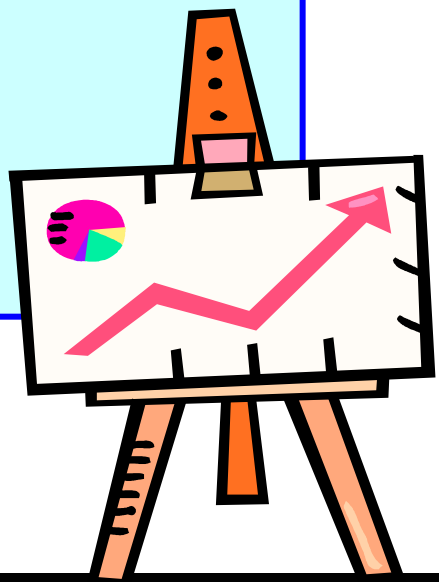
How Do I Run a Report?

- You Can Execute Reports through the ClearQuest API
  - Only two format Choices
    - HTML 3.2 or **HTML 4.0** New in 7.x
      - HTML 4.0 gives a much better layout
    - Can only save to a File

```
use strict;
use CQPerlExt;
my $reportName = "Personal Queries/Sample_report";
my $htmlPath = "c:\\Temp\\my-report.html";
my $Fmt = 2; # 1->HTML3.2, 2->HTML 4.0
my $session = CQSession::Build();
$session->UserLogon ($session, "admin", "", "SAMPL", "");
my $workspace = $session->GetWorkSpace();
my $reportMgr = $workspace->GetReportMgr($reportName);
$reportMgr->SetHTMLFileName($htmlPath);
$reportMgr->SetFormat($Fmt);
$reportMgr->ExecuteReport();
CQSession::Unbuild($session);
```

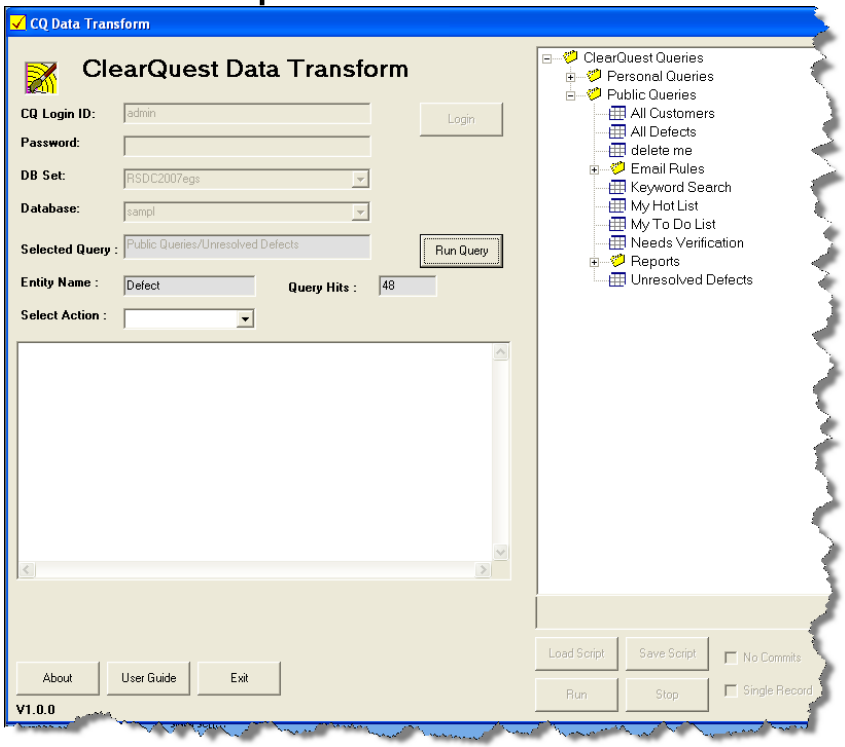
How Do I Run a Chart?

```
use strict;
use CQPerlExt;
my $session = CQSession::Build();
my $user = "admin";
my $pass = "";
my $db = "SAMPL";
my $dbset = "";
$session->UserLogon($user, $pass, $db, $dbset);
my $wkSpc = $session->GetWorkSpace();
my $chartDef = $wkSpc->GetChartDef("Personal Queries/Sample_Chart");
my $resultSet = $session->BuildResultSet($chartDef);
$resultSet->Execute();
my $chartMgr = $wkSpc->GetChartMgr();
$chartMgr->SetResultSet($resultSet);
$chartMgr->MakeJPEG("C:\\Temp\\BBChart.jpg");
CQSession::Unbuild($session);
```



What Sorts of Things Could I Do with this Knowledge?

- Once You Understand how to Manipulate the Workspace & Queries
  - All Sorts of Things Become Possible
    - Time Based Queries via a Scheduled Job
    - Publishing Reports / Charts to Website
    - Build RSS Feeds
    - Do Intelligent Data Transforms
    - …. and much more



- How Do I Create a Session Object?
  - Only needed in standalone Apps

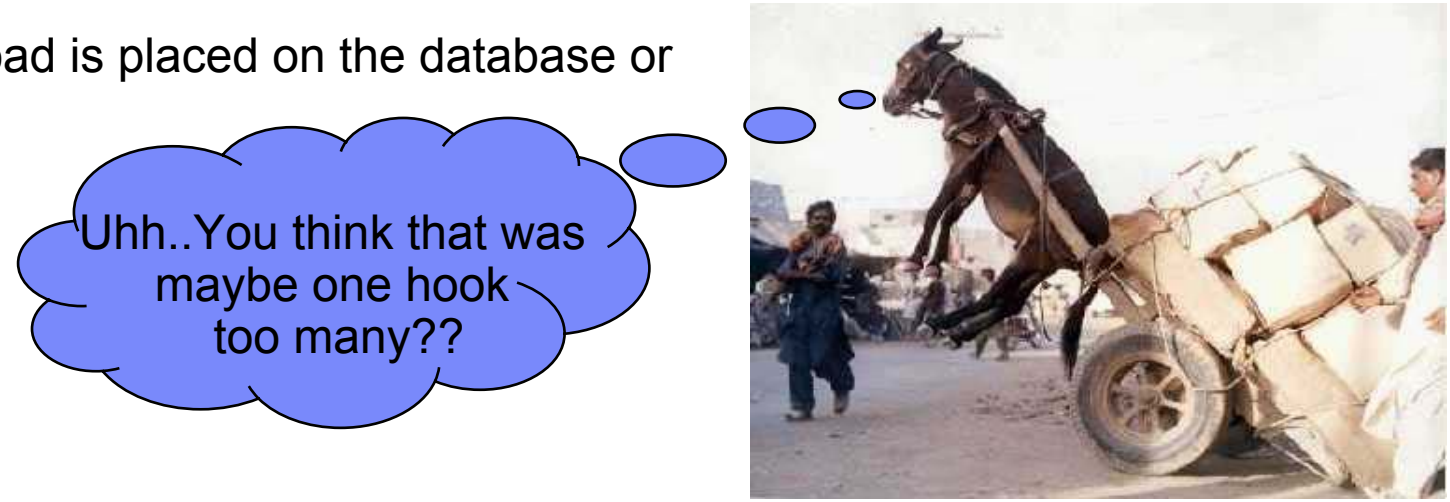
```
Set MySession = CreateObject("ClearQuest.Session")
MySession.UserLogon login_name, password, database_name, _
                                    AD_PRIVATE_SESSION, database_set
```

How Do I Create a Session Object in Perl?

```
use CQPerlExt;
$CQsession = CQSession::Build();
$CQsession->UserLogon(loginID, password, dbname, dbsetname);

…

CQSession::Unbuild($session);
```

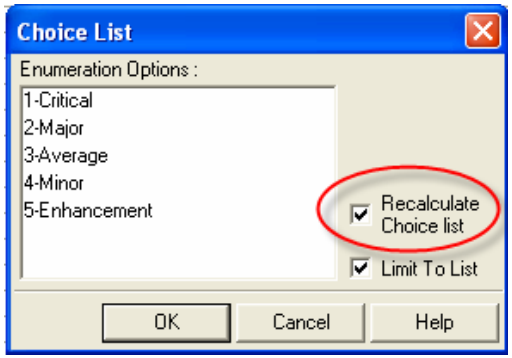
Performance 1 – How to Avoid Common Problems

- Certain ClearQuest Features / Operations are relatively expensive
- Unfortunately, often the easiest/obvious way to do something is often the least efficient.
  - The price of the easy option!!
- Hooks Execute Client Side
  - So Most of the Load is carried by the Client and thus Distributed - **except**
    - When it's a web client or
    - An extra load is placed on the database or
    - Both



Performance 2 – Recalculate Choice List

- Makes Sure the ChoiceList for a field is always up to date
  - Runs the Hook **every** time **any** field changes
  - Potentially Expensive in Performance Terms
    - Especially if there are a lot of them
    - Especially if you have cascading hooks
      - E.g. Change to one field causes the value of other field(s) to change
- Pointless for Static Lists (Constant Lists & Dynamic Lists)
- If the hook populates the list by reference to the database
  - Hook may make many trips to the database for the same data
- So we have Two Issues:
  1. Reduce number of hook executions
  2. Eliminate multiple trips to the database



Performance 3 – Recalculate Choice List

- How do we Avoid Recalculate Choice List?
  - If you've deployed ClearQuest Web you are already familiar with explicitly specifying field dependencies

  - The Values of the choice list for **Field B** are dependent on some way on the value of **Field A**.
  - So if the Value of **Field A** Changes the choice list for **Field B** must also change
  - Suggests a **Field Value Changed** hook is required

```
Sub fielda_ValueChanged(fieldname)
  ' fieldname As String
  ' record type name is Defect
  ' field name is FIeldA

    InvalidateFieldChoiceList "FieldB"
End Sub
```

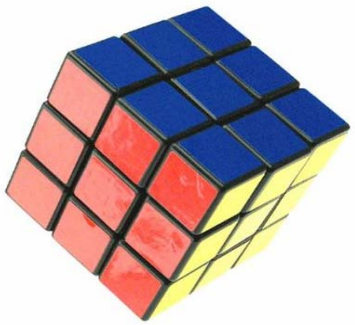
Performance 4 – Recalculate Choice List

- So How do We Eliminate Multiple Trips to the Database?
  - Typically choice list hooks (where a script is involved) run a query to populate the choicelist
    - Either Directly or indirectly
- If we can make the assumption that:
  - For all intent and purpose, the list is static once computed for the duration of a ClearQuest login Session
    - E.g. Members of a ClearQuest Group
- We can cache the values and reuse these later
- But How?
  - ClearQuest Session Object provides an Associative array we can use accessed by two API calls
    - **HasValue** – Allows us to determine is a particular named value exists
    - **NameValue** – Allows us to get/set the value

Performance 5 – Recalculate Choice List

- Typical Framework for a ChoiceList hook that implements caching

```
Sub fielda_ChoiceList(fieldname, choices)
  ' fieldname As String
  ' choices As Object
  ' record type name is Defect
  ' field name is FIeldA

    if not GetSession.HasValue("<<Cachename>>") then

        ' Get the data
        ' ...

        ' Build the cache - ValueList is an array of item values
        GetSession.NameValue("<<Cachename>>") = Join(ValueList, vblf)
    end if

    ' Now we know we always have a cache
    CacheVals = Split(GetSession.NameValue("<<Cachename>>") , vblf)
    for each v in CacheVals
        Choices.AddItem v
    next

End Sub
```



Performance 6 – Membership of a Group

- A Often used choice list hook is membership of a ClearQuest Group
  - A look at the API Guide will quickly reveal that there is a Groups Object and from that you can find a groups membership
    - The **obvious** choice!!
- The Catch
  - Requires the use of an Admin Session
    - Which we don't have
    - Creating an admin session requires
      - Logging in the admin session
        - Reveals a password in the schema
      - Loads another part of ClearQuest
  - It's **very slow** in comparative terms
  - Reports all groups members, not just those subscribed to the current database

```
Sub fielda_ChoiceList(fieldname, choices)
  ' fieldname As String
  ' choices As Object
  ' record type name is Defect
  ' field name is FIeldA

    set adminSession = CreateObject("ClearQuest.AdminSession")
    adminSession.Logon "admin", "admin", ""
    groupmembers = adminSession.GetGroup ("Engineers").Users
    for each m in groupmembers
        choices.additem m
    next
End Sub
```

Performance 7 – Membership of a Group

- There is another way
  - All the information exists in each ClearQuest database already
- So How do we Do It?
  - Use a ClearQuest Query
- For Optimum Performance
  - Apply the Caching Techniques we discussed

```
sub PopulateWithGroupMembers(GroupName, ChoiceList)

    ' This routine populates the choicelist provided with the ClearQuest
    ' login name from the user details records.

    Set MySession = GetSession

    set MyQuery = MySession.BuildQuery("users")
    MyQuery.BuildField("login_name") ' Column 1

    Set MyFilter = MyQuery.BuildFilterOperator(AD_BOOL_OP_AND)
    MyFilter.BuildFilter "groups", AD_COMP_OP_EQ, GroupName
    MyFilter.BuildFilter "is_active", AD_COMP_OP_EQ, 1

    ' Build and execute the query
    set MyResults = MySession.BuildResultSet(MyQuery)
    MyResults.Execute

    ' Process the results sets
    while MyResults.MoveNext = AD_SUCCESS
        ChoiceList.AddItem MyResults.GetColumnValue(1)
    wend

end sub
```

Performance 8 – Getting Data from other Records

- ClearQuest Provides Several Mechanisms for Getting Data from other records
  - Via GetEntity API call followed by GetFieldValue calls
  - A Query
  - Dotted Field Name Notation
    - only works for REFERENCE Fields (not REFERENCE_LIST)
- Lets Take a Look at Each method in turn
  - GetEntity – The **obvious/easy** choice !! – but not the easiest this time
    - Expensive - fetches lots of data – Does Separate Queries for
      - History
      - Attachments
      - Each REFERENCE_LIST Field (Duplicates counts as +1)

Performance 9 – Getting Data from other Records

- Scenario
  - Standard Defect Tracking Record “Defect”
    - Added a Reference to Defect called Child
    - Added a Reference_List to Defect called Children
  - Want to Fetch Two fields from another record
    - Headline, Priority

Performance 10 – Getting Data from other Records

- Example 1 – GetEntity

```
set ent = GetSession.GetEntity("defect", "SAMPL00000002")
x= ent.GetFieldvalue("headline").GetValue()
y= ent.GetFieldvalue("priority").GetValue()
```

- Results
  - Fetches the record itself
  - Does Query on History Table for referenced Record
  - Does a Query on the Attachments Table
  - Does a Query on the Child_Parent_Links Table and the referenced record types for each Reference_List field in the record (Duplicates counts as one)
  - So 3 + No. Reference List Fields Queries for the GetEntity
    - 6 Select Statements In this Scenario
    - Does this **every time** we do a GetEntity

Performance 11 – Getting Data from other Records

- Example 2 – Using Dotted Field Name Notation

```
x = GetFieldValue("child.headline").GetValue()
y = GetFieldValue("child.priority").GetValue()
```

- Results
  - Does the same number of queries as GetEntity !!
  - But
    - It only does them once!!
      - During the context of a single action, dotted field name notation does less overall queries than GetEntity if there is more than one call for the same record during a single action.
      - It's a lot less code

Performance 12 – Getting Data from other Records

- Example 3 – Using a Query

```
set MyQuery = GetSession.BuildQuery("Defect")
MyQuery.BuildField("headline")
MyQuery.BuildField("priority")
set MyFilter = MyQuery.BuildFilterOperator(AD_BOOL_OP_AND)
MyFilter.BuildFilter "id", AD_COMP_OP_EQ, "SAMPL00000002"
set MyResults = GetSession.BuildResultSet(MyQuery)
MyResults.Execute
MyResults.MoveNext
x= MyResults.GetColumnValue(1)
y= MyResults.GetColumnValue(2)
```

- Results
  - Only does 1 Query but does so every time it's used
  - Better to get all fields in one go
  - Really scores when details from multiple records required

Performance 13 – Getting Data from other Records

- Conclusions

Method	Easy of Use	Performance
GetEntity	Medium	Least
Dotted Field Notation	Easiest	Medium
Query 	Hardest	Best

- GetEntity & Dotted Field notation give very similar results in a single hook invocation example. But dotted field notation has the edge
- Reference Lists are relatively expensive since they are de-referenced when GetEntity is called.

Performance 14 - Tips

- Focus efforts on Areas that Take the Longest first
  - Leverage the CQ Diagnostic output to identify areas of concern
  - Don't fix it if it isn't broken!!
- Each Hook Invocation has to setup a Script Engine Session
- Look for ways to minimise the Number of hook invocations you have particularly if a lot of them fire at one time
  - Remove null hooks – they only add load – don't just comment them out
  - Lots of Field Permissions Hooks ?
    - Consider using an Action initialisation Hook instead (Maybe on a Base Action)
      - Action Initialisation Happens after Field Behaviours are Established
    - Don't go mad with Base Actions try and make maximum use of one per record
  - Lots of Field Initialisation Hooks
    - Consider putting them all in the Submit Actions Initialisation Hook

Performance 15 - Tips

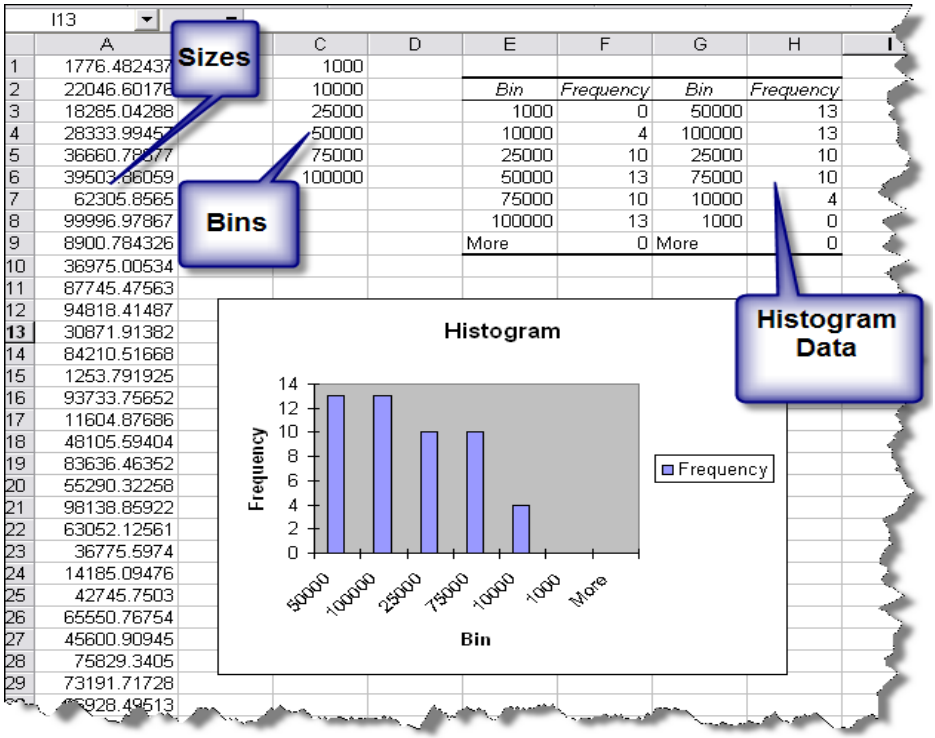
- Beware of Large Numbers of Email Rules
  - Particularly those that use a query
    - Can negatively impact record commit times
    - Can add a significant Database load
- Did you know that
  - Field Value Changed hooks Fire once each time the associated field value changes. That means **every** character typed in the field
  - Field Validation hooks Fire once each time **any** field has changed
    - They **all** fire during action initialisation to check existing value validity
  - Graphic Objects are stored as bitmaps – This makes them potentially big
    - All Downloaded during login – can be many MB
    - Look nice but add to login times
    - Keep Colour depth to a minimum
    - Windows Client will stretch bitmaps to fit the space available

Performance 16 – Tips - Database Indexes

- Databases with 50,000+ records, 250,000 history records
  - Can benefit from adding an additional Index to the History table
- See Technote #1127710 for details
  - ClearQuest **Versions Prior to V7.x only**
- create index history_entity_idx on owner.history (entitydef_id, entity_dbid);

Attachments – How Much Space Do they Occupy ?

- Probably the Biggest Things in Your Database and likely Take More Space than all your records
  - Analysis of RATLC – Our Own ClearQuest Database Showed 87% of space was occupied by Attachments
- Affects Backup / restore times
  - A single Bitmap Screen grab can be > 2MB!
- Can affect record fetch Times
- Analyse in Excel Using Analysis Toolpak Tools > Addins
- Import Attachment Data
- Create Ranges (Bins)
- Use Histogram Tool to Create Chart
- You could also analyse by file type



	A	C	E	F	G	H
1	1776.482437	1000				
2	22046.60176	10000	*Bin*	*Frequency*	*Bin*	*Frequency*
3	18285.04288	25000	1000	0	50000	13
4	28333.99457	50000	10000	4	100000	13
5	36660.78577	75000	25000	10	25000	10
6	39503.86059	100000	50000	13	75000	10
7	62305.8565		75000	10	10000	4
8	99996.97867		100000	13	1000	0
9	8900.784326		More	0	More	0
10	36975.00534					
11	87745.47563					
12	94818.41487					
13	30871.91382					
14	84210.51668					
15	1253.791925					
16	93733.75652					
17	11604.87686					
18	48105.59404					
19	83636.46352					
20	55290.32258					
21	98138.85922					
22	63052.12561					
23	36775.5974					
24	14185.09476					
25	42745.7503					
26	65550.76754					
27	45600.90945					
28	75829.3405					
29	73191.71728					
	65928.49513					

Attachments – How Can I Analyse Them ?

```
const AD_PRIVATE_SESSION = 2
Const AD_SUCCESS = 1
set MySession = Createobject("clearquest.session")
MySession.userlogon “<<admin-id>>", “<<password>>", “<<db-name>>", AD_PRIVATE_SESSION , “<<connection-name>>"
set FSO = CreateObject("Scripting.FileSystemObject")
set f = FSO.Opentextfile("attachment-sizes.txt", 2, true)
set MyQuery = MySession.BuildQuery("attachments")
MyQuery.BuildField("filesize")
set MyResults = MySession.BuildResultSet(MyQuery)
MyResults.EnableRecordCount
MyResults.Execute
Biggest=0:Smallest=999999999
f.write ("Number of attachments present " & MyResults.RecordCount & vbcrlf)
while MyResults.MoveNext = AD_SUCCESS
    ItemSize = clng(MyResults.getColumnValue(1))
    TotSize = TotSize + ItemSize
    if ItemSize > Biggest then Biggest = ItemSize
    if ItemSize < Smallest then Smallest = ItemSize
    f.write((MyResults.getColumnValue(1) & vbcrlf )
wend
f.write ("Total attachments size : " & TotSize & vbcrlf )
f.write(“Smallest=“ & Smallest & “, Biggest = “ & Biggest & vbcrlf)
f.close
msgbox “Complete"
```

Add A Filter to Find Bitmap file attachments

```
Const AD_COMP_OP_LIKE = 7
Const AD_BOOL_OP_OR = 2
set MyFilter = MyQuery.BuildFilterOperator(AD_BOOL_OP_OR)
Myfilter.BuildFilter "filename", AD_COMP_OP_LIKE ,".bmp“
```

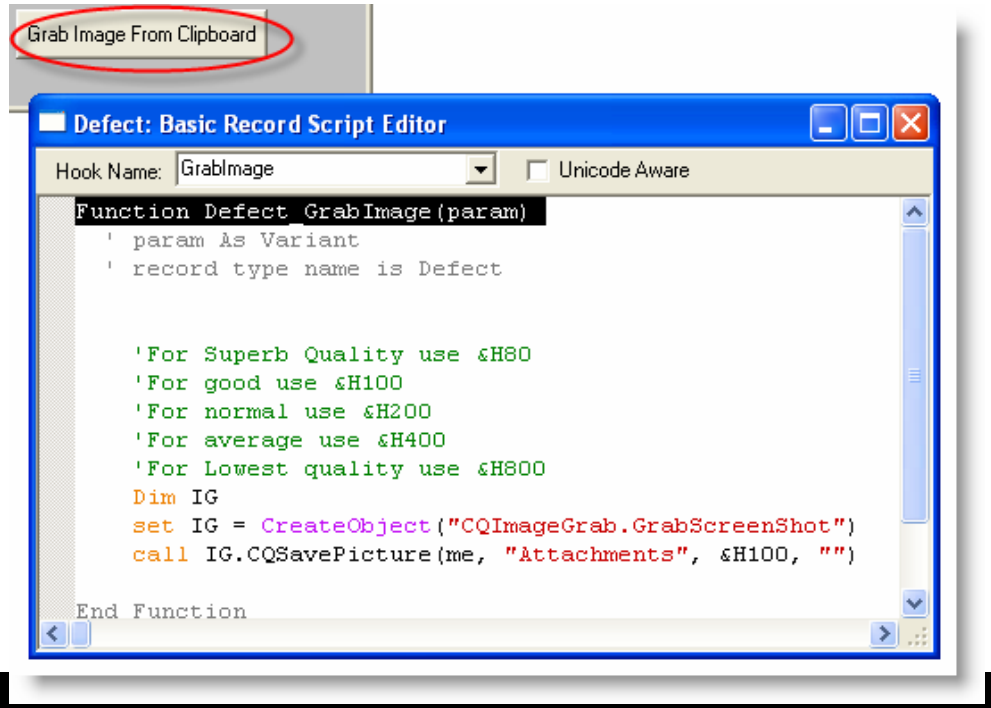
Attachments – 3 Limiting Attachment Size

- An Example Perl Script for Limiting the Maximum Size of Attachments

```
sub MaxFileSize{
    my $result;
    use File::stat;

    my $maxFileSize = 5000000; # 5MB

    # Get a list of the attachment fields in this record type...
    my ($AttachmentFields) = $entity->GetAttachmentFields();
    for (my($AF) = 0; $AF < $AttachmentFields->Count(); $AF++){
        my ($AttachmentField) = $AttachmentFields->Item($AF);
        my($Attachments) = $AttachmentField->GetAttachments();
        for (my($A) = 0; $A < $Attachments->Count(); $A++) {
            my($Attachment) = $Attachments->Item($A);
            $filename = $Attachment->GetFileName();
            $filesize = $Attachment->GetFileSize();
            # File size is always 0 before a first commit
            if ($filesize == 0){
                $fileinfo = -s $filename;
                if ($fileinfo > $maxFileSize){
                    $result = ""Cannot add attachments larger than 5MB."";
                    return $result;
                }
            }
        }
    }
    return $result;
}
```

Attachments – How Do I Grab Screen Shots as JPEGs?

- Need A Little External Help
  - FreeImage Project has a Free Image Conversion Library (dll)
  - Little bit of VB Wrapped in a dll to:
    - Grab a Bitmap Image from the Clipboard
    - Convert it to a JPEG – Interface to FreeImage
    - Save it as an attachment
- Schema Changes Required
  - Add a Button to a Form
  - Add a Record Script behind the button
- Downsides
  - Need to Deploy 2 DLLs to all Desktops
  - Won't work on Web



Questions

Thank You

Alan Murphy
IT Specialist - IBM Rational Brand Services
alan.murphy@uk.ibm.com

Rational. software

Code Samples

IBM Rational Software Development Conference 2007

What keeps me **Rational**?

Clone Record Global Script

```
' This global script contains code to assist in the 'cloneing' or copying of data
' from one record to another

sub CacheData(CacheName, FieldToCopyList, FieldToOutputList)

        Dim MySession
        set MySession = GetSession

        ' Save the values of the specified fields into a session name value since
        ' these are passed between engine instances.  Although you can store objects in
        ' namevalues, you cant pass objects between engine instances (this includes arrays)
        ' We're going to convert the array to a string and use a series of bell characters as
        ' to delimit them
        MySession.NameValue(CacheName) = Join(GetFieldStringValues(FieldToCopyList), string(5,7))

        ' Now we'll store the map of field names for the destination of these values
        MySession.NameValue(CacheName & "_MAP") = Join(FieldToOutputList, string(5,7))
end sub

sub ReplayCachedData(CacheName)

        Dim MySession, ValueList, FieldList
        set MySession = GetSession
```

Clone Record – Record Script

```
Function Defect_CopyToChild(param)
  ' param As Variant
  ' record type name is Defect

                ' This will be our new button pre-click hook
                ' It will setup the cache to copy fields from the current defect to the new one
                call CacheData("DEFECTDATACACHE", _
                                array("keywords","symptoms","Headline","Submitter","Version"), _
                                array("symptoms","keywords","Headline","Owner","note_entry"))

                ' Now say the cache is active
                GetSession.NameValue("CacheActive")= "TRUE"
End Function
```

Clone Record – Submit Action Initialisation Script

```
Sub Defect_Initialization(actionname, actiontype)
  ' actionname As String
  ' actiontype As Long
  ' action is Submit
  ' record type name is Defect

                ' This script looks to see if there are any cached values to initilaise with
                if GetSession.NameValue("CacheActive")= "TRUE" then
                                call ReplayCachedData("DEFECTDATACACHE")
                end if
                GetSession.NameValue("CacheActive")=""
End Sub
```

Useful ClearQuest Diagnostic Keys

```
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Rational Software\ClearQuest\Diagnostic]
"Trace"="DEBUG=1;LICENSE=22;DB_CONNECT=2;THROW;EMAIL;EMAIL_VB;API;SQL"
"Behavior"=""
"Report"="MESSAGE_INFO=0x1;DIAG_FLAGS=-1"
"Output"="ODS"
"EMailSendVB"="ODS"
"Name"=""
```

AdminEdit Code

```
sub AdminEdit(ExceptionList)

        ' This routine implements the AdminEdit pattern.
        ' Its purpose is to allow an administrator to edit almost any field
        ' in a record when the action that calls it is invoked.  Its main purpose
        ' is to recover form errors e.g. A picklist field has it valid picklist
        ' changed to remove an old value but when a recod using the old value is
        ' actioned and the field is read-only in the current state, it becomes
        ' impossible to correct the fault.
        '
        ' This routine makes all fields editable except for :
        ' a. Those fields CQ never lets you change e.g. STATE, History
        ' b. Those fields listed in the exceptionlist array parameter e.g. NotesLog
        '
        ' NOTE: If a field in the exception list is already editable, this routine
        ' doesn't over-ride that and make it non editable.
        '
        Dim MySession
        Set MySession = GetSession()

        ' get the list of fields for the current record type
        Dim EntityName
        EntityName = GetEntityDefName

        Dim MyEntityDef
```

Caching Populate ChoiceList with Group Members

```
sub PpopulateWithGroupMembersDetails(FieldName, GroupName, ChoiceList)

        ' This routine populates the choicelist provided with the requested
        ' field from the user details records.  If a groupname is provided then
        ' the users are limited to being a member of the named group
        ' The routine caches the list once it has been built

        Dim CacheName
        Dim CacheValue
        CacheName = "USERDETS::" & FieldName & "::" & GroupName
        CacheValue = ""
        Dim MySession
        Set MySession = GetSession
        if not MySession.HasValue(CacheName) then

                ' Build the cached details
                Dim MyQuery
                Dim MyFilter
                Dim MyResults
                set MyQuery = MySession.BuildQuery("users")
                MyQuery.BuildField(FieldName) ' Column 1

                ' If there is no group name dont bother with a filter we'll have all users
                if GroupName <> "" then
                        Set MyFilter = MyQuery.BuildFilterOperator(AD_BOOL_OP_AND)
```

Screen Grab Code

Code for Main Module of VB DLL – Other Declarations Required

FreeImage library can be obtained from

http://freeimage.sourceforge.net/

```
Option Explicit

 ' This is a wrapper for clipboard access and for image conversion.
 ' The module grabs an image from the clipboard and converts it to a compressed
 ' format and makes an attachment out of it.

 ' Most of the heavy lifting is done by this wrapper to minimise the work to be done
 ' by the schema designer.
 ' All that is required is a button with a record script to call this routine

 ' JPEG image Files are used in order to minimise the storage requirements of the images.

 ' Directory used to store cq attachments in
 Private Const AttachDir As String = "\Attachments\"

 ' JPEG Quality indicator.  The captured image quality is controllable via
 ' a property of the control interface.  The default quality is ' Normal' but others
 ' are available.  The lower the quality, the higher the compression and thus this
 ' directly affects the amount of space occupied by the image.  The schema designer can choose the
 ' quality of the captured images by calling this routine prior to the committing of an image
 ' This stays in effect for that instance of the control until changed
 Public Enum JPEG_Qualities
    JPEG_QUALITYSUPERB = &H80
    JPEG_QUALITYGOOD = &H100
    JPEG_QUALITYNORMAL = &H200
    JPEG_QUALITYAVERAGE = &H400
    JPEG_QUALITYBAD = &H800
 End Enum
```