



Siebel 8.0 Upgrade

Siebel VB to eScript Conversion Considerations

Robert Ponder

[rponder at ponderproserve.com](mailto:rponder@ponderproserve.com)

770.490.2767



Introduction

- What we have after running PPS Tools Helper© to convert a large body of Siebel VB to eScript.
- Main considerations between VB and eScript.
- How to approach the new ST engine



Big Differences Between VB and eScript

- Dates
 - String vs Date object
 - date.time vs. milliseconds since 1/1/1970
 - Implicit conversion of variant/Date to string
 - mm/dd/yyyy vs Wed April 11, 2007
- Function Return Values
 - Set fn name = value vs. return (value)
- Error Handler
 - Always executed vs. Only executed if error
 - eScript has finally block to destroy objects
- True/False
 - Integer vs. Boolean
 - <>0, 0 vs true, false
- Parameters
 - All by ref vs. Only complex objects by ref



Basic eScript Structure

```
function MyFunction()
{
    //comments
    //variable declarations
    try
    {
        //Normal code goes here
    }
    catch (e)
    {
        //Error code goes here
    }
    finally
    {
        //Object destroys go here
    }
}
```



How We Converted

```
function XYZFunction()
{
    //comments
    //variable declarations
    try
    {
        //Program statements are here
    }
    catch (e)
    {
        //ErrorHandler label and all following code is here
    }
    finally
    {
        //Object destroys from (just the) error handler are moved here
    }
}
```



We Can Re Run e2e If Needed

- Current process is to identify issues and as a team determine how to resolve consistently.
 - Logged as Unresolved Issue until resolution is determined
- Often produce a count to determine how frequent issue is.
- If occurrence is frequent then if possible may choose to automate changes instead of doing them manually.
- Nice to know we can change our minds on certain things and/or fix things that we missed during initial v2e conversion.



Strong Type (ST) eScript Engine

- Officially called ECMA Edition 4 Strong Type Compliant eScript.
- Allows vars, parameters and function returns to have datatypes just like VB allowed.
- Don't have to type vars to use ST engine but we currently have vars typed.
- If you script Siebel OOTB events the function template Tools builds for you will not have types for backward compatibility with the old "T" (type less) eScript engine.
- ST engine offers features such as Script Assist and much improved performance.
- Hit Ctrl + space in Tools to verify ST engine is in use.
- We all need to have our settings consistent and have it enabled.



Data Types (not many to choose from)

- Standard Siebel objects remain the same.
- VB data types change drastically.
 - Number: float
 - String: chars
 - Boolean: bool
 - Date object
- Our decision is to retain the VB types and strongly type all vars, parameters and function returns.
- We will use primitives of float, chars and bool instead of their corresponding object counterparts.



Application Level Functions

- New application level functions are used to make existing code compatible with eScript.
- Allows common functionality to be coded just once and then reused everywhere.
- Date function
 - VB Date[\$] returns a mm/dd/yyyy string representation of the current date.
 - TheApplication().Date() will do the exact same thing
- Date differences
 - If Date1 – Date2 > 1 Then
 - if (TheApplication().DateDiff(Date1, Date2) > 1)
- Format
- Trim



Passing By Reference

- Unlike VB, in eScript non complex / non objects data types are not passed by reference
- function Service_PreCanInvokeMethod (MethodName, &CanInvoke)
 - Have to use & to force by reference pass when value needs to be returned
- We have automatically added & on all String, Number, Boolean and Date parameters where needed
 - Added where parameter is also on left hand side of assignment statement.
 - Recursively worked way up call stack and added on calling functions too.



What We Did With return

- On PreCanInvoke methods converted function assignment to return statements.
- On functions that assign a value to the function name.
 - Created a strongly typed variable the same as the function name.
 - Added return of the function name var as very last statement in try block.
- In order for existing error handlers to work for certain functions will probably go with:
 - Single return statement as very last statement in function body just after finally block.
 - Throwing a “phoney” error at end of try to make all code enter catch
 - No other return statements (anywhere) must execute if this statement to execute
 - Also can not throw(e)/RET in catch block if you want to execute this statement
 - Can throw(e)/RET in try and still execute



Understanding return and What We Have Done With It

```
function XYZFunction(s1 : String) : String
{
    var XYZFuction : String;
    try
    {
        ...
        XYZFuction = "abc";
        ...
        return (XYZFuction);
    }
    catch (e)
    {
        throw(e);
    }
    finally
    {
        myPS = null;
    }
}
```